

Opportunities for the Deep Neural Network Method of Solving Partial Differential Equations in the Computational Study of Biomolecules Driven Through Periodic Geometries

by

Martin MAGILL

A thesis submitted to the School of
Graduate and Postdoctoral Studies in
partial fulfillment of the requirements for
the degree of

Doctor of Philosophy in Modelling and Computational Science

The Faculty of Science

University of Ontario Institute of Technology

Oshawa, Ontario, Canada

August 2022

Copyright © Martin MAGILL, 2022

THESIS EXAMINATION INFORMATION

Submitted by Martin MAGILL

Doctor of Philosophy in Modelling and Computational Science

Thesis title: Opportunities for the Deep Neural Network Method of Solving
Partial Differential Equations in the Computational Study of Biomolecules
Driven Through Periodic Geometries

An oral defense of this thesis took place on August 4th 2022 in front of the
following examining committee:

Examining Committee

Chair of Examining Committee	Dr. Daniel Hoornweg
Research Supervisor	Dr. Hendrick W. de Haan
Research Co-Supervisor	Dr. Ed Waller
Examining Committee Member	Dr. Faisal Z. Qureshi
Examining Committee Member	Dr. Lennaert van Veen
University Examiner	Dr. Kirk Atkinson
External Examiner	Dr. Colin Denniston, The University of Western Ontario

The above committee determined that the thesis is acceptable in form and content and that a satisfactory knowledge of the field covered by the thesis was demonstrated by the candidate during an oral examination. A signed copy of the Certificate of Approval is available from the School of Graduate and Postdoctoral Studies.

Abstract

As deep learning emerged in the 2010s to become a groundbreaking technology in machine vision and natural language processing, it also ushered in many new algorithms for use in scientific research. Among these is the neural network method, in which the solution to a differential equation is approximated by varying the parameters of a deep neural network trial function. Although this idea has been explored with shallow neural networks since the 1990s, it has experienced a resurgence of interest in recent years now that it can be implemented with deep neural networks. A series of empirical and theoretical studies have acclaimed the deep variants of the neural network method for being able to solve many classes of traditionally challenging partial differential equations. These early works emphasized its potential to solve high-dimensional, highly parameterized, and nonlinear equations in arbitrary geometries, all without requiring the discretization of the geometry into a mesh. Problems exhibiting these challenging features abound in computational biophysics, and this thesis presents recent efforts to adapt the neural network method for use in this field.

The investigations in this thesis center on models of biomolecular motion in

periodic geometries. Such models arise, for example, in the study of microfluidic and nanofluidic devices used for the separation of free-draining molecules. These problems exhibit many of the characteristics for which the neural network method is appealing, and serve here as non-trivial test problems on which to characterize its performance. Perspectives from biophysics, numerical analysis, and deep learning are combined to elucidate the true potential of the neural network method as a technique for studying such differential equations. Altogether, these works have moved the neural network method closer to being another reliable numerical method in the computational biophysicist's toolkit.

Keywords: Computational Science; Biophysics; Differential Equations; Neural Networks; Deep Learning

Declaration of Authorship

- I hereby declare that this thesis consists of original work of which I have authored. This is a true copy of the thesis, including any required final revisions, as accepted by my examiners.
- I authorize the University of Ontario Institute of Technology (Ontario Tech University) to lend this thesis to other institutions or individuals for the purpose of scholarly research. I further authorize Ontario Institute of Technology (Ontario Tech University) to reproduce this thesis by photocopying or by other means, in total or in part, at the request of other institutions or individuals for the purpose of scholarly research. I understand that my thesis will be made electronically available to the public.

Signed:

Martin Magill

Date:

17/Aug/2022

Statement of Contributions

This thesis contains published research articles, and most of this work was collaborative in nature. My contributions to each project are listed below. Regarding all the work in this thesis that has not been published previously, I hereby certify that I am the sole author; that I have used standard referencing practices to acknowledge ideas, research techniques, or other materials that belong to others; and that I am the sole source of the creative works and/or inventive knowledge described in these portions of the thesis.

- Chapter 3 contains the publication

Magill, Martin, Ed Waller, and Hendrick W. de Haan. “A sequential nanopore-channel device for polymer separation.” *The Journal of Chemical Physics* 149.17 (2018): 174903.

I am first author of this work, and I was primarily in charge of the research described therein. The work was conducted under the joint supervision of and in collaboration with Profs. Hendrick de Haan and Ed Waller, who assisted in placing the ideas in the context of relevant literature, structuring the organization of the research and its presentation in the published article, and guiding the high-level direction of the research. The original research question motivating the work emerged in part from previous work by

Prof. Hendrick de Haan. I was primarily responsible for the implementation and analysis of the numerical simulations, the development and mathematical analysis of the multiscale modelling approach, the conception and execution of the analyses, and the writing of the article itself.

- Chapter 4 contains the publication

Magill, Martin, Andrew M. Nagel, and Hendrick W. de Haan.

“Neural network solutions to differential equations in nonconvex domains: Solving the electric field in the slit-well microfluidic device.” *Physical Review Research* 2.3 (2020): 033110.

This work was conducted in close collaboration with Andrew Nagel and under the supervision of Prof. Hendrick de Haan. I am first author of this work, and was primarily responsible for posing the research questions explored in the work, conceiving of and implementing most of the numerical analyses in the work (except the particle simulations, for which Andrew was primarily responsible), developing the theoretical analyses, and writing of the article itself. The idea for a mesh-agnostic measure of error in flux conservation was conceived in close partnership with Andrew Nagel. I was also primarily in charge of positioning the work within related literature.

Appendix B contains some mathematical proofs related to this project that have not been published previously. These were originally conducted by myself, based on a suggestion by my colleague Ermal Rrapaj to try using the method of Green functions in this context.

Appendix D contains the publication

Nagel, Andrew M., Martin Magill, and Hendrick W. de Haan.
 “Studying First Passage Problems using Neural Networks: A Case
 Study in the Slit-Well Microfluidic Device.” *Physical Review E*
 106.2 (2022): 025311.

It is a continuation of the same research project with Andrew Nagel and Prof. Hendrick de Haan. Again this work was conducted in particularly close collaboration with Andrew Nagel, but in this case with Andrew taking the role of first author. I was nonetheless very heavily involved in most aspects of the work: posing the research questions, conceiving of appropriate numerical analyses, positioning the work within related literature, and writing the article. This work is discussed in Chapter 6.

Chapter 6 contains the manuscript entitled *On parallel computing for mobilities in periodic geometries*, which is soon to be submitted for publication. This is a culmination of the research project with Andrew Nagel and under the supervision of Prof. Hendrick de Haan. Again, I was primarily in charge of most aspects of the research, including the discussion of related literature, the conception of most numerical analyses, the theoretical derivations, and the writing of the article itself. Responsibility for undertaking the numerical analyses was shared equally between Andrew and I.

- Chapter 5 contains the publication

Magill, Martin, Faisal Qureshi, and Hendrick de Haan. “Neural networks trained to solve differential equations learn general

representations.” *Advances in Neural Information Processing Systems* 31 (2018).

I am first author of this work, and I was primarily in charge of the research described therein. The work was conducted under the supervision of Prof. Hendrick de Haan and in collaboration with Prof. Faisal Qureshi. Prof. Faisal Qureshi provided guidance on how to organize the research project into a cohesive publication that would be of interest to the machine learning community. Both Professors provided high-level guidance for the research direction and analysis approaches. I was responsible for posing the research questions, conceiving of and implementing and the numerical analyses, positioning the work within related literature, and writing the article itself.

Appendix C contains an extended abstract entitled “Compact Neural Network Solutions to Laplace’s Equation in a Nanofluidic Device”, describing related research conducted as part of the same research collaboration. It was presented at the 2018 Workshop on Compact Deep Neural Network Representations for Industrial Applications (CDNNRIA) by Magill, Martin, Faisal Z. Qureshi, and Hendrick W. de Haan. I was primarily in charge of all aspects of that work, which eventually motivated the subsequent work in Chapter 4.

In addition to the above manuscripts which were included in this thesis document, I have collaborated on the following related works during my doctoral studies.

- Regarding the work published as:

Briggs, Kyle, Gregory Madejski, Martin Magill, Konstantinos Kastritis, Hendrick W. de Haan, James L. McGrath, and Vincent Tabard-Cossa. “DNA translocations through nanopores under nanoscale preconfinement.” *Nano Letters* 18.2 (2018): 660-668.

I was primarily responsible for conducting the simulation work and writing the corresponding sections of the manuscript. I also assisted with the experimental data analysis included in the supplemental analysis (for identifying and excluding outliers in the experimental data).

- Regarding the work published as:

Lam, Michelle H., Kyle Briggs, Konstantinos Kastritis, Martin Magill, Gregory R. Madejski, James L. McGrath, Hendrick W. de Haan, and Vincent Tabard-Cossa. “Entropic trapping of DNA with a nanofiltered nanopore.” *ACS Applied Nano Materials* 2.8 (2019): 4773-4781.

I assisted with the simulation work and edited the corresponding sections of the manuscript.

Each day,
Hike in unknown woods.
Each night,
A fireside chat,
Stars listening above.

Acknowledgements

The work in this thesis would not have been possible without the guidance and support of my supervisors Profs. Hendrick de Haan and Ed Waller. I am particularly grateful that they kept me safe as I drifted from my original research direction into the exciting emerging field of scientific machine learning. Hendrick went above and beyond in this respect, going so far as to join me in co-founding our startup Altrina AI and roping in Andrew Nagel as well.

Thanks also to all the other faculty at Ontario Tech University, past and present, with whom I've had the pleasure of interacting during my studies. Special thanks to Prof. Jane Breen for her excellent course of nonnegative matrices, which prompted many of the ideas for the work in Chapter 6.

I would also like to thank all the collaborators I have had the pleasure of working with during my time as a graduate student. Working with Prof. Vincent Tabard-Cossa's group at the University of Ottawa was always enjoyable and enlightening; although those collaborative works are not attached in this thesis, they are thematically linked and motivated many of the ideas herein. The work in Chapter 5 was a direct outcome of Prof. Faisal Qureshi's graduate course on machine learning and his subsequent direction and assistance.

Generous funding was provided for this work by the Ontario Graduate Scholarship program. Additional funding and computing hardware was provided

by Mitacs in partnership SOSCIP. Special thanks to the Brilliant Incubator and all its wonderful staff who, in addition to starting me on an entrepreneurial adventure, supported Altrina AI in obtaining this research grant. Additional thanks to Prof. Lennaert van Veen for his role in making that grant possible.

I am also very grateful to the Vector Institute. In addition to hosting networking opportunities, professional development resources, and local research conferences, its Postgraduate Affiliate program provided me with additional support and funding. I would also like to thank Ermal Rrapaj, Luca Celotti, and all the other colleagues I have met through the Vector Institute, our scientific machine learning reading group, or the junior-researcher-led Scientific Machine Learning (SMaL) research community that emerged from those connections.

Thanks to all of the members of Prof. de Haan's cNAB.LAB for friendship, support, and stimulating conversations. A particular thank you to Mohammad Hassan Khatami for his mentorship during and after his time as a postdoctoral fellow in the lab. A very special thank you to Andrew Nagel, with whom I have worked very closely these last few years—I could not have asked for a better colleague and friend to get through the pandemic era.

Finally, I would like to thank all of the other friends and family who have made it possible for me to spend so many years of my life cloistered away in the ivory tower. In particular, this thesis is dedicated to Amber Maharaj, who spent countless evenings and weekends discussing the academic and commercial idiosyncrasies of the neural network method, and who was always available for practice run-throughs of presentations or last-minute proofreading. Besides, if not for her I would surely have starved or gone mad before finishing this dissertation.

Contents

Abstract	iii
Declaration of Authorship	v
Statement of Contributions	vi
Acknowledgements	xii
1 Introduction	1
1.1 Computational biophysics	2
1.2 Microfluidic and nanofluidic devices	5
1.3 Numerical methods meet deep learning	6
1.3.1 Introduction to the neural network method	9
2 Mathematical models	13
2.1 Models of biological macromolecules	13
2.2 Implicit solvent models: Langevin dynamics	16
2.3 Electric field models: Laplace’s equation	20
2.3.1 Drift velocity, effective charge, and mobility	22
2.4 Modelling ensembles: The Smoluchowski equation	24

2.5	Modelling first passage processes: The time-integrated Smoluchowski equation	27
3	A sequential nanopore-channel device for polymer separation	33
3.1	Motivation	34
3.2	Results	38
3.3	Manuscript	39
4	Neural network solutions to differential equations in nonconvex domains: Solving the electric field in the slit-well microfluidic device	59
4.1	Motivation	60
4.2	Results	64
4.2.1	Irregularity is a bottleneck	64
4.2.2	Convergence of error with loss	66
4.2.3	No pathologically unphysical solutions	67
4.2.4	Convergence of loss with capacity	69
4.3	Manuscript	72
5	Neural networks trained to solve differential equations learn general representations	87
5.1	Motivation	88
5.2	Results	89
5.2.1	Intrinsic dimensionality	89
5.2.2	Reproducibility and specificity	91
5.2.3	Interpreting the features	92

5.3	Manuscript	92
6	On parallel computing for mobilities in periodic geometries	110
6.1	Motivation	111
6.2	Results	116
6.2.1	Direct and indirect mobilities are equivalent	116
6.2.2	Indirect mobility better exploits parallel computing	117
6.3	Manuscript	120
7	Conclusions	146
	Bibliography	151
A	Numerical methods	157
A.1	Particle simulations	157
A.1.1	Basic algorithms	157
A.1.2	Rates of convergence	159
A.1.2.1	Timestep	159
A.1.2.2	Geometry	160
A.1.2.3	Ensemble size	161
A.1.2.4	Many-body systems	161
A.2	The finite element method	164
A.2.1	Basic algorithms	164
A.2.1.1	Mesh generation	165
A.2.1.2	The method of weighted residuals	166
A.2.1.3	Mixed formulation	171

A.2.1.4	Model order reduction	173
A.2.2	Rates of convergence	176
A.3	The neural network method	186
A.3.1	Background	186
A.3.2	Basic algorithms	187
A.3.2.1	Neural network architectures	189
A.3.2.2	Training neural networks	193
A.3.2.3	Loss functionals	195
A.3.2.4	Regarding generalization	196
A.3.2.5	Model order reduction with the NNM	197
A.3.3	Rates of convergence	198
A.3.3.1	Expressivity	200
A.3.3.2	Trainability	202
B	Error bounds using Green functions	206
C	Compact neural network solutions to Laplace’s equation in a nanoflu- idic device	209
D	Studying first passage problems using neural networks: A case study in the slit-well microfluidic device	215

List of Abbreviations

BC	B oundary C ondition
BD	B rownian D ynamics
BVP	B oundary V alue P roblem
CG	C oarse- G ained
CGBD	C oarse- G ained B rownian D ynamics
CGLD	C oarse- G ained L angevin D ynamics
CNN	C onvolutional N eural N etwork
DENN	D ifferential- E quation(- S olving) N eural N etwork
DNA	D eoxyribonucleic A cid
dsDNA	D ouble- S tranded D eoxyribonucleic A cid
FEM	F inite E lement M ethod
FENE	F initely E xtensible N onlinear E lastic
FPP	F irst P assage P rocess
FPT	F irst P assage T ime
GPU	G raphics P rocessing U nit
i.i.d.	I ndependent and I dentically D istributed
LD	L angevin D ynamics
MCMC	M arkov C hain M onte C arlo
MD	M olecular D ynamics

MFPT	M ean F irst P assage T ime
MNFD	M icro- (or) N anofluidic D evice
MOR	M odel O rders R eduction
NN	N eural N etwork
NNM	N eural N etwork M ethod
ODE	O rdinary D ifferential E quation
PDE	P artial D ifferential E quation
ReLU	R ectified L inear U nit
ROM	R educed O rders M odelling
SDE	S tochastic D ifferential E quation
SMaL	S cientific M achine L earning
SVCCA	S ingular V ector C anonical C orrelation A nalysis
WCA	W eeks- C handler- A nderson

Chapter 1

Introduction

The document is a compendium of research articles exploring the use of the deep neural networks to solve partial differential equations (PDEs) arising in computational nanobiophysics. Specifically, this thesis is focused on models of biomolecules confined in periodic geometries. Such problems are important to the research and development of micro- and nanofluidic devices (MNFDs; Sec. [1.2](#)), and also arise in the study of natural biological systems.

This chapter will set out the high-level motivation for this line of research. Section [1.1](#) contains an overview of the role of computational nanobiophysics in understanding natural and synthetic biological systems at the molecular scale. Section [1.2](#) features a brief review of MNFDs, and especially the important subclass of MNFDs with periodic geometries. Lastly, Sec. [1.3](#) makes the case for deep learning as a new tool in the computational nanobiophysics toolkit and includes an introduction to the neural network method of solving differential equations.

Chapter [2](#) provides an introduction to the specific mathematical models that will be studied throughout the thesis. Appendix [A](#) contains a review of the

numerical methods that are used throughout this thesis to solve these mathematical models. The remaining chapters of this thesis present several research articles related to the computational study of periodic MNFDs. In addition to examining the use of deep learning for solving PDEs in computational nanobiophysics, these works also contain some contributions relevant to biophysics outside the context of deep learning (Chapters 3 and 6) and to deep learning outside the context of biophysics (Chapter 5).

1.1 Computational biophysics

Biophysics is a broad field encompassing the study of physical concepts, such as statistical mechanics and electrodynamics, in the setting of biological systems, both natural and synthetic. This thesis will focus on biophysics at the molecular scale and in the context of engineered devices, although many of the discussions may be of interest more generally. In any case, all areas of biophysics have the potential to contribute greatly to human understanding of natural biological functions as well as to stimulate and guide research and development of biotechnologies.

Biological phenomena are notoriously complicated. Natural biological systems have evolved complex mechanisms built of interdependent components ranging in size from the atomic scale to the macroscopic scale of normal human experience. In particular, many of the remarkable biological features we can easily witness at the scale of millimeters to meters actually emerge from structures that are too small for humans to witness unassisted. For instance, biological cells are fundamental components of all life, but remained undiscovered until the

mid-nineteenth century. Cellular theory marked a pivotal revolution in biology, and was enabled by the use of optical microscopes. Molecular biology marked an equally fundamental development in biology, and this paradigm shift was enabled in part by the use of technologies that can probe scales smaller than the optical resolution limit of light¹.

Today, technologies for sensing, analysing, and manipulating biological systems at scales below the optical resolution limit continue to play a crucial role in molecular biology. Super-resolution microscopy, atomic force microscopy, and microscopy based on x-rays, electrons, or even neutrons all play vital roles in making measurements of these systems. Microfluidic and nanofluidic devices, discussed below, offer another avenue for studying small biological elements.

Nonetheless, many of these technologies generally remain relatively challenging to build and operate. More importantly, even the best varieties and qualities of equipment still face intrinsic limitations as to the information they can measure in practice. Dynamics at very small length scales also tend to occur over very brief time scales, making them challenging to measure in real time. At some point, invasive measurements techniques will disrupt or destroy the systems under study. For instance, cryogenic electron microscopy requires samples be frozen at temperatures that are not hospitable to most forms of life. Conversely, biological systems can themselves be dangerous, and potentially disrupt or destroy the humans attempting to study them! Laboratory biosecurity incidents remain an

¹The Abbe diffraction limit prevents classical optical instruments from resolving features that are less than a few times smaller than the wavelength of the light being used by the device. For visible light, the practical resolution limit is roughly 250 nanometers. For reference, cells are typically at least a few micrometers in diameters, but viruses are usually about 100 nanometers across. Proteins are about 10 nanometers large, and other important biomolecules are even smaller.

important hazard even in the modern world. Typically, as a result of these and other factors, measured data is expensive, noisy, and limited to one or a few modalities at once.

In light of this, theoretical and numerical techniques play a fundamental role in advancing molecular biophysics. The use of mathematical modelling can greatly extend the value of experimental data measurements. Abstractions of true biological systems can be used to identify platonic ideals that underlie general phenomena (e.g., Brownian motion) or to rapidly predict biophysical quantities that are easier to calculate than to measure (e.g., alchemical techniques in drug design). Indeed, computational biophysics is often described as a computational microscope: another technique that the biophysicist can leverage to study phenomena that are too small or rapid for humans to witness directly.

Just as the experimental biophysicist has access to a range of experimental tools, some of which are more appropriate than others for certain applications, so too does the computational biophysicist have recourse to a diverse toolkit of models and methods with varying strengths and limitations. The relative merits of mathematical models and numerical methods must generally be considered simultaneously. A very high-fidelity model is of little use if it cannot be solved or analysed. Conversely, a more modest model coupled to a fast and flexible numerical method may provide substantial practical insight in many applications. Chapters [2](#) and App. [A](#) of this thesis are devoted to reviewing some common biophysical models and methods, respectively. The main goal of this thesis is to explore the neural network method (NNM; Secs. [1.3](#) and [A.3](#)) as a new tool for this toolkit, and to establish how its advantages and disadvantages compare to

and complement well-established techniques like particle simulations (Sec. [A.1](#)) and the finite element method (FEM; Sec. [A.2](#)).

1.2 Microfluidic and nanofluidic devices

The investigations in this thesis are focused on models of micro- and nanofluidic devices (MNFDs). These are engineered systems that can be used to detect, analyse, and manipulate biomolecules with high precision. These devices are even sometimes capable of interacting with individual molecules one at a time.

The capabilities of these systems are due in great part to their intricate geometric structures at the micron and nanometer scale. They typically function by forcing confined macromolecules to interact with obstacles of varying shape, with motion commonly driven by fluid pressure gradients or electrostatic fields. The fluid flows, electrostatic fields, and diffusive trajectories of the confined macromolecules interact to produce the mechanisms that make the MNFD useful.

MNFD research involves many of the challenges that are also encountered more generally throughout computational biophysics. The phenomena of interest emerge due to the nonlinear coupling of multiple underlying physical processes (e.g., solvent flows, electrostatics, and molecular motion). The many-body dynamics of macromolecules, in particular, is very challenging to analyse. The complexity of the problem is further increased by the intricacy of the confining geometry. Ultimately, the goal of MNFD research is generally to understand how all of these physical phenomena depend on the relevant MNFD design parameters, such as device shape, solvent composition, applied voltage, and macromolecular

characteristics. Similar features are important throughout synthetic and natural molecular biology, arising for instance in the crowded intracellular environment.

The studies in this thesis have focused primarily on a few representative MNFDs. Nanopores, and the translocation of polymers through nanopores, is discussed in Chapter 3 as part of a novel MNFD design. The slit-well MNFD is featured heavily in Chapters 4 and 6, as well as Apps. C and D. The work in Chapter 6, in particular, aims to connect the results of this thesis to all MNFDs that have periodic geometric designs.

1.3 Numerical methods meet deep learning

As alluded to above, the mathematical models arising in biophysics tend to exhibit several properties that make them particularly challenging to study analytically or numerically:

- Intricate geometries,
- Many-body molecular dynamics,
- Nonlinear interactions, and
- Many interdependent physical parameters.

As discussed in Chapters 2 and App. A, these features have directly shaped contemporary techniques in computational biophysics. Currently, the dominant paradigm entails modelling molecular dynamics using stochastic differential equations and varying levels of coarse-graining, then simulating the time evolution of the system using molecular dynamics algorithms. Complicated

interactions are approximated as much as possible, or else calculated dynamically throughout the particle simulations.

In particular, partial differential equation (PDE) models of molecular dynamics are not presently very popular in computational biophysics. The PDE models corresponding to many-body molecular motion are high-dimensional. Such PDEs are traditionally considered intractable because classical numerical methods, such as the finite element method, suffer from the curse of dimensionality: their computational cost grows exponentially with the dimensionality of the domain. Indeed, particle-based simulation methods are used precisely in order to circumvent the need to solve high-dimensional PDEs directly. PDE models of molecular motion are thus mostly restricted to applications where simplifications like the mean field approximation can be made.

PDE models can also be used to describe force fields in biophysics, including solvent flow fields and electrostatic fields. These fields are often coupled to the motion of the molecule: as a molecule moves, it deforms the fluid flows and electrostatic fields around it. As a result, the corresponding PDE models depend on the many degrees of freedom specifying the molecular conformation, and are once again high-dimensional. These PDE models can still sometimes be solved for one molecular configuration at a time during a molecular dynamics simulation. However, this can substantially increase the computational cost of the simulations, which can be large to begin with. Again, many simplifications are often used in practice, and there even exist particle-based methods for circumventing PDE models of force fields (e.g., dissipative particle dynamics are used to approximate solvent flows).

This thesis explores the possibility of directly solving these types of biophysical PDEs by using deep neural networks. This is an example of an innovative numerical method (i.e., the NNM) potentially unlocking the practical use of an otherwise unfavoured class of mathematical models (i.e., PDEs). In contrast with classical techniques like the finite element method, the neural network method is mesh-free. In particular, it does not appear to suffer from the curse of dimensionality, and a growing number of theoretical and empirical studies have demonstrated its ability to solve various high-dimensional PDEs. Furthermore, it is reputed to be an effective tool for handling intricate geometries (as it is mesh-free) and nonlinear problems (as it is intrinsically nonlinear). It can even solve highly parameterized problems directly² (see App. D). Ultimately, the hope is that this methodology can leverage the strengths of PDE models in a context that is usually only accessible to particle-based models.

The caveat to this vision is that the neural network method of solving differential equations is a relatively new method that is not yet fully understood. It lacks a thorough theoretical foundation comparable to those available for well-established numerical methods, and it is unclear to what extent the empirical demonstrations available in the literature are representative of its performance on biophysical problems. This thesis aims to improve the understanding of when and how the neural network method can be applied fruitfully in computational biophysics. The results included here suggest that, although further development work is necessary, it is indeed feasible that the neural network method can provide

²Model order reduction methods can provide similar functionality for classical PDE solution methods such as the finite element method, but these have limitations; see Sec. A.2.1.4. There is no straightforward equivalent to this for particle-based simulations.

useful capabilities that prior numerical methods cannot deliver. Several applied and theoretical contributions are presented as steps towards realizing this goal.

1.3.1 Introduction to the neural network method

This section provides a brief introduction to the basic elements of the neural network method (NNM) of solving differential equations. Additional details are available in the manuscripts attached in Chapters 4 and 5 and in App. D. Furthermore, App. A contains a more comprehensive review of the NNM along with reviews of particle-based simulation methods and the finite element method.

The basic idea of the NNM is to train a deep neural network to approximately satisfy a target PDE. The hope is that if the neural network approximately satisfies the PDE, then the neural network will approximately equal the solution. That is, for a PDE of the form

$$Lu(x) = f(x), \quad x \in \Omega \quad (1.1)$$

with boundary conditions

$$Bu(x) = g(x), \quad x \in \partial\Omega, \quad (1.2)$$

we define an approximate solution

$$\tilde{u} \equiv N(x; \theta) \quad (1.3)$$

where the trial function $N(x; \theta)$ is a deep neural network parameterized by θ .

Many varieties of deep neural networks exist (Sec. A.3), but essentially a deep neural network is just any parameterized function that is the composition of many simpler functions. The works in this thesis will focus on fully-connected feedforward neural networks, which are of the form

$$\begin{aligned}
 f_1 &= \sigma_1 (A_1 x + b_1), \\
 f_2 &= \sigma_2 (A_2 f_1 + b_2), \\
 f_3 &= \sigma_3 (A_3 f_2 + b_3), \\
 &\dots \\
 f_L &= \sigma_L (A_L f_{L-1} + b_L), \\
 N(x; \theta) &= c \cdot f_L + d.
 \end{aligned} \tag{1.4}$$

Each f_i is known as a hidden layer of the neural network; the number of hidden layers L is called the depth of the network. In the special case of $L = 1$, the network is called a shallow neural network; for $L > 1$, it is a deep neural network. The A_i are called weight matrices and the b_i are called bias vectors. The shapes of A_i and b_i are dictated by the desired sizes of the f_i ; these are often fixed to some common value w , called the width of the network. The σ_i are known as activation functions; these are scalar nonlinear functions that are applied elementwise to their inputs. In this case, the network output $N(x; \theta)$ is a scalar value of x obtained by projecting the last hidden layer f_L onto the vector c and adding the scalar bias d . Altogether, the parameters θ of this network are the A_i , the b_i , c , and d ; all activation functions in this thesis are chosen as the hyperbolic tangent function, \tanh .

The goal of the NNM is to find parameters θ such that the neural network is approximately equal to the true solution of the PDE. Unfortunately, the true solution is not known *a priori*. Hopefully, if we can find parameters such that

$$L\tilde{u} \approx Lu = f, \quad x \in \Omega, \quad (1.5)$$

and

$$B\tilde{u} \approx Bu = g, \quad x \in \partial\Omega, \quad (1.6)$$

then it will be the case that

$$\tilde{u} \approx u, \quad (1.7)$$

i.e., the deep neural network will be an approximate solution to the PDE. In practice, this can be accomplished by optimizing the neural network parameters θ to minimize a loss functional, such as

$$\mathcal{L}[N(x; \theta)] = \int_{\Omega} (LN(x; \theta) - f(x))^2 dx + \beta \int_{\partial\Omega} (BN(x; \theta) - g(x))^2 ds. \quad (1.8)$$

The derivatives of N can be computed using automatic differentiation, and the optimization of θ to minimize the loss functional is typically accomplished using variants of gradient descent (Sec. A.3).

This optimization process is referred to as training the neural network. Although training algorithms are usually fairly simple to describe, the actual training process is generally quite challenging. The optimization is nonlinear and

nonconvex, and many heuristics are leveraged to improve the speed and reliability of convergence. Despite these tricks, very long training times can sometimes be required to achieve good performance.

With a very simple modification, the NNM can also directly solve parameterized PDE problems, i.e. those that depend on some problem parameters p . For instance, the NNM trial function can be adjusted to the form

$$\tilde{u}(x, p) \equiv N(x, p; \theta), \tag{1.9}$$

and the original loss functional can simply be averaged over all feasible values of p . Besides this, most of the NNM algorithm is exactly the same as in the unparameterized case. This approach is reviewed in Sec. [A.3.2.5](#) and its application to MNFD design is discussed in Chapter [6](#) and demonstrated in App. [D](#).

Chapter 2

Mathematical models

2.1 Models of biological macromolecules

Macromolecules are, as the name suggests, conspicuously large molecules. Four classes of macromolecules are ubiquitous in biological systems: nucleic acids (including DNA, RNA, and their variants), proteins, carbohydrates, and lipids. These large molecules are primarily organized as polymers: they are constructed from a large number of simpler molecular units, called monomers. A single macromolecule can contain thousands of atoms, with constituent monomers composed from as little as one single atom (carbon being the base unit for lipid tails) to dozens of atoms. Many of the phenomena of interest in molecular biophysics involve the interactions of macromolecules with other macromolecules, solvent environments, confining boundaries, and electric fields.

At a very fine level of physical realism, macromolecules are quantum mechanical many-body systems described by some high-dimensional Schrödinger equation based on some hideously complicated Hamiltonian. Such fine-grained models are tremendously expensive to solve directly by any computational means.

Occasionally, enormous amounts of computing power are leveraged to simulate systems near this level of detail explicitly, but even then such simulations are inevitably restricted to evolutions over very short timescales.

Far more commonly, a variety of coarse-graining and multiscale modelling methods are used to produce mathematical models of greatly reduced complexity that nonetheless capture the essential qualitative (and sometimes quantitative) properties of a given macromolecular system. In some cases mathematical tools such as renormalization group theory can be applied to systematically construct or justify multiscale or coarse-grained models¹. Modern classical atomistic molecular dynamics formulations treat the atoms of macromolecules as classical particles whose motion is essentially governed by Newton’s second law. Problem-specific force fields are developed and calibrated numerically to approximate the true quantum mechanical interactions of these systems. Within a certain range of applicability, these effective force fields can produce fairly realistic physical behaviours. These bottom-up approaches to coarse-graining endeavour to systematically approximate physical interactions at small length scales by simpler effective interactions acting over longer length scales.

In contrast to bottom-up methods, the work in this thesis will focus on top-down modelling methods. In this approach, macromolecular systems are simply represented by groups of featureless hard spherical particles (for an example, see Fig. 1(a) in Sec. 3.3). Each of these particles may loosely be understood to correspond to large groups of atoms in the original molecule being modelled; otherwise, the model may simply be interpreted as a generic

¹The work of Michael Levitt and Ariel Warshel in the 1970s culminated in a Nobel prize towards the development of such techniques.

abstraction of a certain class of molecules under investigation. Dynamics are governed by variants of Newton’s second law,

$$m_i a_i = \sum_j F_{ij}, \quad (2.1)$$

where m_i is some effective mass for particle i , a_i is its acceleration, and the index j iterates over all forces F_{ij} that act on particle i . Complex polymer shapes can be created by connecting particles into a given topological arrangement using simple forces to model chemical bonds. Additional forces can be incorporated to reflect other important phenomena, such as solvent dynamics (Sec. 2.2) and electric force fields (Sec. 2.3). The manuscript in Sec. 3.3 features a typical example of how such top-down models can be applied to the study of MNFDs. Similar models were used successfully to study experimental systems in the works by Briggs et al. [5] and Lam et al. [24], of which I am a co-author.

Top-down coarse-grained models typically omit many physical details occurring in actual macromolecular systems, including chemical properties such as charge groups. They are nonetheless appropriate for studying many real nanobiophysical systems of interest. Indeed, in many cases the most important dynamics of macromolecules are governed predominantly by their geometries, with chemical details being of secondary importance. Conversely, important phenomena often occur on relatively long time scales. Fine-grained models that incorporate more chemical information are inevitably more computationally expensive to study than the simpler generic polymer models. Often, the loss of information from omitting chemical properties is more than offset by the ability to investigate dynamics spanning much longer timescales. That is, studying less realistic

molecular models over more relevant time scales can be more useful than studying more realistic molecular models over less relevant time scales. Another advantage of generic polymer models is that they are more amenable to theoretical analysis. Moreover, because they are generic, these models and their associated theories can capture general polymeric behaviours that are exhibited, at least approximately, by a great variety of macromolecules.

Of course, in some circumstances chemical properties and other fine-grained details cannot be neglected. Nanobiophysical models must typically be adapted to the specific phenomena under consideration to achieve a balance of precision and tractability. The work in this thesis focuses on the most tractable models because it is aimed at developing a firm understanding of the mathematical and numerical properties of a novel computational tool: the neural network method of solving differential equations. However, this technique can in principle be extended to more sophisticated macromolecular models as well. We hope that the insights gleaned here from the study of simpler models will be of value in those contexts as well.

2.2 Implicit solvent models: Langevin dynamics

The liquid environments of biological systems play a fundamental role in determining macromolecular dynamics. On the one hand, macromolecules are in constant chaotic motion due to perpetual collisions with the thermal bath of solvent molecules in which they are dissolved. On the other hand, the solvent also acts to rapidly damp out inertial motion through frictional forces. The combination of these effects produces the famous Brownian motion.

Beyond this, solvents can also propagate more subtle hydrodynamic interactions, wherein forces are transferred between nearby particles via disturbances in the intervening solvent. Additionally, the liquids of biophysical systems almost always contain substantial amounts of dissolved ions, and biological macromolecules themselves are frequently charged. Thus, electrohydrodynamic effects are also important to these systems.

As with molecular modelling, solvents can also be modelled at varying levels of detail. The discussion from Sec. 2.1 concerning the hierarchy of models ranging from quantum mechanical precision to heavily coarse-grained approximations applies equally well to solvents. This thesis will again adopt a more minimalist, top-down approach to solvent modelling. Specifically, the large number of molecules constituting the solvent will be treated implicitly as a thermal bath of generic, mutually independent particles.

In the implicit model used here, known as Langevin dynamics, the solvent effectively exerts two forces on the particles in solution. The first force is a frictional drag force,

$$F_f = -\gamma v, \tag{2.2}$$

where γ is a positive constant called the friction coefficient and v is the particle velocity. The force is linear in the particle velocity due to the laminar nature of fluids at the length scales of interest. In this Stokesian flow regime, the friction

coefficient for a spherical particle can be written as

$$\gamma = 6\pi\eta R, \quad (2.3)$$

with η being the dynamic viscosity of the solvent and R being the radius of the particle.

The second force exerted by the solvent on the particles is a stochastic forcing term representing the net effect of the thermal motion of the solvent. It is typically modelled as

$$F_r = \sqrt{2\gamma k_B T} R(t) \quad (2.4)$$

where R is a delta-correlated Gaussian noise function, k_B is the Boltzmann constant, and T is the temperature of the solvent.

The magnitude of two solvent forces in Langevin dynamics are connected via the Einstein fluctuation-dissipation relation. It can be shown that the mean squared displacement of a particle subject only to these two forces will converge to

$$\langle (\Delta x)^2 \rangle \rightarrow 2dDt, \quad (2.5)$$

where d is the dimension of the space in which the particle is moving and the diffusion coefficient D satisfies

$$D = \frac{k_B T}{\gamma}. \quad (2.6)$$

Combining the two solvent forces with Newton's second law, we obtain the free-solution Langevin equation for a particle,

$$ma = -\gamma v + \sqrt{2\gamma k_B T} R(t). \quad (2.7)$$

In biophysical systems, typically the dynamics are highly over-damped; that is, the acceleration is nearly always negligibly small. In this limit, it is common to approximate the acceleration as being zero to obtain the overdamped Langevin equation:

$$v = -\sqrt{2D} R(t). \quad (2.8)$$

The dynamics generated by the overdamped Langevin equation are sometimes called Brownian dynamics, to distinguish them from the original Langevin dynamics.

The standard Langevin dynamics models of molecular motion explored in this thesis neglect hydrodynamic interactions. In this context, hydrodynamic interactions can be understood to introduce correlations into the solvent forces experienced by nearby particles. Various methods exist for incorporating hydrodynamic interactions into coarse-grained simulations, including techniques based on the Oseen tensor, dissipative particle dynamics, and lattice Boltzmann methods (see, e.g., Winter and Geyer [41] for a discussion). Often, however, it is acceptable to omit hydrodynamic effects when modelling macromolecules being driven by electric fields. Serendipitously, the induced flow of ions through the system tends to disrupt the hydrodynamic correlations between particles in such

a way as to render the true dynamics of the system more akin to idealized Langevin dynamics. Similarly, hydrodynamic effects can be suppressed for molecules in confining geometries. Thus, since this thesis is particularly focused on the electrically driven motion of macromolecules through confining geometries, substantial progress can be made without considering these phenomena.

2.3 Electric field models: Laplace's equation

Forces encountered in biophysical systems are predominantly of electrostatic origin. At the small length scales relevant to nanobiophysics, electrostatic fields are actually quite complicated. Each atom in each solvent molecule or macromolecule carries electric charges; some of these molecules are neutral, but many are polarized or even carry net charges. Of particular importance are the dissolved ions present in the solvent. These ions typically move much faster than other electrical charges in the system. Acting as nearly free charges, they tend to form boundary layers surrounding other charged objects. These boundary layers have the effect of screening the electric charges of macromolecules, reducing their apparent charge.

Beyond this, electrohydrodynamic flows also arise in systems when an external electric field is applied. Ions tend to flow in accordance with the direction of the net electric field. However, this bulk flow of ions leads to accumulation of net charge near the charged boundaries of the system. At equilibrium, ions tend to establish recirculant flows along the edges of the system. Paradoxically, these ion flows can induce drag forces on macromolecules near boundaries acting in the direction opposite to the overall electric field.

Miraculously, many factors conspire such that the net effect of electrostatic fields in specific systems is often far less complicated than it appears in general. The boundary layers surrounding macromolecules tend to equilibrate nearly instantly on timescales of interest. Thus, macromolecules can often be modelled as having fixed (screened) charge distributions. In higher fidelity atomistic simulations, this enables domain decomposition methods that only explicitly calculate electrostatic interactions between nearby charged particles. In the lower fidelity coarse-grained models considered in this thesis, it is adequate to model each generic particle in the system as carrying a single effective charge. The other electrohydrodynamic boundary layers, such as a recirculating ion flows, tend to decay on a length scale called the Debye length. This length scale is often very small in experimental conditions relevant to periodic MNFDs, as very large ion concentrations are commonly utilized. Thus, to a good first approximation, these additional electrohydrodynamical effects can often be neglected.

In this thesis, electrostatic fields are modelled by the simplest applicable PDE model: Laplace's equation,

$$\nabla^2 u = 0, \tag{2.9}$$

where u is the electrostatic potential such that $E = -\nabla u$ is the electric field. Once the electric field has been obtained, an electric force is added to the Langevin equations in the form $F_E = qE$, where q is some effective charge for each particle. See Sec. 2.3.1, below, for a discussion of how the effective charge q is modelled.

Despite its simplicity, Laplace's equation is fairly effective at capturing the

most important feature of an electric field: its shape. For instance, the non-uniform shape of the electric field plays an important role in the physics of the nanopore-based MNFD studied in Chp. 3 and the slit-well MNFD studied in Chps. 4 and 6. Laplace’s equation is very commonly used to model the fields in such systems, and is a far more expressive model than assuming constant or piecewise-constant electric force fields.

Nonetheless, one of the opportunities for neural networks in the study of MNFDs is the potential to enable the use of more advanced force field models without incurring prohibitively large computational costs. Such models as the Poisson-Boltzmann or Poisson–Nernst–Planck equations are nonlinear and can be high-dimensional functions of all the particle positions in the system. The neural network method is an appealing option for representing such force fields efficiently. This opportunity is discussed further in App. C and Chp. 4.

2.3.1 Drift velocity, effective charge, and mobility

The overdamped Langevin equation in the presence of a constant and uniform electric field is of the form

$$\gamma v = \sqrt{2\gamma k_B T} R(t) + qE. \quad (2.10)$$

The ensemble average of this equation is simply

$$\gamma \langle v \rangle = qE, \quad (2.11)$$

since the average of R is zero. Thus, a Brownian particle (i.e., a particle moving according to Brownian dynamics) subject to a constant electric force will move, on average, with some constant velocity proportional to the applied field².

In this result, the mean velocity is referred to as the drift velocity v_d . We can write

$$v_d \equiv \langle v \rangle = \frac{q}{\gamma} E \equiv \mu E, \quad (2.12)$$

where the constant of proportionality μ governing the drift velocity established by a given electric force is known as the mobility. More precisely, the mobility described here is the free-solution electrophoretic mobility. Similar mobilities can be defined for other types of driving forces, such as pressure gradients.

Various theories exist for characterizing the electrophoretic mobility μ of macromolecules and other colloids. The most popular theory is that of Von Smoluchowski [40], which predicts that

$$\mu = \frac{\epsilon \zeta}{\eta}, \quad (2.13)$$

where ϵ is the electric permittivity of the solvent, η is the dynamic viscosity of the solvent, and ζ is an important quantity known as the zeta potential. Various corrections and extensions to Smoluchowski's theory exist, but the basic result is sufficient for the systems of concern in this thesis.

The zeta potential ζ of a charged object is defined as the electrostatic potential measured at a certain distance from its surface. Whereas ϵ and η are typically

²It can be shown that the mean velocity of a particle moving according to the full Langevin equation will converge to the same behaviour after an initial transient behaviour.

properties of the solvent alone, the zeta potential ζ can be a complex function of the chemical interactions between the solvent and the surface of the charge object in question. On the other hand, ζ is typically constant if these chemical factors are held fixed, and is therefore independent of the shape of the object. As a result, it is often stated that the Smoluchowski model predicts that the mobility μ is independent of the size and shape of the charged object under consideration.

Mixtures of molecules with different shapes and sizes but identical μ are called free-draining mixtures. Important examples include: mixtures of nanoparticles made of the same material but of varying sizes; and mixtures of DNA molecules of varying length. A common and difficult goal is to separate out the constituent molecules of a free-draining mixture into subpopulations of similar shapes and/or sizes. A broad variety of technologies exist for accomplishing this goal. In particular, periodic MNFDs can be used for this purpose, and this application is one of the main focuses in this thesis.

2.4 Modelling ensembles: The Smoluchowski equation

Equation 2.10 and related Langevin equations are stochastic differential equations (SDEs) describing the random trajectories of particle systems subject to a mix of deterministic and random forces. A roughly equivalent description of these same models can be obtained in terms of the the probability distributions generated over the ensemble of all possible SDE trajectories. In particular, these distributions are governed by deterministic partial differential equations (PDEs).

The conversion of SDEs into PDEs is fraught with subtleties, not the least of which being the nomenclature. For models concerned with the evolution of an ensemble forward in time from some initial condition, the resulting PDE is called the Kolmogorov forward equation of that SDE. Historically, the name of Kolmogorov forward equation emerged in the mathematics community, and was later identified with what was already known as the Fokker-Planck equation in the physics community. However, the modern preference is that the Fokker-Planck equation should refer specifically to the Kolmogorov forward equation of the *velocity* distribution of particles undergoing Langevin dynamics. The PDE of interest in this thesis is actually the Kolmogorov forward equation describing the *position* distribution of particles undergoing Langevin dynamics, which is conventionally called the Smoluchowski equation. This equation, in turn, is formally identical to the convection-diffusion equation, which describes the dynamics of a dissolved substance inside a flowing solvent.

The simple Smoluchowski equation that describes the probability density function of position for a particle moving randomly according to overdamped Langevin dynamics (Eqn. 2.10) is given by

$$\rho_t(x, t) = \nabla \cdot (D(x)\nabla\rho(x, t) - \mu(x)E(x)\rho(x, t)). \quad (2.14)$$

Here D is the diffusion coefficient, μ is the free-solution mobility, and E is the external electric field. Typically this is augmented with boundary conditions and an initial condition for the initial distribution $\rho(x, 0)$.

A useful concept for understanding the Smoluchowski equation is the notion of mean flux of particle positions, also known as the probability flux. For Eqn. 2.14,

the probability flux is

$$j = D\nabla\rho - \mu E\rho. \quad (2.15)$$

In general, the Smoluchowski equation can be rewritten as

$$\rho_t = \nabla \cdot j \quad (2.16)$$

for the appropriate choice of j . If an additional deterministic force field F is added to the model, it will appear as an additional term in Eqn. 2.15 of the form $-F/\gamma$ (where γ is the friction coefficient; Sec. 2.2).

For many-body systems containing n interacting particles, j will be a function of all particles' positions, and inter-particle forces will be present in the probability flux. In a three-dimensional time-dependent system, j and ρ will depend on $3n + 1$ variables. As noted before, high-dimensional PDEs are traditionally very difficult to solve directly. Much of the work in this thesis has been aimed at the long-term goal of applying deep learning to solving the many-body Smoluchowski equation directly. The studies in Chapters 4 and 6 and Apps. B, C, and D have identified and overcome barriers to this goal and will hopefully act as a foundation for future work in this direction.

2.5 Modelling first passage processes: The time-integrated Smoluchowski equation

As the name suggests, a first passage process (FPP) is one that depends on the first time at which a certain event occurs. In this thesis, we are focused specifically on FPPs describing the first time at which a macromolecules makes contact with a certain boundary of the space in which it is moving. Starting from an SDE model, such as Langevin dynamics, the first passage time (FPT) can be described by imposing an appropriate termination condition to the SDE. The FPT distribution can then be sampled empirically using numerical simulations (Sec. A.1), tabulating the times at which simulated trajectories encounter the termination conditions for the first time.

Conversely, modelling the FPT starting from a PDE like the Smoluchowski equation (Sec. 2.14) is very different. In this case, the termination condition of the SDE is replaced by an absorbing boundary condition for the PDE. That is, the probability density function is required to go to zero at the same locations at which the corresponding SDE model would meet its termination condition. In such a model, the probability density function is interpreted as the distribution of the positions of particles that have not yet touched the absorbing boundary at least once.

In this PDE model of a FPP, the particle probability distribution ρ is not properly normalized at $t > 0$ —its integral over the problem domain is strictly less than 1 after the initial time, and decreases monotonically with time. This

motivates the definition of the survival probability,

$$S(t) = \int_{\Omega} \rho(x) dx, \quad (2.17)$$

which is the probability that the initial particle has not yet encountered the absorbing boundary condition by time t . From this quantity it is possible to state the FPT distribution:

$$\tau \sim -S'(t). \quad (2.18)$$

Essentially, $-S'(t)$ is the instantaneous rate of probability flux (Eqn. 2.15) passing into the absorbing boundary at the time t , which is equivalent to the probability of the particle reaching the termination condition for the first time at that time t .

Both the PDE and SDE formulations of the FPT suffer from a common practical limitation: they are defined over an infinite time domain. In practice, particle simulations of the SDE model will terminate when the finite number of simulated particles have all encountered the terminal condition. This can mean that a very large number of particles must typically be simulated to resolve any phenomena that only occur rarely; conversely, it means that the simulation time is a stochastic quantity that is not known *a priori*, and that there is some probability that some particle trajectories will require very long computation times (this is discussed in more detail in Chapter 6). Conversely, to determine FPT distributions using the PDE models, the time-dependent solution $\rho(x, t)$ would be computed until some finite time horizon in order to approximate Eqn. 2.18. Both approaches necessitate costly time-dependent numerical

solutions.

Rather than attempting to deal with the challenges of solving these time-dependent problems over long time horizons, a useful manipulation can be applied to reformulate the problem into a hierarchy of time-independent problems. Consider the Smoluchowski equation for a single particle, given by Eqn. 2.14. Assuming a fixed initial condition and integrating both sides of the equation over the entire time domain yields

$$\int_t \rho_t dt = \nabla \cdot \left(D(x) \nabla \left[\int_t \rho dt \right] - \mu(x) E(x) \left[\int_t \rho dt \right] \right). \quad (2.19)$$

This form assumes the linearity of the various operators and, crucially, the time-invariance of D, μ, E , of the PDE domain Ω , and of the PDE boundary conditions. Next, assume that the problem is reasonably well-behaved such that $\rho \rightarrow 0$ as $t \rightarrow \infty$. Integrating the LHS by parts then yields

$$-\rho_0 = \nabla \cdot \left(D(x) \nabla \left[\int_t \rho dt \right] - \mu(x) E(x) \left[\int_t \rho dt \right] \right), \quad (2.20)$$

where ρ_0 is the initial condition for ρ . This will be referred to as the time-integrated Smoluchowski equation. The time integral of ρ is an important quantity, and accordingly will be given a name:

$$g_0 \equiv \int_t \rho dt. \quad (2.21)$$

Notice that the field $g_0(x)$ does not depend on time; it satisfies the time-independent PDE

$$-\rho_0 = \nabla \cdot (D(x)\nabla g_0 - \mu(x)E(x)g_0). \quad (2.22)$$

Assuming that the domain Ω and the boundary conditions for ρ are also independent of time and linear in ρ , it follows that g_0 satisfies essentially the same boundary conditions as ρ .

The above provides a PDE formulation via which the field g_0 can be solved directly; but what is the value of this quantity? Consider the definition of the FPT distribution in terms of ρ :

$$\tau \sim -S'(t) = -\frac{d}{dt} \int_{\Omega} \rho(x, t) dx = \int_{\Omega} (-\rho_t(x, t)) dx \quad (2.23)$$

where the last equivalence follows as we have assumed that Ω is independent of time. Then the first moment of the FPT, i.e., the mean first passage time (MFPT), can be written as

$$\langle \tau \rangle = \int_T \tau (-S'(t)) d\tau = \int_T \tau \int_{\Omega} (-\rho_t(x, \tau)) dx d\tau = \int_{\Omega} \left[\int_T (-\tau \rho_t) d\tau \right] dx. \quad (2.24)$$

Applying integration by parts to the inner integral of the last expression yields

$$-\int_T \tau \rho_t d\tau = -(\tau \rho)|_{\tau=0}^{\tau=\infty} + \int_T \rho d\tau = g_0. \quad (2.25)$$

Here, it is assumed that $\tau\rho \rightarrow 0$ as $\tau \rightarrow \infty$, which is generally true for time-independent FPPs in biophysics. Thus, it follows that

$$\langle \tau \rangle = \int_{\Omega} g_0 dx. \quad (2.26)$$

In other words, the field g_0 has the property that its volume integral is equal to the MFPT.

The above derivation can be extended to generate all of the higher moments of the FPT distribution. Specifically, a hierarchy of PDEs defines the quantities g_0, g_1, \dots such that the volume integral of g_i yields the i th moment of the FPT distribution and each g_i for $i > 0$ satisfies

$$-g_{i-1} = \nabla \cdot (D(x)\nabla g_i - \mu(x)E(x)g_i). \quad (2.27)$$

This mathematical framework for the moments of the FPT distribution has proven very useful. Redner [36] illustrates its utility for the theoretical analysis of FPPs. This approach is especially fruitful in the case of uniform diffusion, where $E = 0$ and D is constant. In that case, the equations for the g_i reduce to the well-studied Poisson equation, and analogies to electrostatics can be leveraged to draw conclusions about MFPTs in diffusion.

One of the major arguments of this thesis is that g_0 , in particular, has practical value for applied research and development of MNFDs. The work in Chapters 3 and 6 culminate in the conclusion that a key physical property of these devices (the effective mobility) can be expressed exactly as a FPP. The study in App. D illustrates that deep learning can be used to reliably and efficiently solve the

parameterized time-integrated Smoluchowski equation modelling that FPP. As long as the PDE model for g_0 is well-posed, the solution g_0 will depend smoothly on the parameters of the PDE model, such as D , μ , E , and even Ω . Thus, g_0 acts as a proxy for numerically approximating smooth end-to-end mappings from key physical inputs variables to key physical observables. Deep learning plays a key role in efficiently solving and representing these highly parameterized mappings. Although the demonstration in App. D is restricted to the study of single-body molecules (i.e., nanoparticles), extensions of the method to many-body molecules is potentially possible given deep learning's success at solving other high-dimensional PDEs (Sec. A.3). Besides this application, Chapter 3 and App. D suggest that the g_0 fields themselves may contain useful qualitative information about molecular dynamics pertaining to the MFPT. To the best of my knowledge, these efforts are the first demonstrations that the g_0 field is of practical value in applied computational biophysics, rather than just being a mathematical tool for analyses in theoretical biophysics.

Chapter 3

A sequential nanopore-channel device for polymer separation

A sequential nanopore-channel device for polymer separation is a purely computational study in which Magill, Waller, and Haan [28] propose a micro/nanofluidic device (MNFD) design for sorting free-draining polymer mixtures by size, which is named the nanopore-channel device. This kind of molecular sorting is a key technological capability in many biological sciences. Whereas traditional separation methods like gel electrophoresis are still widespread today, MNFD-based sorting technologies could enable the integration of polymer separation pipelines into lab-on-a-chip devices. The device in Magill, Waller, and Haan [28] is an attempt to induce size-dependent effective mobilities by exploiting the size-dependent translocation time of polymers traversing nanopores. In particular, the device geometry consists of a series of nanopores connected by channels, through which polymers are forced via an electric field. The published manuscript is included in Sec. 3.3.

3.1 Motivation

The size-dependence of polymer translocation time has been the subject of extensive study [34, 37]. Many works have considered the idea of estimating polymer lengths by measuring the translocation time. However, the translocation process typically features a large amount of intramolecular variability; that is, a polymer of a given length can generate a very broad distribution of translocation times. This makes it difficult to accurately estimate a polymer's length based on only a single measured translocation time. The more variable the mapping from chain length to translocation time, the greater the number of measurements necessary to make an accurate estimate.

The intramolecular variability of the translocation time τ can be quantified conveniently by the coefficient of variation,

$$c_v(\tau) = \frac{\sigma_\tau}{\langle \tau \rangle}, \quad (3.1)$$

where $\langle \tau \rangle$ and σ_τ are the mean and standard deviation of the translocation time, respectively. When the coefficient of variation is large, the signal-to-noise ratio on measured translocation times is small, and length estimations will be unreliable. Technologies like the filtered nanopore of Briggs et al. [5] can substantially reduce the coefficient of variation of translocation time, thereby improving the estimation of polymer lengths by translocation time.

The problem of highly variable translocation time can be particularly problematic for certain exotic nanopore designs. For instance, the cavity-nanopore proposed in Magill et al. [29] generates non-monotonic

translocation times. It was proposed in that study that such a phenomenon could be tuned to enable polymers of a specific target length to traverse the device more quickly than both shorter and longer chains, with the separation of polymers by size as a potential technological application. However, unpublished follow-up work suggested that, for polymers such as double-stranded DNA, the intramolecular variability of translocation through the cavity-nanopore was very large. Moreover, variance-reduction techniques like the filtered nanopore of Briggs et al. [5] are not applicable to such devices; the noise is intrinsic, coming primarily from the dynamics within the cavity of the device.

The original motivation for the nanopore-channel device being discussed in this chapter was to exploit the central limit theorem to combat this problem. By combining multiple noisy nanopores into a compound system, the idea was to create an overall device that exhibited substantially less variability. For example, consider a set of n identical nanopores, and suppose that a polymer will translocate through all of them consecutively. Then if τ_i is the (stochastic) translocation time of the polymer through the i th nanopore, then its overall time spent translocating is

$$\tau_{\text{total}} = \sum_{i=1}^n \tau_i. \quad (3.2)$$

Suppose further that the polymer is allowed to come to equilibrium between translocation events, so that it is reasonable to treat each translocation time as a statistically independent random variable. Then the central limit theorem states

that, for large n ,

$$\tau_{\text{total}} \sim \mathcal{N}(n\langle\tau\rangle, n\sigma_{\tau}^2), \quad (3.3)$$

where τ is the translocation time through any of the individual nanopores (assumed to be identical), and $\mathcal{N}(a, b)$ denotes the normal distribution with mean a and variance b . The coefficient of variation c_v of the total translocation time is

$$c_v(\tau_{\text{total}}) = \frac{1}{\sqrt{n}} \frac{\sigma_{\tau}}{\langle\tau\rangle} = \frac{1}{\sqrt{n}} c_v(\tau). \quad (3.4)$$

In particular, this quantity approaches zero with increasing n . In other words, even if the individual nanopores generate highly variable translocation dynamics (i.e., $c_v(\tau)$ is large), the overall variability of the system can be made arbitrarily small by stacking a sufficiently large number of nanopores in series. In particular, this approach to variance reduction works even for pores such as the cavity-nanopore, where techniques such as the filtered nanopore will not work.

The nanopore-channel device is a minimalist attempt at a tangible implementation of this mathematical intuition. As described in Fig. 1 of Magill, Waller, and Haan [28] (Sec. 3.3), the device geometry consists of a series of standard nanopores connected by cylindrical channels. Polymers are forced through one nanopore after another by an applied electric field. So long as the channels are sufficiently large, the polymers will return to equilibrium between translocation events.

This device should be capable of enhanced polymer chain length determination. If a single polymer is introduced at one end of the device and driven through

nanopore after nanopore by an electric field, the net ionic current through the system will intermittently be interrupted as the polymer translocated through a nanopore¹. The duration of each translocation can be estimated by the duration of the corresponding current perturbation, and after many translocations the length of the chain will be accurately determined by the mean time per translocation. In practice, the condition of there only being a single polymer in the device at any moment in time is crucial, or signals would become entangled. If polymers are introduced through the first pore in the nanopore-channel device by drawing them electrophoretically from some bulk reservoir, then this condition can be ensured by calibrating the device such that the rate at which polymers are captured from bulk solution is lower than the rate at which polymer traverse the entire nanopore-channel device.

Beyond this, the manuscript (Sec. 3.3) concerns itself with the extended goal of physically separating polymers by length. Separating polymer mixtures is a more broadly applicable goal than determining the lengths of each molecule. In the context of this application, the mathematical intuition presented above overlooks two crucial considerations. Once the intramolecular variability of the overall translocation time in the system is made small, how will this be translated into the distributions of polymer positions? Second, what will happen to polymers in the interval of time between consecutive nanopore translocations? As discussed below, these two questions are closely related.

¹So long as the channels between the nanopores are large compared to the polymer, the polymer should have no detectable effect on the current through the system when it is far from the pores.

3.2 Results

The mathematical analysis in the first part of the manuscript explains how the total translocation time of a polymer traversing the nanopore-channel device can be related to the evolution of polymer position over time. Essentially, the distribution of the time required for the polymer to travel a certain distance can be transformed into the distribution of the distance travelled at a certain moment in time. Importantly, the time to travel a certain distance through the device is the sum of the total translocation time plus the total channel-crossing time (i.e., time taken to cross the channels connecting the consecutive nanopores).

The channel-crossing time emerges from the analysis as a crucial quantity for practical nanopore-channel designs. Paradoxically, the results of Magill, Waller, and Haan [28] (Sec. 3.3) demonstrate that the long-time dynamics of polymers through the nanopore-channel device are sometimes essentially independent of the translocation time. This is the case when the channels are very large, so that polymers spend only a small fraction of their overall transit time inside the nanopores. The nanopores are nonetheless an essential component of the device, because of the bottleneck effect they create at the end of each channel. Indeed, in the absence of these bottlenecks, the polymer dynamics in the channels would be free-draining, and separation would not occur.

Another surprising behaviour of the nanopore-channel device is that it can readily produce separation of polymers into monotonically increasing, monotonically decreasing, or non-monotonic ordering of chain length. This is possible even though the device is constructed using only standard nanopores, for which the translocation time increases monotonically with polymer chain length.


The behaviour arises because the mean channel-crossing time generally decreases monotonically with chain length. The different sorting orders are produced by varying the size and shape of the channels, which controls the relative importance of the two trends.

3.3 Manuscript

A sequential nanopore-channel device for polymer separation

Cite as: J. Chem. Phys. **149**, 174903 (2018); <https://doi.org/10.1063/1.5037449>

Submitted: 24 April 2018 . Accepted: 15 October 2018 . Published Online: 01 November 2018

Martin Magill, Ed Waller, and Hendrick W. de Haan 



View Online



Export Citation



CrossMark

ARTICLES YOU MAY BE INTERESTED IN

[Anomalous packing and dynamics of a polymer chain confined in a static porous environment](#)

The Journal of Chemical Physics **149**, 174902 (2018); <https://doi.org/10.1063/1.5043629>

[Perspective: Dynamics of confined liquids](#)

The Journal of Chemical Physics **149**, 170901 (2018); <https://doi.org/10.1063/1.5057759>

[Single molecule electrophoresis of star polymers through nanopores: Simulations](#)

The Journal of Chemical Physics **149**, 163306 (2018); <https://doi.org/10.1063/1.5029980>

PHYSICS TODAY
WHITEPAPERS

ADVANCED LIGHT CURE ADHESIVES

Take a closer look at what these environmentally friendly adhesive systems can do

READ NOW

PRESENTED BY
 **MASTERBOND**
ADHESIVES | SEALANTS | COATINGS

A sequential nanopore-channel device for polymer separation

Martin Magill,¹ Ed Waller,² and Hendrick W. de Haan^{1,a)}

¹*Faculty of Science, University of Ontario Institute of Technology, 2000 Simcoe St N, Oshawa, Ontario L1H7K4, Canada*

²*Faculty of Energy Systems and Nuclear Science, University of Ontario Institute of Technology, 2000 Simcoe St N, Oshawa, Ontario L1H7K4, Canada*

(Received 24 April 2018; accepted 15 October 2018; published online 1 November 2018)

In this work, we investigated whether a series of nanopores connected by channels can be used to separate polymer mixtures by molecular size. We conducted multiscale coarse-grained simulations of semiflexible polymers driven through such a device. Polymers were modelled as chains of beads near the nanopores and as single particles in the bulk of the channels. Since polymers rarely escape back into the bulk of the channels after coming sufficiently close to the nanopores, the more computationally expensive simulations near the pores were decoupled from those in the bulk. The distribution of polymer positions after many translocations was deduced mathematically from simulations across a single nanopore-channel pair, under the reasonable assumption of identical and independent dynamics in each channel and each nanopore. Our results reveal rich polymer dynamics in the nanopore-channel device and suggest that it can indeed produce polymer separation. As expected, the mean time to translocate across a single nanopore increases with the chain length. Conversely, the mean time to cross the channels from one nanopore to the next decreases with the chain length, as smaller chains explore more of the channel volume between translocations. As such, the time between translocations is a function of the length and width of the channels. Depending on the channel dimensions, polymers are sorted by increasing length, decreasing length, or non-monotonically by length such that polymers of an intermediate size emerge first. *Published by AIP Publishing.* <https://doi.org/10.1063/1.5037449>

I. INTRODUCTION

Nano- and microfluidic devices show great promise as next-generation polymer separation technologies.^{1–3} Potential advantages over traditional separation techniques include faster throughput, higher efficiency, miniaturization and automation (as in lab-on-a-chip designs), and the ability to deal with long polymer chains.

Nanopores, small holes in thin membranes whose diameters and thicknesses are on the order of tens of nanometers, are an important class of nanofluidic devices.² They occur pervasively in biological systems, usually formed by membrane-bound proteins, but can also be fabricated synthetically in, for instance, thin films. They have attracted much attention recently for technological applications, especially DNA sequencing.⁴ However, since the mean passage time of a polymer forced to translocate through a nanopore is a function of its length, nanopores could also be used for polymer separation. Unfortunately, polymer separation using nanopores has proven challenging in practice, as the translocation process is highly variable.^{5–7}

In this work, we study a device consisting of a series of nanopores connected by channels. Our hypothesis is that repeated translocation through multiple nanopores in series should exhibit decreased overall variability relative to translocation through a single nanopore so that this nanopore-channel device could be used for polymer separation. We explore

channel dimensions from hundreds to thousands of nanometers and restrict our attention to cases where polymers can fit in the channels without conformational restrictions. Polymer dynamics in such channels are essentially the same as those in bulk solution.^{2,3} Since these channels have no polymer separation power on their own, we consider this device to be a minimalist implementation of multiple nanopores connected in series. Nonetheless, as we will show, the overall separation power of the nanopore-channel device still depends greatly on polymer dynamics in the channels.

In order to study the polymer separation power of these devices, we present a multiscale model of non-interacting semiflexible homopolymers driven by an applied electric field through a series of nanopores connected by cylindrical channels. We analyze simulations of polymers traversing a single channel and a single nanopore to infer the average speed of polymers moving through the entire nanopore-channel device. Our results indicate that polymers can be sorted with good resolution using hundreds to thousands of pores in series. Channel geometry plays a fundamental role in determining the polymer dynamics. For instance, depending on the channel dimensions, the polymers can be sorted into increasing, decreasing, or non-monotonic order by chain length.

We begin by reviewing some of the relevant literature on polymer separation with micro- and nanofluidic devices. Next, we introduce the details of our multiscale model, which models the system at three levels of detail. We then present our simulation results at each of these three scales. We show that the

^{a)}Electronic mail: Hendrick.deHaan@uoit.ca

nanopore geometry used here is not especially optimized for polymer separation. Such a geometry was chosen by design to demonstrate that the nanopore-channel device provides enhanced polymer separation even without carefully manufactured pores. We also discuss the rich polymer dynamics revealed by simulations in the interior of the channels, far from the pores. We derive a simple physical model that accounts for much of the interplay between the polymers' lengths and the channels' dimensions. We finish with the results demonstrating the separation of polymers by length as they move through many consecutive nanopores.

A. Background and related work

At a high level, the goal of polymer separation can be stated as follows: given a mixture of polymers, group polymers according to some property into distinct spatial regions.⁸ For linear polymers, the goal is usually to separate them according to the chain length. Unfortunately, some polymers of interest (most notably DNA) are free-draining, which means that chains of different lengths move with the same drift velocity under an applied electric field in bulk solution.³ As a result, devices for DNA separation must introduce a length-dependence on polymer motion.

Traditional approaches to polymer separation include gel electrophoresis techniques (the physics of which was reviewed, for instance, by Viovy⁹) and capillary electrophoresis methods (which were recently reviewed, for instance, by Harstad *et al.*¹⁰). Many nano- and microfluidic devices have been considered for next-generation polymer separation technologies. In a review of this topic, Levy and Craighead classified these sorting approaches into entropic sorting devices, Brownian ratchets, structured media, and free solution sorting devices.² In a separate review, Dorfman *et al.* explored two classes of sorting devices: post-arrays (which are a subset of what Levy *et al.* called structured media devices) and slit-well devices (which fall under what Levy *et al.* called entropic sorting devices).³

In entropic sorting devices, length-dependent mobility is created by a series of entropic traps. For instance, this has been accomplished by Han and Craighead with the slit-well motif, which consists of a series of small nanoslits connected by larger slits, or wells.^{11,12} There is an entropic barrier for polymers to enter the small nanoslits from the wells, and the dynamics of this escape process depend on the chain length. Specifically, as they can enter the smaller nanoslit by any point along their length, longer chains make the transition more quickly than smaller chains. This device is similar in spirit to the nanopore-channel device considered here, with the nanopores playing the role of the small nanoslits and the channels playing the role of the wells. However, the devices are fundamentally different because the slit-well device is essentially two-dimensional, whereas nanopores are essentially one-dimensional. For instance, in the nanopore-channel device, chains of all length can only enter the nanopore at a single position, so the primary dynamics of the slit-well motif have no analogues. Another intrinsic difference between the two devices is in the shape of the electric field. The electric field in the well of the slit-well devices is reduced from that in the slits by a linear ratio of their respective length scales

(by conservation of flux). In contrast, the electric field in the channels of the nanopore-channel device is reduced from that in the nanopores by the square of the ratios of the two length scales. Thus the nanopore-channel device intrinsically supports much larger field gradients. Nonetheless, the systems do share some similarities: as we will show, in both systems, polymers can exhibit the unusual behaviour of increasing mobility with increasing chain length.³

As mentioned earlier, in addition to their use for sequencing applications, nanopores have also been previously considered for polymer separation. Length-dependent motion arises because longer polymers take longer, on average, to translocate through a given nanopore. For instance, Carson *et al.* and Bell *et al.* demonstrated experimentally that the length of a translocating DNA chain can be identified by its translocation time.^{5,6} Unfortunately, the use of nanopores for polymer separation by length is limited by the intrinsic variability of the translocation process. For instance, typical nanopores cannot identify the length of double-stranded DNA (dsDNA) molecules with better resolution than roughly 1000 bp.⁶ By carefully manufacturing nanopores with diameters very close to 3 nm, Carson *et al.* achieved improved resolutions on the order of 100 bp.⁵ Briggs *et al.* also demonstrated resolution on the order of 100 bp in nanofiltered nanopore devices, where a nanoporous membrane pre-confines DNA molecules before translocation.⁷ These innovations improve the sensitivity of the translocation time to the chain length by reducing the variability of the translocation process, but this comes at the cost of additional manufacturing requirements.

In principle, a simple way of improving the resolution of any nanopore-based filters would be to use many of them in series. Repeated independent applications of a stochastic filter n times should theoretically reduce the net variability of the process by a factor of $1/\sqrt{n}$. One might therefore hope to simply study a single nanopore in isolation and then extrapolate to determine the filtering potential of n nanopores in series. However, as this work will demonstrate, the precise fashion in which the nanopores are connected is fundamental to the overall polymer dynamics and cannot be neglected.

To our knowledge, there have not been many previous studies exploring the translocation of polymers through multiple nanopores in series. Langecker *et al.* conducted experiments with dsDNA in a micron-scale cavity bounded by two nanopores.¹³ They used 10 kbp dsDNA and conducted time-of-flight measurements at a variety of voltages. In contrast to the work presented here, those experiments varied voltage but did not vary the chain length. Thus, those experiments cannot be used directly to speculate about the separation power of nanopores in series. Instead, they demonstrate that nanopores in series can yield a more detailed analysis of molecular properties than single nanopores.

II. MODEL AND METHODOLOGY

In this section, we will describe our multiscale model of polymers in the nanopore-channel device as well as our simulation methodology. The system was modelled at three scales, which will be referred to as the microscopic, mesoscopic, and macroscopic scales in order from the finest to

coarsest level of detail. Figure 1 summarizes the models used at each scale. At the largest scale [Fig. 1(c)], the system geometry consists of nanopores connected in series by channels, with all pores and channels centered on a common axis. Subsections II A–II C will describe the model at each scale in turn, and additional details are available in the [supplementary material](#).

A. Microscopic model

The microscopic model [Fig. 1(a)] was used to capture the dynamics of the molecules near the nanopores. This phase included capture from free solution into the pore and translocation through the pore. This is the most detailed scale in our model, as the polymer dynamics in this region are both complicated and crucial to a proper understanding of the device. The simulations utilized a standard coarse-grained Langevin dynamics (CGLD) polymer consisting of N beads.¹⁴ This level of detail has been used extensively to study polymer translocation through nanopores.^{15–26} Simulations were conducted using the ESPResSo software package on the Shared Hierarchical Academic Research Computing Network (SHARCNET).²⁷

Polymers were constructed using N identical monomers arranged linearly using finitely extensible nonlinear elastic (FENE) forces to bond monomers and Weeks-Chandler-Anderson (WCA) forces to model the excluded volume.¹⁴ The semiflexibility of the chain was modelled using a harmonic

potential on the angle formed by any three consecutive monomers along the chain backbone. A persistence length of $L_p = 10\sigma$ was imposed, where σ is the effective monomer diameter dictated by the WCA interaction. Length scales throughout the remainder of this paper will be expressed in terms of this effective chain width σ . This choice was motivated by double-stranded DNA molecules, which have an effective width of a few nanometers (larger than the steric width ≈ 2.5 nm) and a persistence length of roughly 30–50 nm.^{28–31} As discussed in the [supplementary material](#), the aspect ratio of the model polymer was somewhat smaller than that of real DNA, which enabled longer polymers to be simulated.

WCA forces were also used to apply purely repulsive interactions between monomers and the nanopore walls. An effective nanopore radius of 0.8 was used. This radius was selected as it only enabled polymers to traverse the pore by an end, i.e., polymers could not enter the pore in a folded configuration. This simplified the current analysis, but future work will explore the impact of folding dynamics.

The thermal motion of the polymer was modelled via Langevin dynamics. The thermostat used a thermal energy of $kT = 1$ and a friction coefficient of $\gamma = 1$ for each monomer, and the monomer mass was $m = 1$ as well. Translocation was driven by an electric force field. The shape of the electric field in the microscopic region was approximated by the analytic solution for the electric field of a nanopore in an

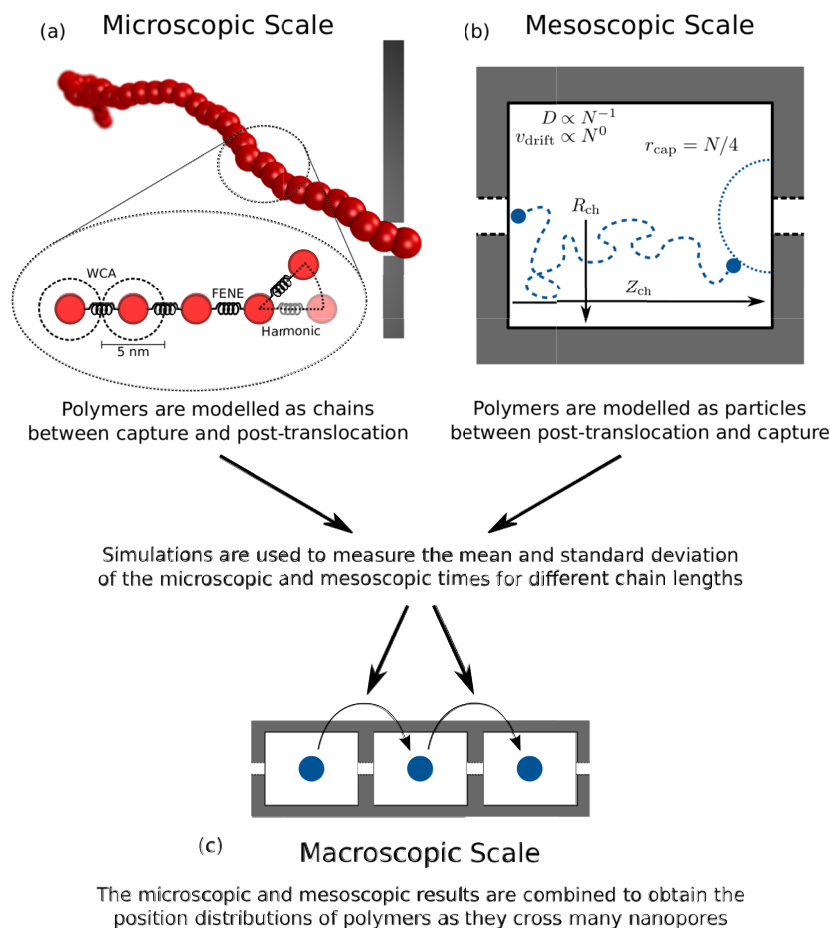


FIG. 1. Schematics illustrating the three modelling scales used for this system. (a) In the microscopic model, WCA and FENE interactions were used to construct a linear homopolymer of N beads, and harmonic angular potentials were used to implement semiflexibility. (b) At the mesoscopic scale, the entire polymer was represented by a single Brownian particle. Simulations were initialized with particles at the inlet pore of a channel of radius R_{ch} and length Z_{ch} and terminated when particles were absorbed near the outgoing pore. (c) At the macroscopic scale, the net motion of polymers through the nanopore-channel device was inferred by treating each of the consecutive nanopore-channel pairs as identical and independent subunits of the device.

infinite unbounded domain.³² This is a good approximation when the channel dimensions are much larger than the pore radius, which is the case for all simulations in this paper. The advantage of this approximation is that the results of the microscopic simulations become independent of the channel geometry, enormously improving the computational efficiency of the model. The magnitude of the electric field was scaled to match the Péclet number of the simulations to relevant experimental conditions for dsDNA translocation, as discussed in the [supplementary material](#).³³

At the start of the microscopic simulations, polymers were initialized in an equilibrated conformation a distance $r_{\text{cap}} = N/4$ from the nanopore. This distance was chosen to balance two effects. We found that this r_{cap} was far enough from the pore that the electric field strength was not strong enough to significantly deform the polymer conformations from equilibrium as they diffused around that distance. Conversely, this r_{cap} was also close enough to the pore that the electric field was strong enough that polymers were unlikely to diffuse very far from the pore before translocating. See the [supplementary material](#) for more details.

The primary measurement in the microscopic model was the microscopic time, defined as the time after the release of a polymer (i.e., after equilibration) until all of its monomers were located on the *trans* side of the nanopore. If polymers ever moved far enough away from the pore that the closest monomer to the pore was farther than the cut-off radius of N from the pore, then the event was considered a failure. Failed events were restarted so that the total number of successful translocations measured at each chain length was 2000.

B. Mesoscopic model

The mesoscopic model [Fig. 1(b)] was used to study the dynamics farther from the nanopores, in the bulk of the channels. In this region, the electric field gradient was small over the length scale of the polymer so that it could not significantly deform the polymer conformations. In other words, these dynamics were dominated by the translational motion of the center of mass. As such, in the mesoscopic model, each polymer was represented by a single effective particle and was simulated using coarse-grained Brownian dynamics (CGBD).¹⁴

The mesoscopic CGBD simulations were made compatible with the microscopic CGLD simulation conditions. The diffusion coefficients of the mesoscopic particles were set to $1/N$, corresponding to the center of mass diffusion coefficients predicted by the Rouse model for polymers of length N in the microscopic model. The CGBD thermostat and force magnitudes were set as in the microscopic model, with each polymer experiencing a net friction coefficient equal to N times the monomer γ , in accordance with the Rouse model. Similarly, each CGBD chain experienced a net force equal to N times the force that a single monomer would feel. As a result, large chains diffused more slowly than smaller chains, but all chains exhibited identical free solution electrophoretic drift velocities.

The electric field in the mesoscopic model was obtained by solving Laplace's equation in cylindrical coordinates using the finite element method with the FEniCS software package.³⁴

The channel was modelled as a cylinder of length Z_{ch} and radius R_{ch} [see Fig. 1(b)]. The mesoscopic simulations were initialized with particles in the inlet nanopore of a channel and proceeded until the particles contacted the hemisphere of radius $r_{\text{cap}} = N/4$ surrounding the outgoing nanopore. We call the first passage time for particles to reach this absorbing hemispherical boundary the mesoscopic time.

As stated above, the starting radius of the microscopic model is included in the mesoscopic model as an absorbing boundary. As argued in the [supplementary material](#), this is valid because it was possible to choose a set of capture radii $r_{\text{cap}} = N/4$ such that the failure rate was simultaneously small for all chain lengths. However, the failure rate is counted by the number of events that diffuse to a distance of $4r_{\text{cap}}$ from the pore. Within that distance, the microscopic model neglects the channel walls, which is a limitation of the current model. Nevertheless, the computational benefit of neglecting the channel walls in the microscopic model is enormous, as it enables the same microscopic results to be used across many channel geometries. To justify the use of this simplifying assumption, the mesoscopic model was only studied for polymer chains that satisfied the condition that

$$\min(R_{\text{ch}}, Z_{\text{ch}}) > r_{\text{cap}} + R_G, \quad (1)$$

where R_G is the radius of gyration and was approximated with the wormlike chain model

$$R_G \approx \left[\frac{L_p N}{3} - L_p^2 + \frac{2L_p^3}{N} \left(1 - \frac{L_p}{N} \left(1 - e^{-\frac{N}{L_p}} \right) \right) \right]^{1/2}, \quad (2)$$

where N is the nominal contour length and $L_p = 10$ is the persistence length.³⁵ This restriction reduces the influence of omitting the walls from the microscopic model, since it ensures that the walls are far from the pore, where the electric field is weak. Furthermore, it also ensures that the polymers have R_G much smaller than the dimensions of the channel, which is another assumption of the model.

C. Macroscopic model

The transport of polymers across multiple channels was captured in the macroscopic model. The behaviour of polymers at this scale was solved analytically on the assumption that their dynamics in distinct nanopores and channels were independent and identical. When this is the case, the macroscopic dynamics are entirely determined by the micro- and mesoscopic dynamics in any given channel.

Since the boundary between the microscopic and mesoscopic domains was chosen such that the rate of transport from the microscopic zone back into the mesoscopic region was negligible, the time to cross a given nanochannel, t_{macro} , can be approximately modelled as

$$t_{\text{macro}} = t_{\text{micro}} + t_{\text{meso}}, \quad (3)$$

where t_{micro} and t_{meso} , the times to cross the respective sub-domains, are statistically independent. Thus the probability density function of t_{macro} is the convolution of the other two variables', i.e.,

$$\rho(t_{\text{macro}}) = \rho(t_{\text{micro}}) * \rho(t_{\text{meso}}). \quad (4)$$

Similarly, the probability density function of the time at which the polymer will enter the k th channel for the first time, $t(k)$, is given by

$$\rho(t(k)) = \underbrace{\rho(t_{\text{macro}}) * \rho(t_{\text{macro}}) * \dots * \rho(t_{\text{macro}})}_{k \text{ times}}. \quad (5)$$

Equation (5) for the distribution of $t(k)$ can be computed directly from the distributions of t_{micro} and t_{meso} . However, by the central limit theorem, $\rho(t(k))$ will converge in the limit of large k to

$$\rho(t(k)) \approx \mathcal{N}(k\mu_{\text{macro}}, k\sigma_{\text{macro}}^2), \quad (6)$$

where μ_{macro} and σ_{macro}^2 are the mean and variance, respectively, of t_{macro} , and $\mathcal{N}(\mu, \sigma^2)$ denotes the normal distribution. This approach treats k as a continuous random variable when it is in fact discrete; this is justified for large values of k .

From this, we derive the distribution of polymer positions as a function of channel number k . The probability of a polymer being in channel $k' \geq k$ at time t is given by

$$p(k' \geq k) = p(t(k_0) \leq t) = \int_0^t p(t(k) = t') dt' \quad (7)$$

$$\approx \frac{1}{2} \left(1 + \operatorname{erf} \left(\frac{t - k\mu_{\text{macro}}}{\sqrt{2k\sigma_{\text{macro}}^2}} \right) \right), \quad (8)$$

when k is large enough to apply the central limit theorem to $\rho(t(k))$. This is the cumulative distribution of polymer positions in the macroscopic system. To obtain the corresponding probability density function, we take the derivative and observe that the position distribution over k at time t is

$$\rho(k; t) \approx \frac{-\partial p(k' \geq k)}{\partial k} \quad (9)$$

$$= \frac{t + k\mu_{\text{macro}}}{\sqrt{8\pi k^3 \sigma_{\text{macro}}^2}} \exp \left(-\frac{(t - k\mu_{\text{macro}})^2}{2k\sigma_{\text{macro}}^2} \right). \quad (10)$$

Finally, we can apply the central limit theorem again to find that the position distribution when t is large is given by

$$\lim_{t \rightarrow \infty} \rho(k; t) = \mathcal{N} \left(\frac{\sigma^2 + 2\mu t}{2\mu^2}, \frac{5\sigma^4}{4\mu^4} + \frac{\sigma^2 t}{\mu^3} \right) \quad (11)$$

$$= \mathcal{N} \left(\frac{t}{\mu} + \frac{1}{2} \left(\frac{\sigma}{\mu} \right)^2, \left(\frac{\sigma}{\mu} \right)^2 \frac{t}{\mu} + \frac{5}{4} \left(\frac{\sigma}{\mu} \right)^4 \right), \quad (12)$$

where μ and σ are the mean and variance of t_{macro} , but the subscripts have been omitted for ease of reading. The mean and variance of $\rho(k; t)$ were calculated using Mathematica 11.1.³⁶

Equation (12) can be used to compute the position distribution's coefficient of variation CV , which is its standard deviation divided by its mean. This describes the relative width of the distribution. In the limit of large t , the coefficient of variation will approach

$$\lim_{t \rightarrow \infty} CV = \frac{\sigma}{\sqrt{\mu}} \frac{1}{\sqrt{t}} \propto \frac{1}{\sqrt{t}}. \quad (13)$$

Thus, although the position distributions become arbitrarily broad at large times, they become progressively narrower relative to the mean displacement.

Furthermore, when t is large, the mean polymer position given by Eq. (12) will be roughly t/μ_{macro} , and so the mean polymer speeds through the nanopore-channel device will approach $1/\mu_{\text{macro}}$. Since $\mu_{\text{macro}} = \mu_{\text{micro}} + \mu_{\text{meso}}$ by linearity, the mean position of polymers after many translocations depends equally on the microscopic and mesoscopic dynamics. The same conclusion applies to the variance of the position distribution. In other words, one cannot consider the filtering effect of a series of nanopores without also considering the exact process by which polymers are fed from one nanopore into the next. As will be shown below, in Sec. III B, the mesoscopic dynamics are quite rich and can in fact be more important than the microscopic dynamics.

III. SIMULATION RESULTS

In this section, we present the results of our simulations at the microscopic, mesoscopic, and macroscopic scales as follows:

A. Results at the microscopic scale

The microscopic simulations were conducted for chains of length $N = 10, 20, 50, 75, 100, 150$, and 200 . Figures 2(a) and 2(b) show the mean and variance, respectively, of the microscopic time. Recall that the microscopic time includes *both* the translocation time, which is commonly studied, and the capture time, which is the time for the polymer to enter the nanopore after the start of simulations. The capture process has often been neglected in the literature, although recent work has shown that it can radically alter the subsequent translocation process.²⁶

Figures 2(a) and 2(b) demonstrate that the mean and variance of the microscopic time increase monotonically with the chain length over the range studied here. Although our results do not suggest that the mean and variance of microscopic time are related to the chain length N by power laws, it is still interesting to consider the approximate scaling of these quantities with N . Linear regression between the logarithms of the respective quantities yields

$$\mu_{\text{micro}} \approx 0.98N^{2.09}, \quad (14)$$

$$\sigma_{\text{micro}}^2 \approx 2.87N^{4.01}. \quad (15)$$

Simulation studies measuring the scaling of just the mean translocation time with N under conditions similar to those used here have generally reported exponents in the range of 1.2–1.6.^{16–26} Conversely, the mean microscopic time measured here scales with N to an exponent of roughly 2.09. The fact that the microscopic time contains the capture time in addition to the translocation time is likely a major factor in this discrepancy. In fact, the capture radius $r_{\text{cap}} = N/4$ was increased in proportion to the chain length, further complicating a direct comparison between the microscopic time and translocation time. Actually, given these significant differences between the two quantities, it is remarkable that their scaling with N is as similar as it is.

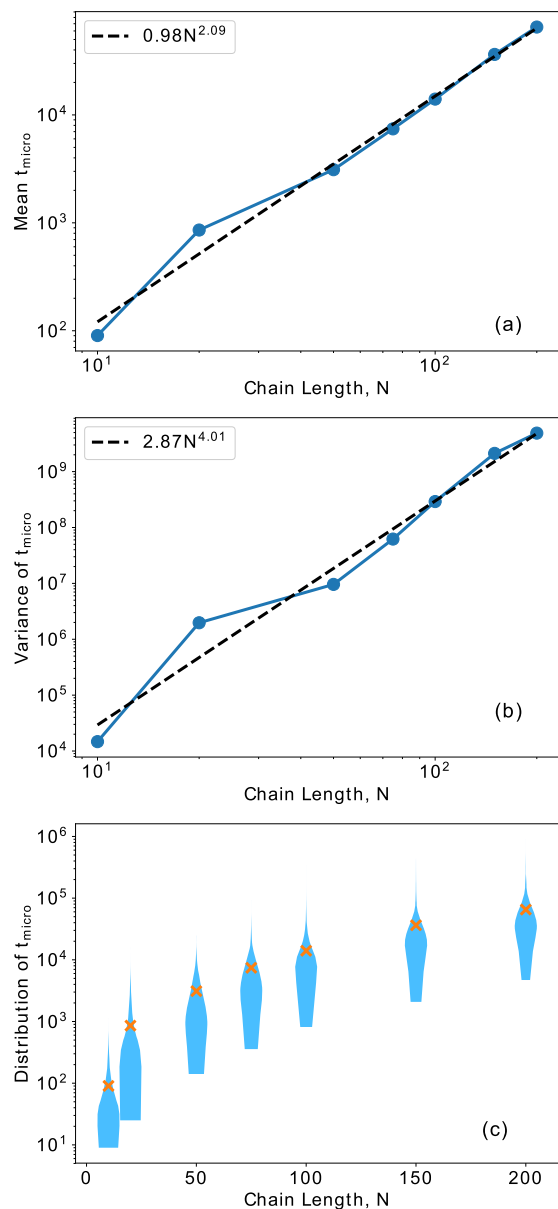


FIG. 2. Results of the microscopic simulations. (a) The mean of the microscopic time, t_{micro} , as a function of chain length. Error bars of one standard error are much smaller than the marker size. (b) The variance of t_{micro} as a function of chain length. (c) Violin plots showing the distributions of t_{micro} as a function of chain length. The markers indicate the mean of each distribution.

It is also interesting to consider the intrinsic polymer separation power of the nanopores studied here. Figure 2(c) contains violin plots of the microscopic times, from which microscopic time distributions can be compared directly between different values of N . It is clear from Fig. 2(c) that the microscopic time distributions are heavy-tailed at every chain length and that there is a significant amount of overlap between distributions for different chain lengths. Thus the current nanopore setup does not appear to be optimized for separation applications. We chose such a nanopore because the goal of the present work is to demonstrate that even nanopores that clearly could not be used to separate polymers in a single pass can successfully separate polymers when connected in series. Of course, nanopore-channel devices constructed using nanopores with superior length resolution, such as those demonstrated by

Carson *et al.* or Briggs *et al.*, would be expected to achieve even better polymer separation.^{5,7}

B. Results at the mesoscopic scale

Figure 3(a) shows the mean mesoscopic time for the same range of chain lengths N as used in the microscopic model, as well as for $N = 300$, all for various combinations of the channel dimensions (R_{ch} , Z_{ch}). Also shown is the mean microscopic time, for comparison. Mesoscopic simulations were run for every combination of both R_{ch} and Z_{ch} in $\{30, 45, 60, 75, 90, 150, 300\}$; and for every combination of Z_{ch} in $\{500,$

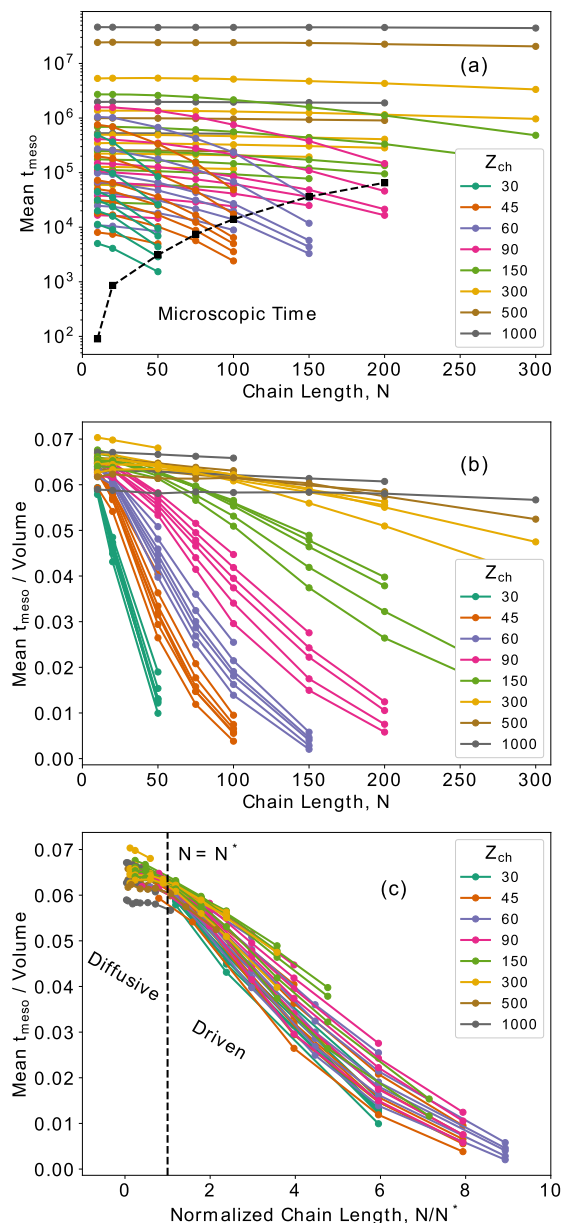


FIG. 3. In all three plots, the line color indicates the channel length Z_{ch} . Different lines of the same color correspond to different channel radii R_{ch} such that the mean of the mesoscopic time, t_{meso} , always increases monotonically with R_{ch} . (a) Mean t_{meso} for a range of chain lengths in channels of various dimensions (R_{ch} , Z_{ch}). The dashed line shows the microscopic time. (b) Mean t_{meso} normalized by mesoscopic volume for a range of chain lengths in channels of various dimensions. (c) Mean t_{meso} normalized by mesoscopic volume shown as a function of the normalized chain length N/N^* (discussed in the text) in channels of various dimensions.

1000} and R_{ch} in {50, 100, 500}. The channel lengths, Z_{ch} , are shown by the line color; different lines of the same color correspond to different channel radii, R_{ch} , such that the mean t_{meso} increases monotonically with R_{ch} in all cases. Only choices of $(N, R_{\text{ch}}, Z_{\text{ch}})$ that satisfy the restriction of Eq. (1) were studied.

Some dependence of the mean t_{meso} on the chain length is apparent: in all cases, longer chains cross the mesoscopic region somewhat faster than shorter chains. However, the extent of this effect depends greatly on the channel dimensions. In particular, mesoscopic time is only comparable to microscopic time when the channel volume is small. In large channels, the mesoscopic time is much larger than the microscopic time; furthermore, it changes very little with the chain length. This suggests that connecting nanopores in series with large gaps of bulk solution between them will not lead to any polymer separation because the thermal motion between subsequent pores will overwhelm the length sensitivity introduced by the pores.

Some of the mesoscopic dynamics can be understood by considering particle trajectories. Figure 4 illustrates two typical mesoscopic simulations. The heatmaps show numerical measurements of the time-integrated position probability density throughout the channel (in cylindrical coordinates). These distributions, which will be referred to as g_0 , show the average residence time of polymers in each region of the channel before absorption near the exit nanopore. The integral of g_0 over the channel volume equals the mean mesoscopic time.³⁷ Figure 4 highlights the most important influence of chain length N on dynamics in the mesoscopic simulations. The plot for $N = 10$ shows a g_0 distribution that is nearly uniform over the entire channel, illustrating that these short chains typically diffuse throughout the entire channel before being captured. Conversely, the longer chains with $N = 50$ typically drift axially into the capture radius without diffusing very far in the radial direction. As a result, the average residence time g_0 decays rapidly towards zero away from the channel axis.

Nearly uniform g_0 distributions, like the one in Fig. 4(a), are typical when the channel volume is large or the polymer chain length is small. We will refer to such conditions as the diffusive regime. Since t_{meso} is the volume integral of g_0 , it follows that normalizing Fig. 3(a) by channel volume

might account for some of the dependence of t_{meso} on channel geometry. Figure 3(b) shows this by plotting the following ratio:

$$\frac{t_{\text{meso}}}{V_{\text{meso}}} = \frac{t_{\text{meso}}}{\pi R_{\text{ch}}^2 Z_{\text{ch}} - \frac{1}{2} \frac{4}{3} \pi r_{\text{cap}}^3}. \quad (16)$$

As expected, $t_{\text{meso}}/V_{\text{meso}}$ is nearly independent of N , R_{ch} , and Z_{ch} in the diffusive regime.

The remaining mesoscopic dynamics arise for chains that are long enough to drift axially into the capture radius without first diffusing throughout the channel volume [as in Fig. 4(b)]. We will call such conditions the driven regime. We can estimate the chain length at which the dynamics transition between the diffusive and driven regimes by comparing the characteristic time scales of axial drift and radial diffusion.

The characteristic time scale on which particles drift axially across the channel is

$$\tau_{\text{drift},z} \sim \frac{Z_{\text{ch}} - r_{\text{cap}}}{v_{\text{drift},z}}, \quad (17)$$

where $v_{\text{drift},z}$, the characteristic axial drift velocity, is roughly

$$v_{\text{drift},z} \sim (NF_z^c)(\gamma/N) = F_z^c \gamma, \quad (18)$$

where F_z^c is the characteristic axial force in the bulk of the channel. The characteristic force in the bulk of the channel can be expressed as

$$F_z^c \approx F_p^* \left(\frac{r_p}{R_{\text{ch}}} \right)^2, \quad (19)$$

where $F_p^* \approx 5.19$ is the average axial force in the pore (after tuning the Péclet number; see the [supplementary material](#)) and r_p is the radius of the pore. Altogether, then the characteristic time scale for drifting across the channel is

$$\tau_{\text{drift},z} \sim \frac{Z_{\text{ch}} - \frac{N}{4}}{\gamma F_p^*} \left(\frac{R_{\text{ch}}}{r_p} \right)^2, \quad (20)$$

using $r_{\text{cap}} = N/4$.

The characteristic time scale on which particles diffuse radially is

$$\tau_{\text{diff},r} \sim \frac{R_{\text{ch}}^2}{D} = \frac{R_{\text{ch}}^2}{\frac{k_B T}{\gamma N}} = \frac{N \gamma R_{\text{ch}}^2}{k_B T}. \quad (21)$$

Setting this equal to the characteristic drift time yields

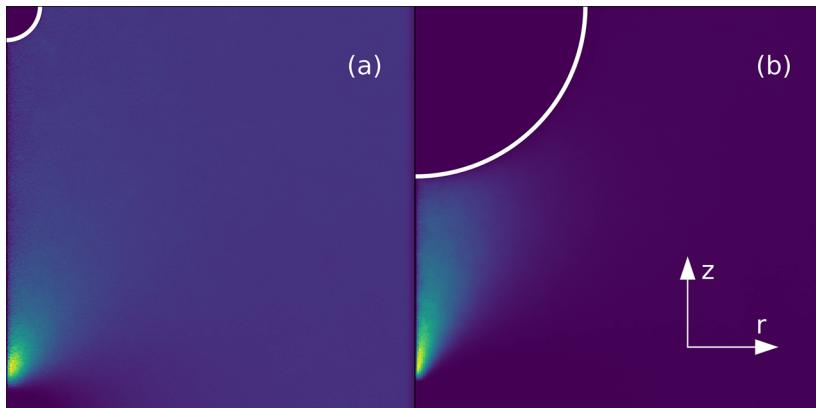


FIG. 4. Heatmaps of g_0 (see the text) comparing two mesoscopic simulations in the same nanochannel geometry of $(R_{\text{ch}}, Z_{\text{ch}}) = (30, 30)$ (a) for $N = 10$ and (b) for $N = 50$. Brightness indicates the average residence time of polymers in each region of the channel before reaching the absorbing boundaries (shown in white). Longer chains spend less time far from the axis of the channel.

$$N^* = Z_{\text{ch}} \left/ \left(\frac{1}{4} + \frac{\gamma^2 F_p^* r_p^2}{k_B T} \right) \right. \approx \frac{Z_{\text{ch}}}{3.57}. \quad (22)$$

It is interesting to note that this estimate of N^* does not depend on R_{ch} . This fortuitous result arises because the magnitude of the axial electric field in the bulk of the channel decreases with R_{ch} with the same scaling as the radial diffusion time.

In the same way that normalizing time by V_{meso} accounted for most of the mesoscopic dynamics in the diffusive regime, the dynamics in the driven regime can be accounted for by normalizing the chain lengths by N^* . As shown in Fig. 3(c), plotting $t_{\text{meso}}/V_{\text{meso}}$ against N/N^* produces nearly the same curve for all channel geometries currently under consideration. In other words, the mean mesoscopic time is, to a reasonably good approximation, a function of the channel volume and N^* .

Overall, Fig. 3 clearly demonstrates that t_{meso} generally decreases monotonically with N . This is in direct contrast to t_{micro} , which increases monotonically with N . In Sec. III C, we will show that the interplay between these trends can be exploited to sort polymers into increasing, decreasing, or non-monotonic functions of N , depending on the relative magnitudes of t_{micro} and t_{meso} .

The standard deviation of the mesoscopic time, σ_{meso} , also plays an important role in ultimately understanding the macroscopic dynamics of the device. Unfortunately, it was not possible to find a simple characterization of σ_{meso} comparable to that obtained above for μ_{meso} . To some extent, larger channels produce larger standard deviations. The details are more complicated than this and are shown in the [supplementary material](#). Detailed characterization is left to future work, as the purpose of the present work is primarily to show that the nanopore-channel device can indeed sort polymers under reasonable experimental conditions.

C. Results at the macroscopic scale

Finally, we will combine the results of Secs. III A and III B to compute the dynamics of polymers at the macroscopic scale, after they have crossed many consecutive pores in the nanopore-channel device. In particular, this section will show the results with three choices of the channel dimensions, which cause polymers to become sorted into increasing, decreasing, and non-monotonic orders by length, respectively. Furthermore, we will discuss some of the general qualitative trends that are suggested by our analysis.

1. Sorting into monotonically increasing order of length

Recall that the microscopic time always increases with chain length N , whereas the mesoscopic time always decreases with N . To produce sorting in an increasing order of length, then, the geometry must be chosen so that $t_{\text{micro}} \gg t_{\text{meso}}$. As summarized in Fig. 3, the mean t_{meso} can be made smaller at all chain lengths by reducing the channel volume. Conversely, decreasing the channel length Z_{ch} accentuates the decrease of the mean t_{meso} with N , which is counterproductive in the pursuit of increasing sorting. Thus to produce increasing sorting,

one must choose a channel with a small volume but a large Z_{ch} ; therefore, R_{ch} must be small.

Figures 5(a) and 5(d) show the results at the macroscopic scale for a device with $(R_{\text{ch}}, Z_{\text{ch}}) = (30, 90)$. Figure 5(a) shows the time evolution of the approximated means and standard deviations of the polymer position distributions over the channel number k . Specifically, these are obtained by combining the results of the microscopic and mesoscopic simulations at each value of N with Eq. (12), which is valid for large k . The inset of Fig. 5(a) shows $1/\mu_{\text{macro}}$ as a function of N , which we previously argued is the average polymer speed in the long-time limit. Conversely, Fig. 5(d) shows the detailed position distributions for each chain length, computed using Eq. (10), after 40×10^6 units of simulation time. These results demonstrate increasing sorting by chain length N over this range of chain lengths.

Figure 5(a) shows that a good separation only occurs after a very large number of channels are traversed. In practice, it is likely desirable from a manufacturing point of view to minimize the number of requisite channels. For increasing sorting, this can be accomplished by increasing the magnitude of t_{micro} . Future work will explore options for accomplishing this, such as by using a nanopore with an internal cavity. As shown in previous work, when the cavity size is slightly smaller than R_G , it acts as an entropic trap, greatly increasing the translocation time.³⁸

2. Sorting into monotonically decreasing order of length

Next, we will demonstrate decreasing sorting by length. In this case, in contrast to Sec. III C 1, the geometry must be chosen so that $t_{\text{micro}} \ll t_{\text{meso}}$. This is accomplished by making the channel volume large. However, as Z_{ch} increases, the dependence of t_{meso} on N becomes less pronounced, which reduces the separation power of the device. Thus decreasing sorting occurs when the volume is large and Z_{ch} is small, in direct contrast to increasing sorting.

Figures 5(c) and 5(f) demonstrate decreasing sorting in a channel with $(R_{\text{ch}}, Z_{\text{ch}}) = (90, 45)$. A good separation is achieved with far fewer channels than for increasing sorting. This can be understood as follows. Sorting into a decreasing order of length relies on the dependence of t_{meso} on N . As shown in Fig. 3(c), increasing R_{ch} increases the mean mesoscopic time without significantly changing the dependence of t_{meso} on N . As a result, the difference in t_{meso} between short and long chains can be made large by increasing R_{ch} , increasing the sorting power per channel.

On the other hand, increasing R_{ch} too much compromises the filtering effect, as it broadens the position distributions. This broadening arises because in very wide, short channels, most of the channel volume is far from the axis, where the electric field is weak. Polymers that diffuse away from the axis before crossing the length of the channel axially remain trapped in the channel for a long time. Conversely, since the channel is also short, polymers will occasionally drift axially straight from pore to pore without first diffusing far from the channel axis. As a result, the spread in mesoscopic times is very large, which leads to a broadening of the macroscopic position distributions.

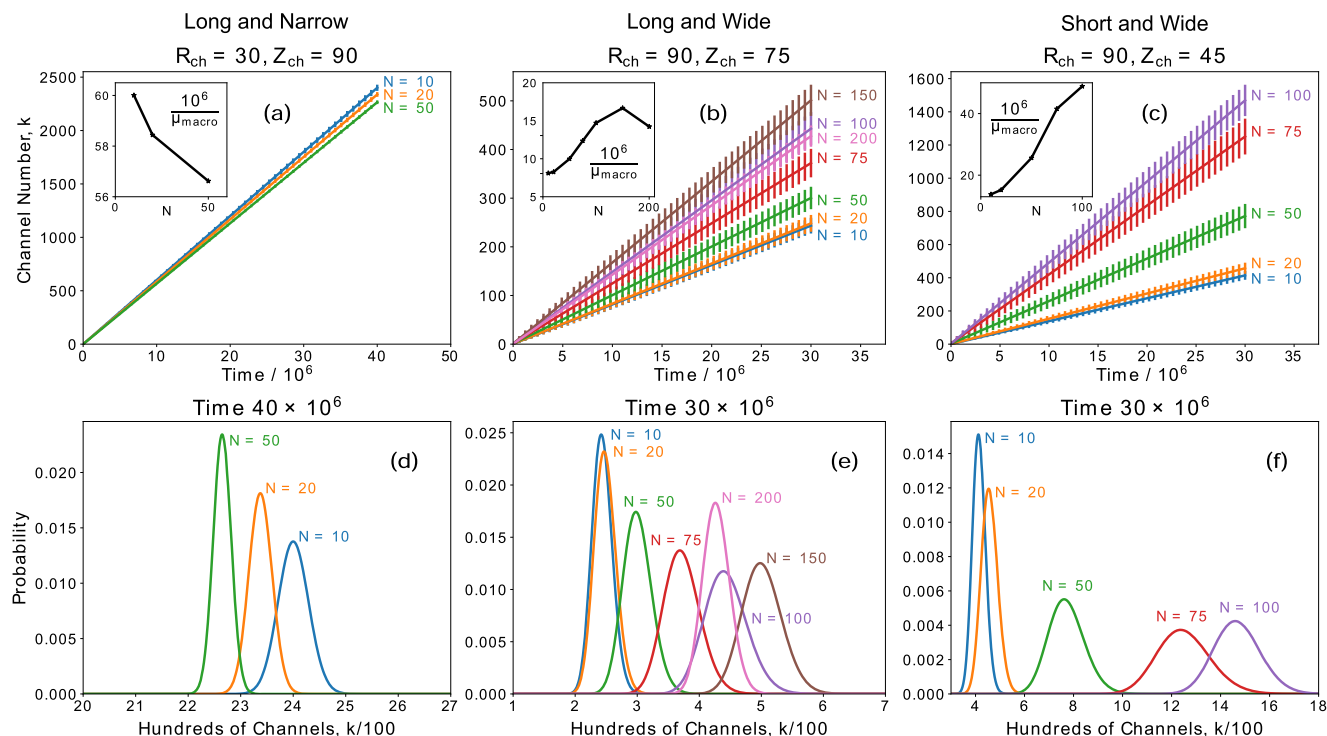


FIG. 5. Results of the simulations at the macroscopic scale. The lines are labeled and colored by chain length N . (a)–(c) show the means and standard deviations of polymer positions over channel number k as a function of time, whereas (d)–(f) show the complete polymer position distributions over k at fixed moments in time. The insets in (a)–(c) show the (rescaled) average polymer speeds at large time, $10^6/\mu_{\text{macro}}$, against the chain length N . Note that μ_{macro} denotes the mean macroscopic time. (a) and (d) demonstrate sorting into an increasing order of chain length in a long and narrow geometry, with $(R_{\text{ch}}, Z_{\text{ch}}) = (30, 90)$; (b) and (e) demonstrate sorting into a non-monotonic order of length in a long and wide geometry, with $(R_{\text{ch}}, Z_{\text{ch}}) = (90, 75)$; and (c) and (f) demonstrate sorting into a decreasing order of chain length in a short and wide geometry, with $(R_{\text{ch}}, Z_{\text{ch}}) = (90, 45)$.

3. Sorting into non-monotonic order of length

The strictly increasing or decreasing sorting cases are extreme presentations of the nanopore-channel device. In general, the device will sort polymers into a non-monotonic order by length because typically $t_{\text{micro}} \ll t_{\text{meso}}$ for short chains, whereas $t_{\text{meso}} \ll t_{\text{micro}}$ for sufficiently long chains. Following the reasoning of Secs. III C 1 and III C 2, then, short chains will be sorted into a decreasing order of length, and long chains will be sorted into an increasing order of length. Specifically, chains of some intermediate length will traverse the nanopore-channel device more quickly than both shorter and longer chains.

Figures 5(b) and 5(e) demonstrate this non-monotonic sorting in a channel with $(R_{\text{ch}}, Z_{\text{ch}}) = (90, 75)$. In this configuration, the chains with $N = 150$ move faster than all the other chain lengths. This type of behaviour is ideal for applications where a specific population of chains must be isolated from both longer and shorter chains. Conversely, the population of chains with $N = 200$ was not separated from the chains with $N \approx 100$. This is inevitable in a situation where speed is a non-monotonic function of N . Nonetheless, this example demonstrates that a good separation can still be achieved among the chains that are smaller than the fastest chain length (e.g., $N = 50$ from $N = 75$, in this case) and also among chains that are longer than the fastest chain length (e.g., $N = 150$ from $N = 200$, in this case). In fact, among these two populations, a good separation is achieved in this example using fewer channels than required in either of the previous two

examples. In this sense, the device producing non-monotonic sorting demonstrated better separation power per nanopore than those producing monotonic sorting.

IV. CONCLUSIONS

We have demonstrated that polymers can be sorted by length using a series of nanopores connected by channels. Good length separation was observed despite the relatively poor length sensitivity of the specific nanopore geometry studied here.

Our results clearly indicate that the polymer dynamics in the channels cannot be neglected, even though the channels have no intrinsic separation power in the absence of the nanopores. In fact, whereas ignoring the channels (for instance, by assuming t_{macro} consists of only translocation time) would lead one to expect sorting into increasing order of chain length (since translocation time increases with the chain length), the nanopore-channel device can produce increasing, decreasing, or non-monotonic sorting orders, depending on the channel dimensions. Furthermore, we showed that the separation power per nanopore can actually be greater in devices that sort into non-monotonic and decreasing orders.

Finally, it is interesting to contrast the dynamics of the nanopore-channel devices studied here and the slit-well devices studied extensively in the literature.³ As pointed out above, the devices differ at a fundamental level because the slit-well device has one completely unconfined dimension,

whereas the nanopore-channel device has none. Nonetheless, we recover the counterintuitive result that longer polymers can traverse the nanopore-channel device more quickly than smaller polymers.

SUPPLEMENTARY MATERIAL

See [supplementary material](#) for additional details concerning the model and simulation implementations.

ACKNOWLEDGMENTS

H.W.d.H. gratefully acknowledges funding from the Natural Sciences and Engineering Research Council (NSERC) in the form of Discovery Grant No. 2014-06091. M.M. gratefully acknowledges funding from the Ontario Graduate Scholarship (OGS).

- ¹R. Mulero, A. S. Prabhu, K. J. Freedman, and M. J. Kim, *J. Assoc. Lab. Autom.* **15**, 243 (2010).
- ²S. L. Levy and H. G. Craighead, *Chem. Soc. Rev.* **39**, 1133 (2010).
- ³K. D. Dorfman, *Rev. Mod. Phys.* **82**, 2903 (2010).
- ⁴D. Branton, D. W. Deamer, A. Marziali, H. Bayley, S. A. Benner, T. Butler, M. Di Ventra, S. Garaj, A. Hibbs, X. Huang, S. B. Jovanovich, P. S. Krstic, S. Lindsay, X. S. Ling, C. H. Mastrangelo, A. Meller, J. S. Oliver, Y. V. Pershin, J. M. Ramsey, R. Riehn, G. V. Soni, V. Tabard-Cossa, M. Wanunu, M. Wiggin, and J. A. Schloss, *Nat. Biotechnol.* **26**, 1146 (2008).
- ⁵S. Carson, J. Wilson, A. Aksimentiev, and M. Wanunu, *Biophys. J.* **107**, 2381 (2014).
- ⁶N. A. Bell, M. Muthukumar, and U. F. Keyser, *Phys. Rev. E* **93**, 022401 (2016).
- ⁷K. Briggs, G. Madejski, M. Magill, K. Kastitis, H. W. de Haan, J. L. McGrath, and V. Tabard-Cossa, *Nano Lett.* **18**, 660 (2018).
- ⁸J. C. Giddings, *Unified Separation Science* (Wiley, 1991).
- ⁹J.-L. Viovy, *Rev. Mod. Phys.* **72**, 813 (2000).
- ¹⁰R. K. Harstad, A. C. Johnson, M. M. Weisenberger, and M. T. Bowser, *Anal. Chem.* **88**, 299 (2015).
- ¹¹J. Han and H. Craighead, *J. Vac. Sci. Technol., A* **17**, 2142 (1999).
- ¹²J. Han and H. G. Craighead, *Science* **288**, 1026 (2000).
- ¹³M. Langecker, D. Pedone, F. C. Simmel, and U. Rant, *Nano Lett.* **11**, 5002 (2011).
- ¹⁴G. W. Slater, C. Holm, M. V. Chubynsky, H. W. de Haan, A. Dubé, K. Grass, O. A. Hickey, C. Kingsbury, D. Sean, T. N. Shendruk, and L. Zhan, *Electrophoresis* **30**, 792 (2009).
- ¹⁵C. Forrey and M. Muthukumar, *J. Chem. Phys.* **127**, 015102 (2007).
- ¹⁶T. Ikonen, A. Bhattacharya, T. Ala-Nissila, and W. Sung, *Phys. Rev. E* **85**, 051803 (2012).
- ¹⁷J. L. A. Dubbeldam, V. G. Rostiashvili, A. Milchev, and T. A. Vilgis, *Phys. Rev. E* **85**, 041801 (2012).
- ¹⁸V. V. Lehtola, R. P. Linna, and K. Kaski, *Phys. Rev. E* **78**, 061803 (2008).
- ¹⁹V. V. Lehtola, R. P. Linna, and K. Kaski, *Europhys. Lett.* **85**, 58006 (2009).
- ²⁰V. V. Lehtola, K. Kaski, and R. P. Linna, *Phys. Rev. E* **82**, 031908 (2010).
- ²¹V. V. Lehtola, R. P. Linna, and K. Kaski, *Phys. Rev. E* **81**, 031803 (2010).
- ²²R. P. Linna and K. Kaski, *Phys. Rev. E* **85**, 041910 (2012).
- ²³K. Luo, T. Ala-Nissila, S.-C. Ying, and R. Metzler, *Europhys. Lett.* **88**, 68006 (2009).
- ²⁴J. L. A. Dubbeldam, V. G. Rostiashvili, A. Milchev, and T. A. Vilgis, *Phys. Rev. E* **87**, 032147 (2013).
- ²⁵A. Bhattacharya, W. Morrison, K. Luo, T. Ala-Nissila, S.-C. Ying, A. Milchev, and K. Binder, *Eur. Phys. J. E* **29**, 423 (2009).
- ²⁶S. C. Vollmer and H. W. de Haan, *J. Chem. Phys.* **145**, 154902 (2016).
- ²⁷H.-J. Limbach, A. Arnold, B. A. Mann, and C. Holm, *Comput. Phys. Commun.* **174**, 704 (2006).
- ²⁸D. Stigter, *Biopolymers* **16**, 1435 (1977).
- ²⁹E. Sobel and J. Harpst, *Biopolymers* **31**, 1559 (1991).
- ³⁰A. Savelyev, *Phys. Chem. Chem. Phys.* **14**, 2250 (2012).
- ³¹A. R. Klotz, L. Duong, M. Mamaev, H. W. de Haan, J. Z. Chen, and W. W. Reisner, *Macromolecules* **48**, 5028 (2015).
- ³²F. Farahpour, A. Maleknejad, F. Varnik, and M. R. Ejtehadi, *Soft Matter* **9**, 2750 (2013).
- ³³H. W. de Haan, D. Sean, and G. W. Slater, *Phys. Rev. E* **91**, 022601 (2015).
- ³⁴M. Alnæs, J. Blechta, J. Hake, A. Johansson, B. Kehlet, A. Logg, C. Richardson, J. Ring, M. E. Rognes, and G. N. Wells, *Arch. Numer. Software* **3**, 9 (2015).
- ³⁵I. Teraoka, *Polymer Solutions* (Wiley Online Library, 2002).
- ³⁶Wolfram Research, Inc., *Mathematica*, Version 11.1, Champaign, IL, 2017.
- ³⁷S. Redner, *A Guide to First-Passage Processes* (Cambridge University Press, 2001).
- ³⁸M. Magill, C. Falconer, E. Waller, and H. W. de Haan, *Phys. Rev. Lett.* **117**, 247802 (2016).

Supplementary Material:

A Sequential Nanopore-Channel Device for Polymer Separation

Martin Magill,¹ Ed Waller,² and Hendrick W. de Haan^{1, a)}

¹⁾ *University of Ontario Institute of Technology
Faculty of Science*

²⁾ *University of Ontario Institute of Technology
Faculty of Energy Systems and Nuclear Science*

(Dated: 13 October 2018)

I. MICROSCOPIC MODEL

This section describes the model used for the microscopic simulations.

A. Microscopic Polymer Model

The polymer in the microscopic model was composed of N identical spherical particles. The system was subjected to a Langevin thermostat with thermal energy $k_B T = 1$ and a friction coefficient $\gamma = 1$. The particles were bonded together into a linear chain using the finitely extensible nonlinear elastic (FENE) potential,

$$U_{\text{FENE}}(r) = -\frac{1}{2} k_{\text{FENE}} r_{\text{max}}^2 \ln \left(1 - \left(\frac{r}{r_{\text{max}}} \right)^2 \right), \quad (1)$$

where r is the center-to-center distance between particles, k_{FENE} is the FENE spring constant, and r_{max} is the maximum extension of the FENE bonds¹. Excluded volume interactions were modelled between all pairs of particles by the Weeks-Chandler-Anderson (WCA) potential,

$$U_{\text{WCA}}(r) = \begin{cases} 4\epsilon \left[\left(\frac{\sigma}{r} \right)^{12} - \left(\frac{\sigma}{r} \right)^6 \right] + \epsilon & r \leq 2^{\frac{1}{6}} \sigma \\ 0 & r > 2^{\frac{1}{6}} \sigma \end{cases}, \quad (2)$$

where r is again the center-to-center separation¹. The ϵ and σ parameters describe the energy and length scales, respectively, of the excluded volume interactions.

For a polymer created with FENE and WCA potentials in CGLD, Kremer and Grest demonstrated that the choice of parameters

$$k_{\text{FENE}} = 30 \frac{\epsilon}{\sigma^2}, \quad r_{\text{max}} = 1.5\sigma, \quad (3)$$

was numerically stable and reliably eliminated bond crossing in the polymer². This work adopted these values. Additionally, the simulations used $\sigma = 1$, so that all other simulation lengths were expressed in units of σ . Finally, ϵ was set to $k_B T = 1$, the thermal energy

scale set in the Langevin thermostat. As a result, the time-averaged bond length of the combined FENE and WCA interaction was $\langle r \rangle_t \approx 0.97\sigma \approx \sigma$, so that σ was also representative of the effective monomer size.

To model semi-flexibility, an angular potential was imposed on the angle formed by every three consecutive particles along the polymer backbone, given by

$$U_{\text{angular}}(\theta) = \frac{1}{2} k_{\text{angular}} (\theta - \pi)^2, \quad (4)$$

where θ is the angle formed by the three consecutive particles and k_{angular} is the stiffness of the potential¹. Under the simulation conditions used here, the polymer was found to satisfy

$$L_P \approx k_{\text{angular}}, \quad (5)$$

where L_P is the persistence length of the chain. Thus k_{angular} was set equal to the desired value of $L_P = 10\sigma$ (see the main text).

B. Microscopic Nanopore Model

In each iteration of the simulations, the polymer was initialized on the *cis* side of a nanopore. The nanopore was constructed using the pore constraint object provided by the ESPResSo software package. The pore constraint consists of a mathematical plane, of a nominal thickness t_{nom} , through which a perpendicular hole of nominal radius r_{nom} is removed. The $U_{\text{WCA}}(r)$ potential that was used to model excluded volume interactions between particles was also defined between particles and the pore constraint plane. In this interaction, the distance r is measured from the center of the particle to the nearest point on the plane. Given this excluded volume, it is convenient to define the effective width $t_{\text{eff}} = t_{\text{nom}} + \sigma$ and effective radius $r_{\text{eff}} = r_{\text{nom}} - \frac{\sigma}{2}$ of the nanopore. In the same sense that the particles behave approximately as hard spheres of radius $\sigma/2$, the nanopore corresponds to a hard wall of its effective dimensions.

For the present study, the nanopore was defined with $t_{\text{nom}} = 10^{-3}\sigma$ and $r_{\text{nom}} = 1.3\sigma$, giving effective dimensions of $t_{\text{eff}} \approx 1.0\sigma$ and $r_{\text{eff}} = 0.8\sigma$. The thickness was chosen as such to model the thin-membrane limit. The nanopore radius was selected to be small enough that translocation through the nanopore could only occur in unfolded configurations.

^{a)} Electronic mail: Hendrick.deHaan@uoit.ca

C. Microscopic Electric Field Model

Polymer translocation was driven by an applied voltage drop across the system. The electric field was based on the analytic form used by Farahpour et al.³. This field, $\vec{E}_{\text{Obl}}(\mu, \nu, \phi)$, is obtained as the gradient of the electric potential that solves Laplace's equation in oblate spheroid coordinates (μ, ν, ϕ) around a hyperboloid-shaped nanopore, which is of the form

$$V_{\text{Obl}}(\mu, \nu, \phi) = \frac{\hat{\mu}V_0}{\pi a \cosh(\mu) \sqrt{\sinh^2(\mu) + \sin^2(\nu)}}. \quad (6)$$

Here, V_0 is the total voltage drop across the system, applied infinitely far away on either side of the membrane. The parameter a corresponds to the radius of the pore, which was set to $a = r_{\text{eff}}$.

The mapping from oblate spheroid to cartesian coordinates depends on a parameter ν_0 , which is the hyperboloid surface of constant ν corresponding to the insulating boundary condition of the membrane wall. The ESPResSo pore constraints, however, have flat planar walls. Similarly, the solution $\vec{E}_{\text{Obl}}(\mu, \nu, \phi)$ cannot capture the shape of the field inside the pore of the pore constraint.

For simplicity, the nanopore was approximated as a cylinder of radius r_{eff} . As described by Farahpour et al., the electric field for cylindrical pores through a flat membrane can be captured using $\vec{E}_{\text{Obl}}(\mu, \nu, \phi)$ in the limit $\nu_0 \rightarrow 0$ outside of the pore, and combining this with a uniform, purely axial field $\vec{E}_{\text{pore}} = E_{\text{pore}}\hat{z}$ inside the pore³. Given a total voltage drop V_{total} across the system, the voltage drop across the pore is $V_{\text{pore}} = E_{\text{pore}}t_{\text{eff}}$, and the remaining voltage drop is assigned to V_0 for the field outside the pore. The two voltages are obtained by enforcing that \vec{E}_{pore} be equal to \vec{E}_{Obl} at the center of the pore.

In this approach, the nanopore is approximated as a cylinder, which is a reasonable approximation to the toroidal boundary of the ESPResSo pore constraint. It is important to note that the intersection of a plane with a cylinder forms a corner, which manifests in $\vec{E}_{\text{Obl}}(\mu, \nu, \phi)$ as an unusually strong electric field near that region. Such strong fields could create numerical errors in simulation. However, the electric field was applied to the center of each particle, and the center of the particles never entered the anomalous region due to the excluded volume behaviour.

D. Tuning to Experimental Conditions

Since the present study was motivated by the application of sorting dsDNA molecules using nanopores and channels, the simulation parameters were chosen in a manner consistent with experimentally relevant conditions. This was accomplished by matching the drift-diffusion balance in simulation to those commonly found

in related experiments, as proposed by de Haan et al.⁴. Specifically, the simulations were modelled after feasible experimental conditions, utilizing a high concentration of NaCl electrolyte and a voltage drop of 200 mV per nanopore⁵. In this section, subscripts will be used to indicate when each parameter is being expressed in simulation or experimental units. These are also summarized in Table I.

The steric width of dsDNA is roughly 2.4 nm, but in electrolytic solution it behaves with an effective width due to the cloud of counter-ions that moves with it⁶⁻⁹. The effective width and persistence length depend strongly on electrolytic conditions. For the present study, the polymer width $\sigma_{\text{sim}} = 1.0$ in simulation was modelled as corresponding to an effective width of $\sigma_{\text{exp}} = 5$ nm. This is close to, but somewhat larger than relevant experimental values. However, larger values of σ_{exp} enable larger dsDNA molecules to be simulated, as computational demands grow rapidly with N , the number of beads per polymer.

The choice of $\sigma_{\text{exp}} = 5$ nm fixes all other length scales in the system. The pore dimensions of $t_{\text{eff}} = 1.0\sigma$ and $r_{\text{eff}} = 0.8\sigma$ therefore corresponds to a membrane that is 5 nm thick containing a pore with a radius of 4 nm. Conversely, since the experimental persistence length of dsDNA under relevant conditions is roughly $(L_P)_{\text{exp}} = 50$ nm⁶⁻⁹, the persistence length in simulation must be set to $(L_P)_{\text{sim}} = 10.0\sigma$.

The voltage drop across the system was also tuned to experiment. Nanopore translocation experiments commonly use driving voltages on the order of 200 mV⁵. The corresponding simulation voltage is chosen so as to reproduce the experimental drift-diffusion balance. This balance is represented by a Péclet number, defined as

$$\text{Pé} = \frac{v^*L^*}{D^*}, \quad (7)$$

where v^* is a characteristic velocity, L^* is a characteristic length, and D^* is a characteristic diffusion coefficient⁴.

The characteristic velocity was set to the characteristic drift velocity due to the driving voltage,

$$v^* = \mu \frac{V_{\text{total}}}{L^*}, \quad (8)$$

where μ is the polymer mobility, V_{total} is the voltage drop across the system, and L^* is the same characteristic length as above. In simulations, $\mu_{\text{sim}} = 1/\gamma = 1.0$. In experiments, the mobility of dsDNA in free solution in NaCl was taken as $\mu_{\text{exp}} = 3.14 \times 10^{-8} \text{ m}^2/\text{V.s}^{10}$.

The characteristic diffusion coefficient was taken to be that of the polymer. In experiments, the diffusion coefficient of dsDNA in NaCl is described by

$$D_{\text{exp}} = \frac{2.38 \times 10^{-12} \text{ m}^2}{\left(\frac{L_C}{\text{m}} \times 10^6\right)^{0.608} \text{ V} \cdot \text{s}}, \quad (9)$$

where L_C is the contour length of the chain in m, which will be discussed below¹¹. In CGLD simulations, the

Experiment	Simulation
$\sigma_{\text{exp}} = 5 \text{ nm}$	$\sigma_{\text{sim}} = 1.0$
$(L_P)_{\text{exp}} = 50 \text{ nm}$	$(L_P)_{\text{sim}} = 10.0\sigma$
$D_{\text{exp}} = \frac{2.38 \times 10^{-12}}{(L_C/\mu\text{m})^{0.608}} \frac{\text{m}^2}{\text{V}\cdot\text{s}}$	$D_{\text{sim}} = \frac{k_B T}{\gamma N}$
$V_{\text{exp}} = 200 \text{ mV}$	$V_{\text{sim}} = \frac{\mu_{\text{exp}}}{\mu_{\text{sim}}} \frac{D_{\text{sim}}}{D_{\text{exp}}} V_{\text{exp}}$

TABLE I. Physical quantities in simulation and experimental models.

polymer will exhibit Rouse dynamics, so its diffusion coefficient will be

$$D_{\text{sim}} = \frac{k_B T}{\gamma N}, \quad (10)$$

where N is the number of particles used to represent the chain¹². Since σ is approximately the effective diameter of each particle, $N \approx L_C/\sigma$.

Finally, the characteristic length L^* is chosen to be the chain width, σ . Using these values, the experimental and simulation Péclet numbers can be set equal and solved for V_{sim} , as follows:

$$\text{Pé}_{\text{exp}} = \text{Pé}_{\text{sim}}, \quad (11)$$

$$\frac{v_{\text{exp}}^* L_{\text{exp}}^*}{D_{\text{exp}}^*} = \frac{v_{\text{sim}}^* L_{\text{sim}}^*}{D_{\text{sim}}^*}, \quad (12)$$

$$\mu_{\text{exp}} \frac{V_{\text{exp}}}{\sigma_{\text{exp}}} \frac{\sigma_{\text{exp}}}{D_{\text{exp}}^*} = \mu_{\text{sim}} \frac{V_{\text{sim}}}{\sigma_{\text{sim}}} \frac{\sigma_{\text{sim}}}{D_{\text{sim}}^*}, \quad (13)$$

$$V_{\text{sim}} = V_{\text{exp}} \frac{\mu_{\text{exp}}}{\mu_{\text{sim}}} \frac{D_{\text{sim}}^*}{D_{\text{exp}}^*}. \quad (14)$$

Equation 14 gives an equation for the choice of V_{sim} that will match the drift-diffusion balance of a corresponding experimental configuration. However, Equation 14 for V_{sim} depends on the chain length. We evaluated V_{sim} for a representative tuning chain length, and the resulting voltage was used for all simulated chain length. The tuning chain was chosen to correspond to $N_0 = 100$ beads, corresponding to a dsDNA fragment with $L_C = 500 \text{ nm}$.

E. Microscopic Simulation Procedure

The approach of the polymer to the nanopore from free solution, referred to as the capture process, was explicitly simulated. This is in contrast with the standard translocation simulation protocol, where the polymer is typically initialized with one or more monomers already threaded in the pore. Recent work has emphasized the importance of properly simulating the capture process¹³. The procedure used here closely follows that presented by Vollmer et al.¹³. All particle evolution in the microscopic simulations was conducted by ESPResSo using Velocity-Verlet integration with $\Delta t = 0.01\tau$, where τ refers to the simulation time units. The side of the nanopore membrane

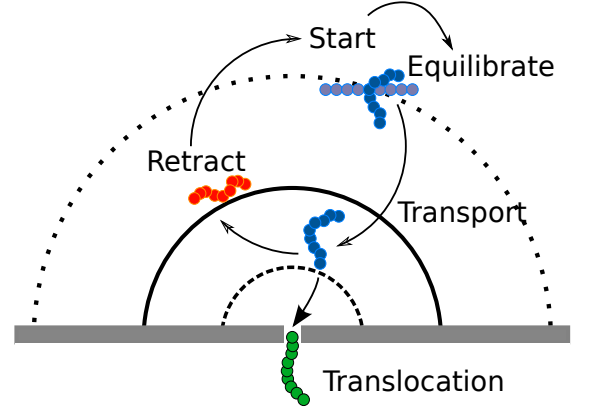


FIG. 1. Illustration of the simulation procedure used for the microscopic model.

on which the polymer was initialized will be referred to as the *cis* region, and the other side as the *trans* region.

The simulation procedure used for the microscopic simulations is illustrated in Figure 1. Initially, the first particle in the polymer was randomly placed on a hemisphere of radius $r_{\text{eq}} = 3N\sigma$ centered at the pore entrance, i.e. the middle of the *cis* face of the nanopore in line with the effective thickness of the membrane. The hemisphere was sampled uniformly for this point, but points located within $\sigma/2$ of the nanopore membrane were rejected, as the particle-wall interaction energy exceeded $k_B T$ at that distance. The remainder of the polymer's N particles were then added parallel to the plane of the nanopore membrane. With the middle monomer held fixed and with γ reset temporarily to 0.1, the chain was allowed to equilibrate for a time $\tau_{\text{eq}} = 200N\tau$.

After equilibration, the polymer was translated closer to the pore as follows. The particle closest to the entrance of the nanopore was translated in a straight line towards the pore entrance until it was a distance $r_{\text{cap}} = (N/4)\sigma$ away. The rest of the polymer was translated by the same vector. As before, if any monomers were within a distance $\sigma/2$ of the nanopore membrane after this translation, the initialization was rejected and the process was restarted.

From this configuration, with γ reset to 1.0 and all particles freed to move, the primary simulation was conducted. Polymer evolution continued until one of two termination conditions were encountered. If the polymer moved entirely to the *trans* region, then the event was terminated and considered a successful translocation. However, if the polymer ever moved far enough from the pore that the minimum distance between the polymer and the pore entrance was farther than a cut-off distance $r_{\text{cut}} = N\sigma$, the event was terminated and recorded as a retracted event.

After retractions, the polymer was re-initialized until a successful translocation was obtained. For each successful translocation, two parameters were recorded: τ_{micro} , the duration of the successful translocation, as well as the

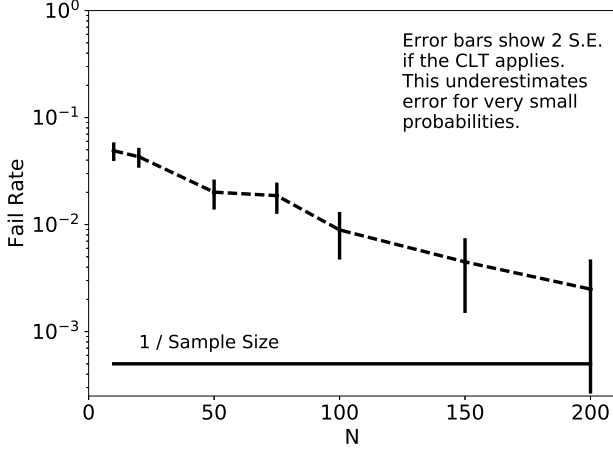


FIG. 2. The fail rates shown as a function of chain length.

number of retractions that occurred before the successful translocation.

F. Microscopic Failure Rates

Figure 2 shows the probability of failure by retraction as a function of N . The failure rate was quite small for all chain lengths, but decreased exponentially with chain length. This implies that scaling the starting radius linearly with N is not enough to render the capture process independent of N . Of course, given the complex dynamics involved in the capture process, this is not surprising. Nonetheless, for the current parameters, the failure rate is small for all chain lengths, so that the bias introduced by treating the capture radius as a purely absorbing boundary in the mesoscopic model is acceptably small.

II. MESOSCOPIC MODEL

This section describes the coarse-grained Brownian dynamics model used to study the mesoscopic simulations. The physical parameters were chosen in consistency with those used for the microscopic model, and thus matched the same experimental conditions. The length and time scales in the mesoscopic model were the same as in the microscopic model, namely σ and τ , and the electric field strength was set equal to that derived for the microscopic model.

A. Mesoscopic Polymer Model

In the mesoscopic model, each polymer was represented by a single particle. The particle position

evolved according to Brownian dynamics with a time-independent external force field,

$$\vec{v}(t) = \frac{1}{\gamma} \vec{F}_{\text{ext}}(\vec{x}(t)) + \sqrt{2D} \vec{R}(t), \quad (15)$$

where \vec{x}, \vec{v} are the position and velocity of the particle, γ is its friction coefficient, D is its diffusion coefficient, \vec{F}_{ext} is the total external force applied to the polymer, and \vec{R} is a unit random process satisfying

$$\langle R_i(t) \rangle = 0 \quad (16)$$

$$\langle R_i(t_1), R_j(t_2) \rangle = \delta_{ij} \delta(t_1 - t_2). \quad (17)$$

The second condition means that the random force acts independently in all three spatial dimensions and acts independently at each instant in time.

In the microscopic simulations, the friction coefficient of each particle was $\gamma = 1$. Since the microscopic dynamics were governed by Langevin dynamics, the Rouse model of polymer dynamics can be applied¹². According to this model, the net friction coefficient for the polymer's center of mass is $N\gamma$. Thus D in the mesoscopic simulations was set to $k_B T / N\gamma$, where $k_B T = 1$ as per the microscopic simulations, and N represents the chain's length. Note that, in the mesoscopic model, N is explicitly assigned to the polymer, as polymers are represented by only a single particle regardless of their length.

The net force on the polymer \vec{F}_{ext} can be written as

$$\vec{F}_{\text{ext}} = N \vec{F}_0, \quad (18)$$

where \vec{F}_0 corresponds to the electric field experienced by each of the polymer's N particles. The final equation of motion was thus

$$\vec{v}(t) = \vec{F}_0(\vec{x}(t)) + \sqrt{\frac{2}{N}} \vec{R}(t). \quad (19)$$

In practice, the BD equations were integrated numerically using a simple first-order finite difference scheme:

$$\Delta \vec{x}(t) = \vec{F}_0(\vec{x}(t)) \Delta t + \sqrt{\frac{2}{N}} \Delta t \vec{R}(t), \quad (20)$$

where a factor of $1/\sqrt{\Delta t}$ arises from the discretization of the stochastic term¹. Again, $\Delta t = 0.01$ was used. The solution was implemented in Python using the Numba module's CUDA API¹⁴. At each timestep, each component of $\vec{R}(t)$ was generated using

$$R_i(t_0) = \sqrt{12}(U - 0.5), \quad (21)$$

where U is a random variable uniformly distributed on $(0, 1)$. This U was sampled using the Numba/CUDA-compatible xorshiftstar function provided by Siu Kwan Lam on the NVIDIA Developer Blog¹⁵.

B. Mesoscopic Channel Model

The channel geometry used for this study consisted of a cylindrical channel of radius R_{ch} and length Z_{ch} with nanopores in the center of both circular faces. The nanopore dimensions matched the effective dimensions of the pore used in the microscopic simulations, namely a radius of $R_{\text{pore}} = 0.8\sigma$ and a membrane thickness of $T_{\text{pore}} = 1.0\sigma$. Since this geometry is cylindrically symmetric, it is convenient to discuss the domain in cylindrical coordinates (r, z) , with z parallel to the axis of the channel. The particle simulations, however, were conducted in Cartesian coordinates.

The mesoscopic simulations were conducted on the domain $r \in [0, R_{\text{ch}}]$, $z \in [-0.5(Z_{\text{ch}} + T_{\text{pore}}), +0.5(Z_{\text{pore}} + T_{\text{pore}})]$. Thus the origin of the mesoscopic coordinate system was at the center of the channel. It is convenient to define $Z_{\text{pore}} = 0.5T_{\text{pore}}$ as the half-width of the membrane, $Z_{\text{eff}} = 0.5Z_{\text{ch}}$ as the half-length of the channel, and $Z_{\text{max}} = Z_{\text{ch}} + T_{\text{pore}}$ as the total length of the domain.

The walls of the channel were treated as reflecting boundary conditions for the mesoscopic polymers. Whenever a particle's displacement $\Delta\vec{x}$ would have resulted in it crossing a wall, its displacement was manually overridden as follows. For a particle that would have crossed the walls at $z = \pm Z_{\text{eff}}$, the particle was moved towards the interior of the domain in the z direction by twice its distance to the wall. For a particle that would have crossed the circular boundary at $r = R_{\text{eff}}$, the particle's displacement was rotated by $\pi/2$ in the appropriate direction to keep it inside domain.

The nanopores were not explicitly included in the channel model for particle motion.

C. Mesoscopic Electric Field Model

The electric field used in the microscopic simulations assumed no insulating boundaries other than the surface of the membrane containing the nanopore, i.e. it neglected the walls of the channel. This approximation is appropriate near the nanopore, but cannot hold for the mesoscopic model. As such, the electric field for the mesoscopic simulations was solved numerically using the FEniCS finite element solver¹⁶. The Python version of FEniCS 2016.2 available through Conda was used.

The polymers were assumed to have a negligible impact on the shape of the electric field. Since the channel is cylindrically symmetric, the channel was solved as a function of (r, z) . The electrostatic potential u was assumed to satisfy Laplace's equation,

$$\nabla^2 u = 0. \quad (22)$$

Finite element methods solve the variational formulation

of the PDE¹⁶. For a test function v ,

$$v(\nabla^2 u) = 0, \quad (23)$$

$$\int_{\Omega} v(\nabla^2 u) dV = 0, \quad (24)$$

$$\int_{\Omega} (\nabla v) \cdot (\nabla u) dV - \int_{\partial\Omega} \frac{\partial u}{\partial n} v dA = 0. \quad (25)$$

Since the test function is defined to be 0 on the boundary of the domain, the variational formulation of Laplace's equation for u amounts to

$$\int_{\Omega} (\nabla v) \cdot (\nabla u) dV = 0. \quad (26)$$

In cylindrical coordinates, the volume element is $dV = r dr d\phi dz$. Integrating over ϕ and dividing by 2π produces

$$\int (\nabla v) \cdot (\nabla u) r dr dz = 0. \quad (27)$$

Equation 27 was solved with FEniCS. The finite element mesh was created with the mshr FEniCS extension¹⁶. The mesh resolution in the mshr `generate_mesh` function was set to 200. Subsequently, the FEniCS `refine` command was applied to all cells containing a vertex at a position (r_0, z_0) satisfying $r_0 < 3R_{\text{pore}}$ and having z_0 within a distance $3Z_{\text{pore}}$ of either nanopore midline.

As in the microscopic field model, the nanopores were approximated as cylindrical. Dirichlet boundary conditions were used to impose $u(r, 0) = 0$ and $u(r, Z_{\text{max}}) = 1$ at the midlines of the entrance and exit nanopores. The walls were modelled as perfect insulators. Thus these boundaries were left unspecified, as this implies homogeneous Neumann boundary in finite element methods. A typical electric potential solution is shown in Figure 3.

The finite element solution was conducted using P2 basis elements¹⁶. The error of the solution was evaluated first by computing its Laplacian. The error of the solution was only significant near the corners of the nanopores. This had minimal impact on the simulation results, since particles never entered that region due to the strong electric field pushing them away from it as well as excluded volume interactions with the walls.

The conservation of electric flux in the solution u was tested as a second means of evaluating its accuracy. Its gradient was computed in FEniCS, and its z component E_z was projected onto the same function space used to solve u . The average of E_z for $z < Z_{\text{pore}}$ was used as a measure of the average electric field in the pore E_p , which should be approximately uniform and purely axial. By the conservation of electric flux, the electric field in the middle of the channel should approach a uniform and axial field of magnitude

$$E_c \approx E_p \left(\frac{R_{\text{pore}}}{R_{\text{ch}}} \right)^2. \quad (28)$$

A plot of $E_z(z)$ for all vertices in a typical mesh is shown in Figure 4, along with a line showing the average field in

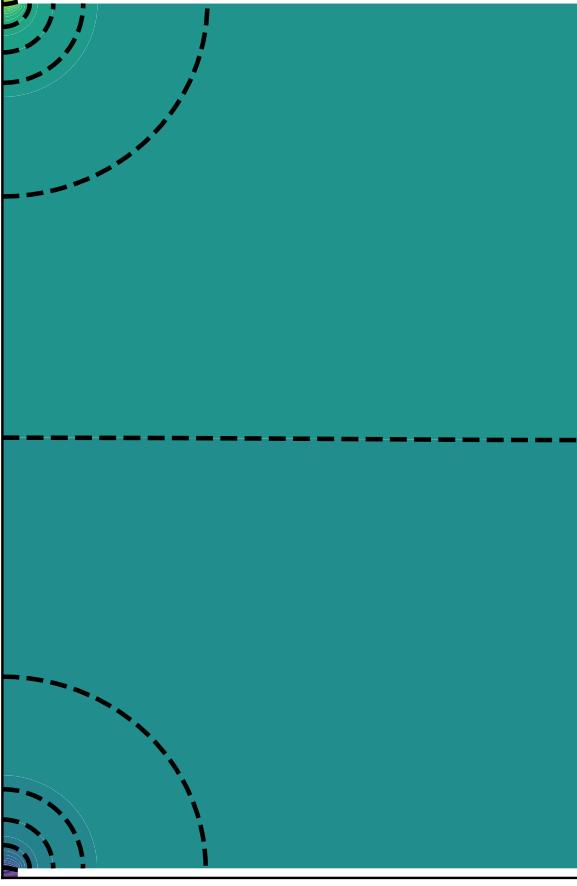


FIG. 3. Contour plot of an electric potential solution in (r, z) coordinates. Contours are drawn at 20%, 40%, 45%, 47%, 49%, and 50% of the voltage drop from either pore.

the pore and the theoretical value of E_c . The numerical solution shows excellent agreement with the conservation of flux, providing additional validation of its accuracy.

The electric potential solution was projected onto a rectangular mesh so that it could more easily be used in the particle simulations. The rectangular mesh spanned only the channel, omitting the nanopores, since the particle simulations only required this domain. This rectangular mesh had a resolution of $N_r = 200$ and $N_z = 400$ in the FEniCS `RectangleMesh` command. The solution u was projected onto this mesh using again P2 basis elements. After this projection, the Laplacian and conservation of flux tests were applied again to ensure the accuracy was not compromised.

From the electric potential on the rectangular mesh, the electric field was computed using numpy's `gradient` function, which uses a second-order central difference method for interior points and a first-order difference method at boundary points. The electric field was applied to the particle simulations using a nearest-

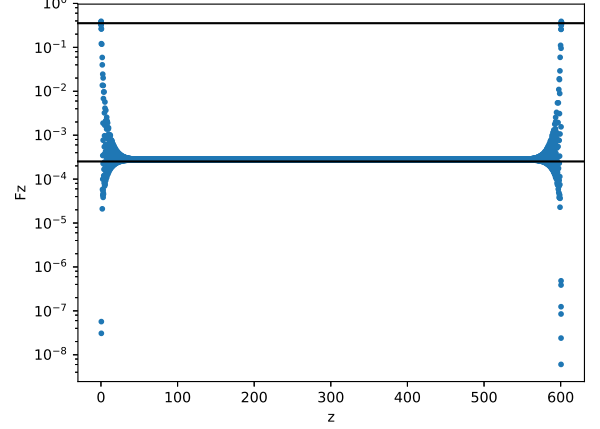


FIG. 4. Axial component of a typical electric field solution as a function of the axial coordinate z . The black horizontal lines show the average field in the pore (upper line) and the corresponding prediction for average field in the center of the channel (lower line).

neighbours approach. At each timestep, the position of each particle was rounded down to the nearest mesh point. This nearest-neighbour interpolation method was chosen because it was computationally efficient. The lower resolution compared to a more advanced interpolation method was deemed acceptable because the electric field changes very slowly for most of the channel domain, and because the system should not be very sensitive to small perturbations in the electric field shape.

D. Mesoscopic Simulation Procedure

The mesoscopic simulations were initialized with 10240 non-interaction particles positioned at the incoming nanopore of a channel (i.e. the point pore of higher potential energy). Particle positions were evolved as described in the previous sections. An absorbing boundary condition was placed at a distance of $r_{\text{cap}} = (N/4)\sigma$ from the outgoing nanopore. This distance corresponded to the starting radius of the microscopic simulations. Simulation proceeded until all particles crossed the absorbing boundary. The total duration of each trajectory, t_{meso} was recorded.

It is important to note that the methodology utilized to couple the microscopic and mesoscopic simulations assumes that polymers that cross the capture radius at r_{cap} never retract very far from the outgoing nanopore thereafter. Indeed, this is why the microscopic failure rate was measured: the values of r_{cap} was chosen by trial and error to keep this value low and roughly independent of N .

Furthermore, some inconsistency is possible in simulations where the microscopic cut-off radius is comparable

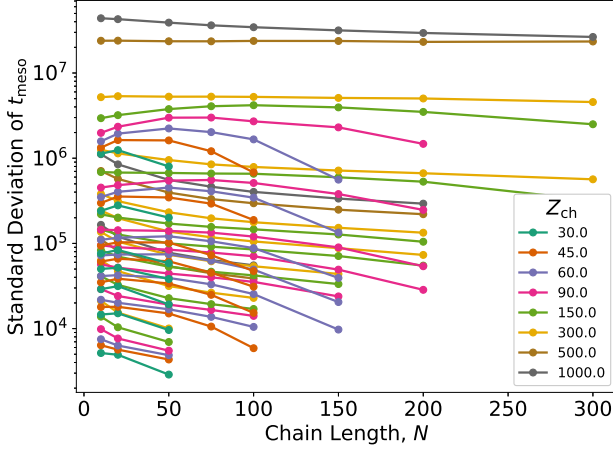


FIG. 5. Standard deviation of t_{meso} for a range of chain lengths in a variety of channel geometries.

to the radius of the channel. The microscopic simulations do not account for polymer interactions with the channel walls, nor is the electric field solution likely to be accurate in that region. Any error arising from this inconsistency is restricted to the simulations of large polymers in small channels. Even then, the discrepancy is expected to be small, as under such circumstances polymers were generally observed to remain close to the nanopore. The cuts described in the main text restrict the data analysis to cases where this error is expected to be tolerable.

Finally, the angular position on the capture radius at which particles were absorbed in the mesoscopic simulations was not recorded. The polymers in the microscopic simulations were initialized randomly at positions uniformly distributed on this hemisphere. Future work will explore the effect of the angle of approach of the incoming polymer on the translocation process. However, the electric field near the pore is primarily radial, so any angular dependence will arise mostly from interactions with the walls, and these are of secondary importance for most angles of approach.

E. Standard Deviation of the Mesoscopic Time

As described in the main text, the standard deviation of the mesoscopic time, σ_{meso} , contributes directly to the long-time macroscopic behaviour of the system. Figure 5 shows the standard deviations measured in the mesoscopic experiments. Figure 6 shows the same data under the normalizations that accounted for much of the variability in μ_{meso} .

Unfortunately, it is clear that the behaviour of the standard deviation of the mesoscopic time σ_{meso} is more complex than that of the mean mesoscopic time μ_{meso} . Whereas μ_{meso} seems to be captured, to first order, by just the two parameters V_{meso} and N^* , Figure 6 clearly

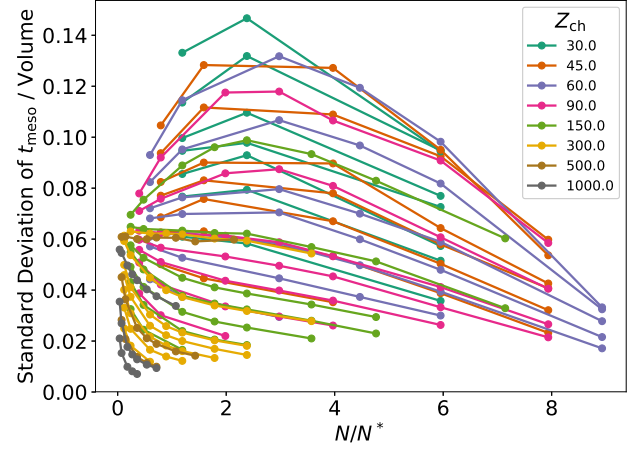


FIG. 6. Standard deviation of t_{meso} normalized by mesoscopic volume shown as a function of chain length normalized by N^* , the critical chain length described in the main text.

demonstrates that σ_{meso} depends non-trivially on additional parameters. Nonetheless, Figure 6 does highlight some interesting features. All geometries tend to agree for very short chains and very long chains. Furthermore, an important curve seems to divide the results. Larger channels lead to normalized standard deviations below this curve, whereas smaller channels lie above it. Overall, Figure 6 seems to illustrate that larger channels lead to more uniform g_0 and thus relatively smaller normalized standard deviations. More detailed interpretations are left to future work.

- ¹G. W. Slater, C. Holm, M. V. Chubynsky, H. W. de Haan, A. Dubé, K. Grass, O. A. Hickey, C. Kingsbury, D. Sean, T. N. Shendruk, and L. Zhan, *Electrophoresis* **30**, 792 (2009).
- ²G. S. Grest and K. Kremer, *Physical Review A* **33**, 3628 (1986).
- ³F. Farahpour, A. Maleknejad, F. Varnik, and M. R. Ejtehadi, *Soft Matter* **9**, 2750 (2013).
- ⁴H. W. de Haan, D. Sean, and G. W. Slater, *Physical Review E* **91**, 022601 (2015).
- ⁵K. Briggs, G. Madejski, M. Magill, K. Kastitis, H. W. de Haan, J. L. McGrath, and V. Tabard-Cossa, *Nano Letters* **18**, 660 (2018).
- ⁶D. Stigter, *Biopolymers* **16**, 1435 (1977).
- ⁷E. Sobel and J. Harpst, *Biopolymers* **31**, 1559 (1991).
- ⁸A. Savelyev, *Physical Chemistry Chemical Physics* **14**, 2250 (2012).
- ⁹A. R. Klotz, L. Duong, M. Mamaev, H. W. de Haan, J. Z. Chen, and W. W. Reisner, *Macromolecules* **48**, 5028 (2015).
- ¹⁰P. D. Ross and R. L. Scruggs, *Biopolymers* **2**, 231 (1964).
- ¹¹D. E. Smith, T. T. Perkins, and S. Chu, *Macromolecules* **29**, 1372 (1996).

- ¹²P. De Gennes, *Macromolecules* **9**, 587 (1976).
- ¹³S. C. Vollmer and H. W. de Haan, *The Journal of Chemical Physics* **145**, 154902 (2016).
- ¹⁴S. K. Lam, A. Pitrou, and S. Seibert, in *Proceedings of the Second Workshop on the LLVM Compiler Infrastructure in HPC* (ACM, 2015) p. 7.
- ¹⁵S. K. Lam, “GPU-Accelerated Graph Analytics in Python with Numba,” <https://devblogs.nvidia.com/gpu-accelerated-graph-analytics-python-numba/> (2015).
- ¹⁶M. Alnæs, J. Blechta, J. Hake, A. Johansson, B. Kehlet, A. Logg, C. Richardson, J. Ring, M. E. Rognes, and G. N. Wells, *Archive of Numerical Software* **3**, 9 (2015).

Chapter 4

Neural network solutions to differential equations in nonconvex domains: Solving the electric field in the slit-well microfluidic device

In *Neural network solutions to differential equations in nonconvex domains: Solving the electric field in the slit-well microfluidic device*, Magill, Nagel, and Haan [26] apply the neural network method (NNM) to solve a partial differential equation (PDE) model of the electric field in the slit-well device. This work builds upon an earlier effort to use the NNM to efficiently solve, store, and evaluate complicated electric force fields for subsequent use in MD simulations (App. C). Early NNM literature suggested that the NNM should be specifically well-suited for handling the challenging features arising in this application. Very few studies had actually investigated these claims directly, however, and Magill, Qureshi, and Haan [27] (included as App. C) uncovered a significant gap between

these expectations and the reality of applying the NNM to even a relatively simple biophysical electric field problem.

In response, Magill, Nagel, and Haan [26] present a systematic analysis of the behaviour of the NNM in this context, identifying the irregularity of the PDE solution as a major bottleneck to NNM performance. The irregularity arises as a direct consequence of the non-convexity of the domain—a geometric feature that is ubiquitous in biophysics (and many other fields). Ultimately, the study demonstrates that the NNM can indeed reliably solve such PDEs so long as it is applied and interpreted appropriately. One of the most significant results was the insight that the loss functional used for training in the NNM can also be used as a conservative *a posteriori* estimator of solution accuracy; this result is formalized by the proof in App. B. The published manuscript of Magill, Nagel, and Haan [26] is included in Sec. 4.3.

4.1 Motivation

Natural and synthetic biophysical systems commonly feature molecular transport driven by external electric fields. These electric fields can be quite complicated as a result of electrohydrodynamic effects, which induce nonlinear couplings between the electric field, the motions of ions in the solvent, and the motion of the molecules being driven by the field (see Sec. 2.3.1). Additionally, the electric fields are shaped by the boundaries of the domains across which they extend, which often have intricate geometries. One particularly important geometric challenge arises in domains that are non-convex, i.e., where portions of the domain boundary form re-entrant corners.

These physical features of biophysical electric fields correspond to challenging mathematical characteristics of the corresponding PDE problems. Non-convex domains often produce boundary layers and/or singularities in PDE solutions. Sophisticated electric field models like the Poisson-Nernst-Planck and Poisson-Boltzmann equations, which account for the coupling between molecular configurations and the external electric field, are generally nonlinear, highly parameterized, and high-dimensional. Although various methods exist for approaching these kinds of problems, they remain difficult and expensive to solve efficiently and reliably.

Moreover, in computational biophysics, solving these challenging PDE models of the electric field is often only the first step towards understanding the actual biophysical phenomena of interest. Afterwards, molecular dynamics simulations must generally be conducted in which the electric fields take the role of force fields driving molecular motion. The electric field solution must thus be evaluated repeatedly throughout the simulation on molecular configurations that are not known *a priori*. Traditionally, it is not feasible to precompute and tabulate field solutions for all allowable particle configurations. Instead, the electric field must be solved on-the-fly, i.e., repeatedly at every timestep of the molecular dynamics simulation. This incurs a very large computational cost.

In particular, the computation of external electric fields can interfere with the acceleration of molecular dynamics simulations using GPU processors. If field calculations are to be conducted on the GPU, this displaces the very limited GPU memory and directly reduces the effective number of parallel threads available for parallel computation of molecular trajectories. Conversely, sharing calculations

between GPU and CPU incurs substantial data transfer costs. The states of the field and/or molecules must be transferred to and from the CPU/GPU at every timestep, and can easily become the limiting factor in the overall computational efficiency of the simulation.

The nominal strengths of the NNM appear to align very well with the specific challenges posed by these biophysical electric field problems. As discussed in Sec. A.3, contemporary studies of the NNM are most commonly motivated by its performance on high-dimensional and highly parameterized problems, which could potentially enable the precomputation of complex force fields separately from subsequent MD simulations. Additionally, because the standard NNM formulation already uses a nonlinear trial function trained via nonlinear optimization, it requires no particular modifications to handle nonlinear equations. Finally, since the NNM is mesh-free and adaptive in nature, it has been promoted as being advantageous for handling complicated geometries. Specifically, other adaptive methods (such as the hp-FEM technique; see Sec. A.2) are potentially extremely memory-efficient.

These notions led to the conjecture that the NNM could automatically and easily learn low-memory representations of the external electric fields in MNFD systems, which could then be leveraged for on-board GPU force field calculations during subsequent molecular dynamics simulations. Despite the varied claims that the NNM should excel at problems with these features, very few authors have critically examined its performance on non-trivial problems exhibiting all of these features at once. Specifically, most earlier demonstrations of the NNM focused on linear PDEs, PDEs with analytic or otherwise well-behaved solutions,

and/or problems posed in simple (convex) geometries.

Magill, Qureshi, and Haan [27] (App. C) presented a preliminary attempt at capitalizing on the NNM to learn electric fields for MD—the results were disillusioning. In the presence of sharp re-entrant corners, the standard NNM formulation failed to train entirely. Approximating the re-entrant corners by circular arcs recovered a problem on which the NNM training could proceed. Nonetheless, the NNM still substantially underperformed relative to a simple FEM implementation. Control experiments in which neural networks were trained directly on the ground truth electric field solutions approached the scaling of the FEM solutions. This suggested two conclusions. First, the training dynamics of the NNM were an important barrier to replicating the convergence of standard non-adaptive FEM. Second, the NNM was very far from achieving the highly memory-efficient representations being pursued.

The substantial gap between expectations and reality uncovered by Magill, Qureshi, and Haan [27] prompted the more systematic investigation presented in Magill, Nagel, and Haan [26]. The study explores several basic questions about the NNM, such as

- At what rate do NNM solutions converge to the true PDE answer as the NNM loss decreases?
- At what rate does the NNM loss decrease with increases in training time, network width, and network depth?
- How can the accuracy of a given NNM solution be verified or guaranteed in practice?

- Do NNM solutions produce physically realistic solutions?
- Can NNM electric fields fruitfully be used as driving force fields in MD?

Prior to this work, such questions had rarely been carefully examined in the NNM literature, and had not been examined at all in the context of problems with irregular solutions like the one analysed by Magill, Nagel, and Haan [26].

4.2 Results

4.2.1 Irregularity is a bottleneck

The study focuses on solving the electric field in the slit-well MNFD, which is illustrated schematically in Fig. 2 of the manuscript (Sec. 4.3). As discussed above, Magill, Qureshi, and Haan [27] found that the standard NNM formulation applied to the standard PDE model for this electric field failed entirely to converge to a reasonable solution. Eventually, it was determined that the problem was due to singular behaviour in the true solution of the PDE in the vicinity of the re-entrant corners of the domain. At the interface of the deeper well regions with the shallow slit regions, the domain geometry is non-convex. In particular, the walls at this intersection of the wells and slits form a sharp 90-degree corner pointing into the domain. It is well-known that solutions to Laplace’s equation exhibit singularities at such corners.

This type of singularity is known to cause substantial problems for many other numerical methods as well, and various techniques have been developed to manage them. Because the solution is not twice continuously differentiable over

the domain, the strong form of Laplace’s equation is no longer well-posed. Thus, a weaker formulation of the problem may be expected to yield better numerical performance. Indeed, as discussed in the manuscript, various methods exist for addressing such singularities in the context of classical numerical methods. Unfortunately, these methods are not appropriate for the goals of this thesis. Such techniques would be difficult or impossible to extend to the Smoluchowski equation or to more sophisticated electric field models. Resorting to such tricks to solve Laplace’s equation would undermine the role of this study as a step towards understanding those more challenging PDE models.

Instead, Magill, Nagel, and Haan [26] overcame the problem by rounding the corners. This modification increases the regularity of the true solution to the PDE and is applicable to any PDE problem. With this relaxation in place, Magill, Nagel, and Haan [26] found that the standard formulation of the NNM performs reliably. However, as the radius of curvature of the circular arcs is made smaller, the optimization of the NNM solutions becomes increasingly difficult. Moreover, for any given curvature, the error after training is still concentrated at the interfaces of the circular arcs with the straight edges (Fig. 4(b) in the manuscript). The boundary is only finitely differentiable at these points, and thus the higher derivatives of the PDE solution are eventually discontinuous at these points. This result suggests a tendency for the NNM error to concentrate on regions of irregularity, even when this irregularity is not sufficient to compromise training entirely. As a result, far from being exceptionally effective at dealing with PDEs in complex geometries as previously promoted, the NNM in fact appears to struggle significantly to solve such problems at all.

4.2.2 Convergence of error with loss

After establishing that the NNM can indeed converge reliably on the PDE model with smoothed re-entrant corners, Magill, Nagel, and Haan [26] proceed to quantify the error of the NNM as a function of architectural hyperparameters. A large ensemble of networks was trained: for each combination of six depths and eight widths, four different randomly initialized networks were trained until the testing loss converged. For each trained network, a variety of error metrics were computed, and the behaviour of these error metrics was compared against the final testing loss of the networks.

The first two error metrics considered are the global relative L_2 error of the potential and the pointwise relative L_2 error of the field (Fig. 5). For all but the worst-performing networks, both relative error metrics are consistently at or below 1%, with some errors significantly smaller. This level of accuracy is generally considered adequate for many computational biophysical purposes. The intrinsic stochasticity of these systems, the modelling error incurred by common models, and the experimental errors faced by biotechnologies tend to exceed this threshold for most purposes.

However, it is noteworthy that the L_2 -based error metrics in Fig. 5 of the manuscript never converge to much lower than 0.1%. In contrast, FEM and other classical methods are often expected to converge to numerical precision. The NNM solutions are constrained to single precision (due to the GPU hardware utilized), but might still be expected to attain better accuracies than this.

Apart from a few very small networks, all the networks exhibiting worse than 1% error in Fig. 5 are shallow, i.e., having only one hidden layer. This reaffirms

the deep learning community’s general observation that, for non-obvious reasons, there is some fundamental difference between shallow and deep nets, even when the depth is only two, which translates into substantially better performance in practice.

The other remarkable result in Fig. 5 of the manuscript is that both relative errors appear to scale in direct proportion to the testing loss. More specifically, the ensemble of relative errors appear to be bounded above by a function that is proportional to the loss functional; the errors of individual networks fluctuate up to an order of magnitude below this upper envelope. This suggests that the loss functional can be used in practical application to define an *a posteriori* upper bound for L_2 errors. In fact, this result (and a generalization to other L_p error norms) is proven in App. B using the method of Green functions. Thus, quite broadly, the loss functional is (up to a problem-dependent constant) a conservative *a posteriori* upper bound of the solution error in L_p norms. This should be a valuable property for empirically verifying the convergence rate of the NNM when the true solution is not available.

4.2.3 No pathologically unphysical solutions

What can be said about error metrics other than those based on L_p norms? At the time this work was being conducted, there were concerns being raised in the computational physics community about the reliability of neural network techniques for use in physics research. The specific concern was that, since neural networks do not “know the physics” of a given problem, they might produce pathologically unphysical solutions that violate important physical symmetries or

invariances of the system under study. Worse, the concern was that neural networks might be capable of such pathological behaviour even while appearing to be accurate solutions. Essentially, this concern is that neural networks might exhibit some kind of severe and subtle overfitting behaviour, in which those loss functions used for training and validation would fail to detect serious errors.

In actuality, this concern applies to traditional numerical methods just as much as it applies to the NNM. For instance, ODE integration methods applied to Hamiltonian particle systems generally do not conserve energy; symplectic integrators were developed specifically to ensure conservation of energy up to numerical precision. Similarly, traditional PDE solution methods do not conserve flux in conservation law problems; the finite volume method specifically ensures this constraint is met. Another example is the case of Gibbs oscillation in the approximation of discontinuous functions by Fourier series: all finite truncations of the series exhibit persistent and pathological behaviour.

Conversely, neither symplectic integrators nor the finite volume method are necessarily the best numerical methods in practice. All numerical methods work with finite capacity to approximate target functions; exact satisfaction of one physical property generally comes at the expense of a loss of accuracy in some other aspect of the problem. Nonetheless, the example of Gibbs oscillation is a relevant illustration of how the NNM might fail problematically. Indeed, the standard formulation of the NNM is similar to Fourier methods in that both use analytic trial functions, and the irregularity of the slit-well electric field is precisely the type of problem on which pathological behaviours might arise.

Magill, Nagel, and Haan [26] present three sets of experiments aimed at

assuaging this concern about the NNM. First, it is shown that the NNM solutions recover the left-right (anti)symmetries of the true solution. Second, it is demonstrated that the NNM solutions satisfy conservation of electric flux approximately at all length scales; indeed, the NNM solutions outperform the FEM solutions in this respect over small length scales. Finally, the NNM solutions are used to guide particle simulations, and are found to perform comparably well to FEM solutions for this application. In all three cases, the performance of the NNM is consistently predicted by the loss functional. Altogether, these results provide further evidence that the NNM can be indeed trusted for use in computational physics.

4.2.4 Convergence of loss with capacity

Altogether, the results discussed above illustrate that the NNM loss functional is a robust indicator of solution performance, regardless of how this performance is measured. In particular, there is no indication that the NNM produces solutions with any kind of pathological unphysical characteristics. To a great extent, it appears the problem of understanding and improving NNM performance can safely be reduced to studies of just the loss functional. Thus, Magill, Nagel, and Haan [26] examine in Fig. 10 of the manuscript how the testing loss converges with increases in the memory consumption of the NNM networks. This is the direct counterpart of the scaling experiments conducted by Magill, Qureshi, and Haan [27] and the convergence results typically discussed for numerical methods like FEM. Magill, Nagel, and Haan [26] identify three regimes of convergence: one in the low-capacity regime, a second for networks of moderate capacity, and a

final regime for very high-capacity networks.

In the low-capacity regime, loss decreases fairly rapidly with increasing capacity. Magill, Nagel, and Haan [26] interpret this in the natural way: performance here is bottlenecked by limited memory capacity, so increasing capacity directly enables better performance. It is also clear in Fig. 10 that, for a fixed capacity, the loss decreases with depth. This again reflects the mysterious advantage of deep networks over shallow networks. However, the advantage diminishes rapidly with increasing depth: networks with 3 or more layers have essentially the same loss for a given capacity.

At moderate capacities, the equivalence of deep networks of different depths is even more pronounced. All the networks with 2 or more hidden layers achieve essentially the same loss at the same capacity. Shallow networks are difficult to train at all in this regime, but do appear to perform roughly two orders of magnitude worse than deep networks. In fact, loss appears to stop decreasing at all with increasing capacity. Thus, in this regime, neither increasing the depth nor the width of architectures improves loss below roughly 10^{-5} . As noted above, these losses corresponds to relative errors on the order of 0.01-0.1%.

Finally, increasing the capacity substantially more actually leads to deteriorations in the performance of the NNM. In this high-capacity limit, loss is still essentially independent of depth, but grows monotonically when capacity is increased.

The plateau in loss (and therefore error) at large capacities is a limiting factor to achieving lower error values with the NNM. There are many possible explanations for this behaviour. As noted above, the NNM studied here was

implemented in single precision, whereas the FEM solutions were computed in double precision. Besides this, modern theories of deep learning seem to suggest that deep neural networks can become degenerate when they are too wide at a fixed depth. Conversely, fully-connected networks using tanh activation functions may be fundamentally difficult to train beyond a depth of 5 or 6; this practical limit on depth has been reported empirically by Berg and Nyström [4] as well. Further investigations are needed to clarify the limiting factor, or factors, preventing further convergence of the NNM loss at high capacities.

Regardless of this matter, certain conclusions can be drawn with respect to the goal of using the NNM for electric fields in MD. For this application, errors on the order of 0.1-0.01% are likely more than sufficient. What is more problematic is the slow convergence of the NNM with respect to increasing capacity. In Fig. 10, the best scaling regime of loss with respect to capacity is only slightly better than linear: loss (and therefore error) converges nearly in direct proportion with capacity in the low-capacity regime for networks with two hidden layers. This is a far cry from the exponential convergence expected for traditionally adaptive methods, like hp-FEM. It is equally distant from the best achievable expressivity expected from some deep learning theories (Sec. A.3). Indeed, this same gap has now been reported by multiple authors, as has become known as the theory-to-practice gap. Some insights into the mechanisms of this gap may be gleaned from the discussion in Chapter 5. What is clear in any case is that a fundamentally different training paradigm¹ is likely necessary to achieve very

¹Neural architecture search methods bear some resemblance to adaptive FEM mesh refinement methods, and may be a promising starting point for such a research initiative. Pruning methods and similar post-processing compression techniques also seem worth exploring.

memory-efficient representations of PDE solutions with the NNM.

4.3 Manuscript

Neural network solutions to differential equations in nonconvex domains: Solving the electric field in the slit-well microfluidic device

Martin Magill^{1,2}, Andrew M. Nagel,¹ and Hendrick W. de Haan^{1,*}

¹Faculty of Science, University of Ontario Institute of Technology, 2000 Simcoe St N, Oshawa, Ontario, Canada L1H7K4

²Vector Institute, 661 University Ave Suite 710, Toronto, Ontario, Canada M5G1M1



(Received 29 April 2020; accepted 21 June 2020; published 21 July 2020)

The neural network method of solving differential equations is used to approximate the electric potential and corresponding electric field in the slit-well microfluidic device. The device's geometry is nonconvex, making this a challenging problem to solve using the neural network method. To validate the method, the neural network solutions are compared to a reference solution obtained using the finite-element method. Additional metrics are presented that measure how well the neural networks recover important physical invariants that are not explicitly enforced during training: spatial symmetries and conservation of electric flux. Finally, as an application-specific test of validity, neural network electric fields are incorporated into particle simulations. Conveniently, the same loss functional used to train the neural networks also seems to provide a reliable estimator of the networks' true errors, as measured by any of the metrics considered here. In all metrics, deep neural networks significantly outperform shallow neural networks, even when normalized by computational cost. Altogether, the results suggest that the neural network method can reliably produce solutions of acceptable accuracy for use in subsequent physical computations, such as particle simulations.

DOI: [10.1103/PhysRevResearch.2.033110](https://doi.org/10.1103/PhysRevResearch.2.033110)

I. INTRODUCTION

Many important phenomena can be modeled effectively by partial differential equations (PDEs) with appropriate boundary conditions (BCs). When PDE problems are posed in domains with complicated geometries, they are often too difficult to be solved analytically, and must instead be approximated numerically. The standard tools for numerically solving PDE problems in complex geometries are mesh-based approaches, such as the finite-element method (FEM) [1]. In these methods, the problem domain is decomposed into a mesh of smaller subdomains, and the solution is approximated by a linear combination of simple, local functions.

In this work, we will explore a less common numerical solution method for PDE problems, which we will refer to as the neural network method (NNM) [2]. In the NNM, the solution is directly approximated by a neural network (e.g., Fig. 1), rather than by a linear combination of local basis functions. In a process called training, the network parameters are varied until it approximately satisfies the PDE and BCs.

The purpose of this study is to investigate the effectiveness of the NNM on a problem exhibiting a complicated geometry. Specifically, the NNM is used to solve a model of the electric field in the slit-well microfluidic device, which is an applica-

tion of active research interest [3–6]. The problem domain is nonconvex, and the electric field is discontinuous in the limit of sharp corners. Despite the growing popularity of the NNM, relatively few authors have validated it on problems with such ill-behaved solutions. The rest of this Introduction provides an overview of the NNM, including its previous use to study systems similar to the slit-well, as well as a review of the slit-well device itself.

A. Neural network method

The neural network method of solving differential equations was first published by Dissanayake and Phan-Thien [2], and belongs to the broader family of techniques known as methods of weighted residuals [2,7]. Around the same time, Meade Jr. and Fernandez [8] separately demonstrated a variant of the NNM that did not use iterative training, and instead solved a system of linear equations for the network weights; it was, however, designed for solving only ordinary differential equations. Later, van Milligen *et al.* [9] independently proposed a method quite similar to the original approach by Dissanayake and Phan-Thien [2], to solve second-order elliptic PDEs describing plasmas in tokamaks. The NNM was proposed independently again by Lagaris *et al.* [10]. Their modified methodology embedded the neural network within an ansatz that was manually constructed to exactly satisfy the boundary conditions; however, this form is challenging to construct when the boundary conditions or the domain geometry are complicated. Many authors have since contributed to the development of the NNM, and Yadav *et al.* [11] published a book reviewing much of the early work on the NNM.

*Hendrick.deHaan@uoit.ca

Published by the American Physical Society under the terms of the [Creative Commons Attribution 4.0 International](https://creativecommons.org/licenses/by/4.0/) license. Further distribution of this work must maintain attribution to the author(s) and the published article's title, journal citation, and DOI.

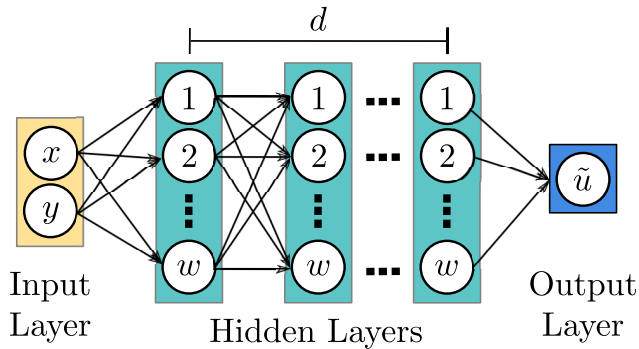


FIG. 1. Schematic of a fully connected feed-forward neural network of depth d and width w mapping coordinates (x, y) to an output $\tilde{u}(x, y)$. Each node computes a weighted sum of its incoming arrows, and the result (plus a bias) is passed to an activation function. In the NNM, the parameters are optimized to make $\tilde{u}(x, y)$ approximately satisfy a target PDE and its BCs.

The NNM has various potential appeals over more common methods like FEM. For instance, the NNM is mesh free, and generally produces uniformly accurate solutions throughout the PDE domain [11,12]. Whereas earlier implementations used shallow neural networks (i.e., those having only one hidden layer), many authors have recently noted the significant benefits of using deep architectures [13–26]. In particular, it appears that the NNM with deep neural networks performs remarkably well in high-dimensional problems [13–15,17–19,21–27]. Such high-dimensional PDEs are typically intractable using FEM and most traditional methods. These suffer from the so-called curse of dimensionality, in which computational cost grows exponentially with the number of dimensions. In addition to the above empirical demonstrations of the NNM, several theorems have been published stating that the computational cost of the NNM grows at most polynomially in the number of dimensions for various classes of PDEs [28–30].

Nonetheless, the theoretical grounding of the NNM is less thoroughly developed than those of other techniques. There are as of yet few guarantees regarding, e.g., under what conditions the NNM will converge to the true solution of a given PDE, at what rate, and to what precision. As such, confidence in the method still relies heavily on empirical demonstrations. However, available empirical demonstrations focus primarily on problems with relatively well-behaved solutions [15,16,18,19,21–26,31]. Indeed, Michoski *et al.* [32] noted this, and conducted an investigation of the NNM applied to irregular problems exhibiting shocks. This work is analogous in this regard, but focuses instead on the nonconvexity of the slit-well domain as the source of irregularity.

B. Slit-well microfluidic device

Microfluidic and nanofluidic devices (MNFDs) are small, synthetically fabricated systems with applications in molecular detection and manipulation [5,6,33–35]. One important use of MNFDs is to sort mixtures of molecules, including free-draining molecules such as DNA that cannot normally be separated electrophoretically in free solution [6]. For instance,

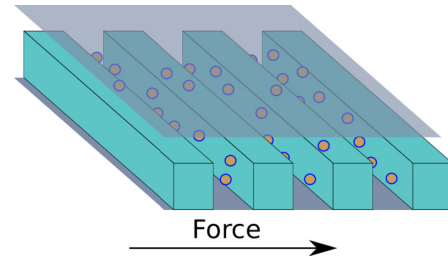


FIG. 2. A schematic of particles being electrically driven through the slit-well device.

the slit-well device proposed by Han and Craighead [3] can be used for sorting polymers (such as DNA [3,4,36,37]) or nanoparticles [38,39]. The device’s periodic geometry, illustrated schematically in Fig. 2, consists of parallel channels (called wells) separated by shallower regions (called slits). An electric field is applied to drive molecules through the device.

MNFDs such as the slit-well exploit the complexity of physical phenomena at the single-molecular scale (often below the optical resolution limit) to produce useful and sometimes surprising behaviors. This, however, makes them challenging to design and optimize, and renders theoretical and computational investigations important to the development of MNFD technologies. For example, the sorting mechanism in the slit-well device depends nonlinearly on the magnitude of the applied electric field as well as the size and shape of the wells, the slits, and the molecules themselves [6,36–40]. For some choices of these parameters, the slit-well sorts molecular mixtures into increasing order of size; for others, however, it sorts them into decreasing order. A rich literature exists exploring these processes, reviewed in part by Dorfman [6] and Langecker *et al.* [40].

C. NNM with complicated geometries

There are relatively few demonstrations of the NNM on problems with complicated domain geometries. Specifically, the NNM has mostly been applied to problems posed in rectangular or circular domains [15,18,19,21–23,25,26]. Of note, Wei *et al.* [27] used the NNM to solve PDEs in nanobiophysics that also arise in MNFDs (i.e., Fokker-Planck for particles and polymers). However, their work did not consider these problems in MNFD geometries. Even among the demonstrations of the NNM in more complicated (e.g., nonconvex) domain geometries, most problems feature boundary conditions that produce relatively smooth, well-behaved solutions [16,24,31]. Sirignano and Spiliopoulos [17] solved a free-boundary problem based on a financial system, but it is not clear whether that PDE exhibits the specific kinds of challenging features considered in this work.

An exception to the above is given by E *et al.* [14], who applied a variant of the NNM to a Poisson equation in a square domain with a reentrant needlelike boundary. This problem exhibits the same singular behavior as the slit-well problem with sharp corners (see Sec. II A). Their Deep Ritz training protocol was based on a variational formulation of Poisson’s equation. However, variational formulations cannot be obtained for all PDEs [41]. For this reason, we have

opted to study the more general NNM algorithm originally presented by Dissanayake and Phan-Thien [2].

When Anitescu *et al.* [42] revisited this needle problem using the original method of Dissanayake and Phan-Thien [2], they reported poorer convergence than obtained by E *et al.* [14] with the Deep Ritz method. A similar observation was made during this work: reentrant corners significantly impair the convergence of the standard NNM (Sec. II A). In contrast to this work, the error analyses reported by E *et al.* [14] and Anitescu *et al.* [42] did not consider the physical realism of the NNM solutions (Sec. ID) nor the accuracy of the NNM solutions' gradients. These characteristics of the NNM are important for use in various applications, including studies of MNFDs, and are investigated directly in this work.

D. Physical realism of NNM solutions

Various modifications of the NNM have been proposed to ensure solutions exactly satisfy problem-specific invariants that are known *a priori*, such as boundary conditions [12,16,31], non-negativity [43], Hamiltonian dynamics [44], or special invariants of the Schrödinger equation [45]. However, manually creating formulations of the NNM that explicitly satisfy specific invariants can be difficult. Furthermore, this approach cannot account for invariants which may be unknown ahead of time. It is natural to question how well the NNM approximates invariant quantities when these are not explicitly enforced.

In fact, although certain numerical methods can be devised specifically to satisfy some conservation laws [e.g., finite volume methods conserve flux [46], symplectic ordinary differential equation (ODE) integrators conserve energy [47]], most numerical methods (including standard FEM formulations) do not satisfy physical invariants exactly. For instance, Zhang *et al.* [48] discussed what modifications of the FEM are necessary to render it flux conserving. As part of this work, we will investigate how well the NNM satisfies physical invariants of the slit-well problem in the absence of any problem-specific customization.

II. METHODOLOGY

A. Problem statement

We use the simplest electrostatic model of the electric field \mathbf{E} in the slit-well, namely, the two-dimensional Laplace equation for the electric potential u . Figure 3 illustrates the geometry of our model over one periodic subunit of the slit-well device. Uniform Dirichlet boundary conditions were imposed on the colored segments (specifically, $u = \pm 1$ on the right and left, respectively) to model an applied voltage across the system. The gray boundaries correspond to homogeneous Neumann (i.e., insulating) boundary conditions. Throughout the interior of the domain (i.e., the yellow area in Fig. 3), the potential was modeled by Laplace's equation.

In contrast with other authors, we have rounded the reentrant corners at the interface of the slits and wells. It can be shown that near sharp (i.e., nondifferentiable) reentrant corners, solutions u to Laplace's equation are not continuously differentiable [49–51]. That is, sharp reentrant corners cause singularities in the electric field \mathbf{E} . Because the magnitude of

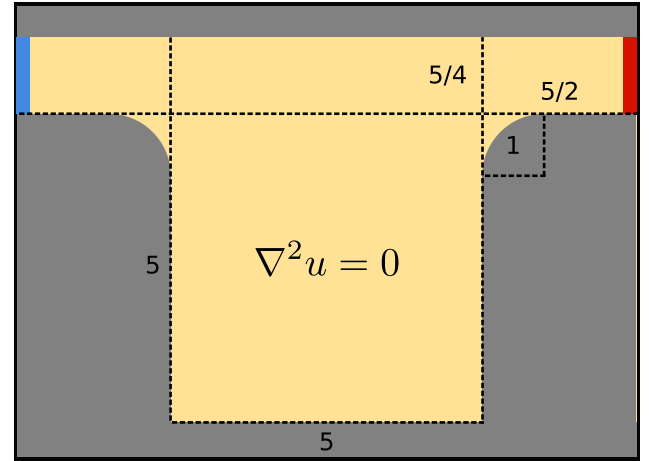


FIG. 3. A cross-sectional view of the slit-well device illustrating our PDE model of the electric potential in one periodic subunit of the device. The reentrant corners follow circular arcs, and the numbers indicate the lengths of each dotted line. The solution satisfies Laplace's equation in the yellow region, Dirichlet conditions on the red and blue boundaries, and homogeneous Neumann conditions on the gray boundaries.

\mathbf{E} near the corners diverges as the curvature goes to zero, the slit-well electric field is ill conditioned, in the sense that small changes in the curvature of the corners produce large changes in \mathbf{E} .

Although such ill conditioning hinders the performance of most numerical methods, including FEM [49–51], they present a particular challenge for the NNM. The fully connected feed-forward neural networks typically used for the NNM are infinitely differentiable functions. However, the true solution to the slit-well problem with sharp corners exhibits a discontinuous electric field, so that significant errors seem likely near the corners. Furthermore, because the neural network is a global approximation method, local errors near the corners can affect performance throughout the domain.

In practice, the training methodology we present here (Sec. IIB), when applied to the problem with sharp corners, failed to converge to even a reasonable approximation of the true solution. Even in preliminary tests with rounded corners, the convergence rate of the NNM was observed to deteriorate as the curvature of the corners was reduced. Therefore, for this work, an intermediate curvature (Fig. 3) was selected to produce a challenging but attainable benchmark for the NNM.

B. NNM implementation

In this section, we describe our implementation of the NNM. It is similar to those of Dissanayake and Phan-Thien [2], van Milligen *et al.* [9], Berg and Nyström [16], Sirignano and Spiliopoulos [17], Magill *et al.* [20], and Wei *et al.* [27], among others. The true solution $u(\mathbf{x})$ to the PDE was directly approximated by a neural network $\tilde{u}(\mathbf{x})$. This was accomplished by training the neural network to minimize the loss functional

$$\mathcal{L}[\tilde{u}] = \int_{\Omega} (\nabla^2 \tilde{u})^2 dA + \int_{\partial\Omega} (B[\tilde{u}])^2 ds. \quad (1)$$

Here, $\nabla^2 u = 0$ is the PDE required to hold in the interior of the domain $\Omega \subset \mathbb{R}^2$, and B is a differential operator describing the boundary conditions $Bu = 0$ on the boundary $\partial\Omega$ of the domain (described in Sec. II A and illustrated in Fig. 3). Thus, $\mathcal{L}[\tilde{u}]$ quantifies the extent to which the neural network fails to satisfy the PDE and its boundary conditions.

The parameters of the network were updated iteratively using the Adam optimizer, a modified gradient descent algorithm [52]. The integrals in $\mathcal{L}[\tilde{u}]$ were approximated via the Monte Carlo method, as described in more detail below. The approximate electric field $\tilde{\mathbf{E}}$ and other required derivatives were obtained exactly via automatic differentiation. The weights of the network were initialized by the Glorot method [53]. Computations were done using TENSORFLOW 1.13, and all hyperparameters not discussed here were set to their default values [54].

The Monte Carlo samples $\mathbf{x}_i \in \Omega$ used to estimate the first term of $\mathcal{L}[\tilde{u}]$ were selected from 100 000 points uniformly distributed in the bounding rectangle $[-L_x, L_x] \times [-L_y, L_y]$, by rejecting those lying outside the domain. Those used to estimate the second term were generated by directly sampling the boundary with a linear density of 40 points per unit length. Altogether, this yielded an expected batch size of roughly 62 000. To reduce the overhead of sampling training points, batches were reused for 1000 parameter updates before resampling.

The testing loss was computed on a set of points sampled once at the beginning of training, generated using the same procedure as the training points. The testing loss was computed and recorded every 100 parameter updates. Early stopping was used to terminate training when the testing loss failed to improve after 100 consecutive tests. The final network was taken from the epoch at which the testing loss was smallest. This training procedure was conceived to ensure that networks converged to very stable local minima, in order to study the behavior of the NNM in the limit of long training time.

The neural networks considered in this study were all fully connected feed-forward networks with tanh activation functions (Fig. 1), consisting of d hidden layers of equal width w . Specifically, the networks mapped an input vector \mathbf{x} , corresponding to a point in the problem domain, to \tilde{u} given by

$$\tilde{u}(\mathbf{x}) = f_{d+1} \circ f_d \circ \cdots \circ f_1(\mathbf{x}), \quad (2)$$

where

$$f_1(\mathbf{x}) = \tanh(W_1 \mathbf{x} + \mathbf{b}_1), \quad (3)$$

$$f_i(\mathbf{x}) = \tanh[W_i f_{i-1}(\mathbf{x}) + \mathbf{b}_i], \quad i = 2 \dots d \quad (4)$$

$$f_{d+1}(\mathbf{x}) = \mathbf{W}_{d+1} f_d(\mathbf{x}) + b_{d+1}. \quad (5)$$

Here, $W_1 \in \mathbb{R}^{w \times 2}$, $W_i \in \mathbb{R}^{w \times w}$ for $i = 2 \dots d$, and $W_{d+1} \in \mathbb{R}^{1 \times w}$ are the network's weight matrices, while $\mathbf{b}_i \in \mathbb{R}^w$ for $i = 1 \dots d$, and $b_{d+1} \in \mathbb{R}$ are its biases.

C. FEM implementation

To provide a reliable ground truth against which to compare the performance of the NNM, the target PDE was also

solved via the FEM using FENICS [55]. The domain and mesh were constructed using the MSHR package. The resolution parameter for `generate_mesh` was set to 200 and the circular reentrant corners were approximated linearly with 100 segments each.

In order to obtain an accurate approximation of the electric field, and not just of the electric potential, the FEM was applied to a standard dual-mixed formulation of Laplace's equation for the electric field and electric potential simultaneously [55]. In this approach, \tilde{u} and $\tilde{\mathbf{E}}$ are approximated simultaneously using separate basis functions. Solving for \tilde{u} alone and reconstructing $\tilde{\mathbf{E}}$ by differentiation was found to yield poor results.

Convergence tests (not shown) confirmed that the FEM solution converged in proportion to the square of the mesh resolution. The tests suggest that the absolute error in the FEM solution relative to the true solution is on the order of machine precision (i.e., 10^{-16}). Note that the FEM solution was computed in double precision, whereas the NNM was computed in single precision.

III. RESULTS

At its core, the NNM is motivated by the rationale that training networks to minimize the loss functional [Eq. (1)] will cause those networks to approximate the correct solution. This section contains investigations into the following related questions:

(1) If a network exhibits a small loss, how close is it to the true solution? Specifically, is the loss functional a reliable estimator of actual network performance?

(2) If a network is close to the true solution, how well does it reproduce the physical characteristics of the true solution? Specifically,

(a) to what extent does it exhibit the same spatial symmetries as the true solution?

(b) to what extent does it conserve electric flux?

(3) If a network is close to the true solution, and the corresponding electric field is used to conduct particle simulations, how accurate are subsequent measurements made using those particle simulations?

(4) How does architecture affect these conclusions?

All experiments were repeated across four random initializations and multiple network architectures: specifically, all combinations of depths $d = 1, 2, 3, 4, 5, 6$ and widths $w = 10, 25, 50, 75, 100, 150, 200, 250$ were examined, as well as networks of depth 1 and widths 500 and 1000.

Figure 4(a) shows an example of an NNM solution obtained using a network of depth 5 and width 75. The approximate electric field $\tilde{\mathbf{E}}$ is superimposed in black lines over colored contours showing the approximate electric potential \tilde{u} . It is visually indistinguishable from the reference FEM solution (not shown). Figure 4(b) shows $(\tilde{u} - u)^2$, the squared error of the NNM potential compared to the FEM potential. Figure 4(c) shows $\|\tilde{\mathbf{E}} - \mathbf{E}\|_2 / \|\mathbf{E}\|_2$, the pointwise relative error of the NNM electric field. Here, $\|\cdot\|_2$ denotes the Euclidean norm. Note that the error in the potential cannot be normalized pointwise, as discussed in the next section.

Both of the error distributions in Fig. 4 are particularly pronounced near the reentrant corners. The electric field intensity

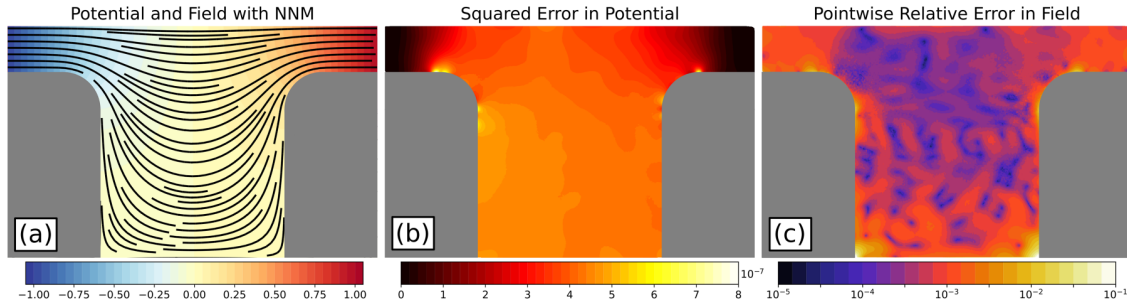


FIG. 4. Example NNM solution using 5 hidden layers of width 75. (a) Approximate electric potential (colored contours) and electric field (solid black lines). (b) Squared error of the electric potential. (c) Pointwise relative error in the electric field (note the logarithmic color scale). The errors in plots (b) and (c) are interpolated from values evaluated on the FEM mesh points.

is also very large in these regions (see Fig. 12). In the limit of small curvature, in fact, it is at these corners that the electric field develops singularities (see Sec. II A). In fact, the peaks in error and electric field intensity both occur precisely where the boundary transitions from flat to curved, i.e., where the second derivative of the boundary curve is discontinuous.

Additionally, Fig. 4(c) shows pronounced relative error in the electric field near the corners at the bottom of the well. These peaks arise because the magnitude of the true electric field approaches zero in those corners (see Fig. 12). Since the denominator of $\|\tilde{\mathbf{E}} - \mathbf{E}\|_2 / \|\mathbf{E}\|_2$ is very small, even small errors in the electric field near those corners manifest as large relative error. The maximum relative error in the domain Ω consistently occurred in these bottom-most corners for all NNM solutions in the data set. Nonetheless, for many applications, errors of this kind are likely to be less important than the errors occurring near the reentrant corners, as they are much smaller in absolute magnitude.

A. Error relative to FEM

The purpose of this section is to investigate the errors of the NNM solutions relative to the reference FEM solution, and to what extent the loss functional correlates with these errors. The error in an approximate electric potential \tilde{u} will be characterized by

$$\delta_u[\tilde{u}] = \sqrt{\frac{\langle (\tilde{u} - u)^2 \rangle_\Omega}{\langle u^2 \rangle_\Omega}}. \quad (6)$$

Here, $\langle \cdot \rangle_\Omega$ denotes the mean over the domain Ω . Whereas Fig. 4(b) shows the distribution of the squared error in \tilde{u} throughout the domain, $\delta_u[\tilde{u}]$ corresponds to the root-mean-squared error of \tilde{u} over Ω , normalized by the root-mean-squared value of the true solution u . Note that one cannot define an unambiguous pointwise relative error for \tilde{u} since the electric potential does not have a physically meaningful zero. The metric $\delta_u[\tilde{u}]$ represents the magnitude of the error in \tilde{u} relative to the magnitude of the true solution u , when both of these are measured in the L^2 norm for functions.

For the electric field, conversely, a meaningful pointwise relative error can be defined as $\|\tilde{\mathbf{E}} - \mathbf{E}\|_2 / \|\mathbf{E}\|_2$, where both the numerator and the denominator vary throughout the

domain. The average of this pointwise relative error is denoted

$$\delta_E[\tilde{\mathbf{E}}] = \left\langle \frac{\|\tilde{\mathbf{E}} - \mathbf{E}\|_2}{\|\mathbf{E}\|_2} \right\rangle_\Omega, \quad (7)$$

and acts as a global error metric for $\tilde{\mathbf{E}}$. This is precisely the mean of error distributions like the one shown in Fig. 4(c).

Figure 5 shows the global error metrics $\delta_u[\tilde{u}]$ and $\delta_E[\tilde{\mathbf{E}}]$ for all networks in the data set, plotted against each network's testing loss. The integrals required to compute the error metrics were approximated via the Monte Carlo method, by sampling the domain interior using the same procedure

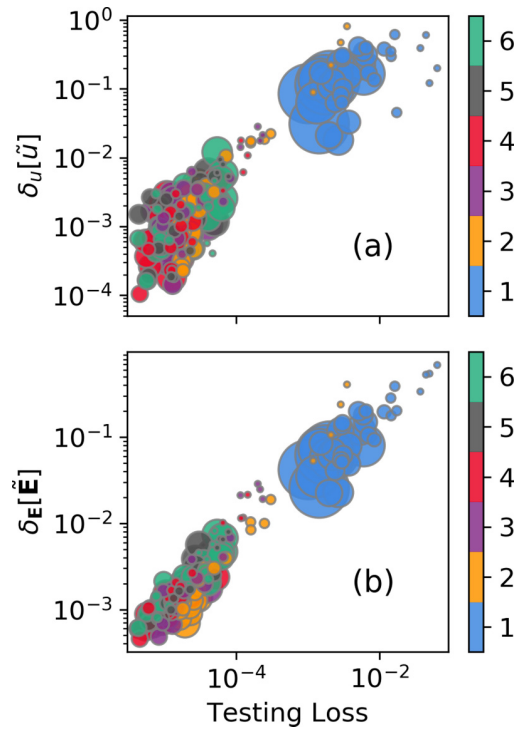


FIG. 5. Global error metrics for the NNM solutions relative to the reference FEM solution, shown against testing loss for a variety of network architectures. (a) The relative error of the electric potentials $\delta_u[\tilde{u}]$. (b) The relative error of the electric fields $\delta_E[\tilde{\mathbf{E}}]$. Marker color indicates the depth of the network, and marker area indicates its width.

described in Sec. II B. Marker color corresponds to network depth, and marker size corresponds to network width.

It is clear in Fig. 5 that lower testing losses correlate strongly with lower values of both $\delta_u[\tilde{u}]$ and $\delta_E[\tilde{\mathbf{E}}]$. This result confirms the basic motivation underlying the NNM, namely, that training neural networks to minimize the loss functional will cause them to approximate the correct solution. It also suggests that, in the absence of theoretical guarantees on the convergence of the NNM, the testing loss may provide a practical proxy for estimating a solution's true accuracy.

The data in both Figs. 5(a) and 5(b) partition conveniently into two clusters. The upper-right clusters consist of those networks achieving relative errors worse than 1% in both $\delta_u[\tilde{u}]$ and $\delta_E[\tilde{\mathbf{E}}]$. This population contains all of the shallow network architectures, suggesting that at least two hidden layers are required to achieve good performance on this problem. Furthermore, as discussed in Sec. III D, shallow networks always underperform relative to deep networks, even when normalized by capacity. The narrowest of the deep network architectures also attain relative errors worse than 1%. This implies that even with two hidden layers, networks require some minimum capacity (i.e., memory consumption) in order to achieve good performance on this problem.

The lower-left clusters in Figs. 5(a) and 5(b) contain the majority of the data set, and consist of those networks attaining relative errors below 1% in both $\delta_u[\tilde{u}]$ and $\delta_E[\tilde{\mathbf{E}}]$. The best networks achieved relative errors as low as $\delta_u[\tilde{u}] \approx 0.01\%$ and $\delta_E[\tilde{\mathbf{E}}] \approx 0.1\%$. For reference, the example solution shown in Fig. 4 corresponds to a testing loss of $\mathcal{L}[\tilde{u}] \approx 9 \times 10^{-6}$, and error values of $\delta_u[\tilde{u}] \approx 0.2\%$ and $\delta_E[\tilde{\mathbf{E}}] \approx 0.08\%$. A variety of architecture choices (i.e., depths and widths) produce comparably good performance, suggesting that the NNM can produce accurate solutions without the need for careful architecture tuning. This is explored further in Sec. III D.

B. Physically motivated error metrics

The results in the previous section suggest that the NNM can reliably produce accurate solutions to the slit-well problem. Furthermore, networks with smaller loss values are closer to the true solution, i.e., they have smaller error values. Finally, the NNM does not appear overly sensitive to the choice of architecture, given at least two hidden layers and sufficient network width.

The purpose of this section is to investigate whether networks with small loss and error values also approximately reproduce physical characteristics of the true solution. Specifically, we investigate the NNM solutions' satisfaction of spatial symmetries and the conservation of electric flux.

1. Deviation from symmetry

The true solution of the target PDE satisfies three spatial symmetries. First, the true electric potential u is antisymmetric in the horizontal direction about the center of the well, i.e.,

$$u(x, y) = -u(-x, y), \quad (8)$$

where (x, y) are the coordinates of a point about the center of the well. As a result, the vertical component of the true electric

field \mathbf{E} also exhibits this antisymmetry in x , i.e.,

$$E_y(x, y) = -E_y(-x, y). \quad (9)$$

Finally, the horizontal component of the electric field is symmetric about the center of the domain, i.e.,

$$E_x(x, y) = E_x(-x, y). \quad (10)$$

The extent to which a network deviates from these symmetries will be quantified using relative error metrics analogous to those used in the previous section. Specifically, the deviation of an approximate electric potential \tilde{u} from symmetry will be quantified by

$$R_u[\tilde{u}] = \sqrt{\frac{\langle (\tilde{u} - \tilde{u}')^2 \rangle_\Omega}{\langle u^2 \rangle_\Omega}}, \quad (11)$$

where $\tilde{u}'(x, y) = -\tilde{u}(-x, y)$. This is the root-mean-squared difference between \tilde{u} and its negative reflection, normalized by the root-mean-squared value of the true potential u . In analogy with $\delta_u[\tilde{u}]$, the metric $R_u[\tilde{u}]$ measures the magnitude of the deviation of \tilde{u} from symmetry relative to the magnitude of the true solution u (when both are measured in the L^2 norm). The deviation of an approximate electric field $\tilde{\mathbf{E}}$ from symmetry will be quantified by

$$R_E[\tilde{\mathbf{E}}] = \left\langle \frac{\|\tilde{\mathbf{E}} - \tilde{\mathbf{E}}'\|_2}{\|\mathbf{E}\|_2} \right\rangle_\Omega, \quad (12)$$

where $\tilde{\mathbf{E}}'$ is the transformed electric field

$$\tilde{E}_x'(x, y) = \tilde{E}_x(-x, y), \quad (13)$$

$$\tilde{E}_y'(x, y) = -\tilde{E}_y(-x, y). \quad (14)$$

In analogy with $\delta_E[\tilde{\mathbf{E}}]$, this is the mean pointwise relative deviation from symmetry of the electric field.

These metrics of deviation from symmetry are closely connected to the relative error metrics of Sec. III A. Specifically, the triangle inequality implies that

$$\sqrt{\langle (\tilde{u} - \tilde{u}')^2 \rangle_\Omega} \leq \sqrt{\langle (\tilde{u} - u)^2 \rangle_\Omega} + \sqrt{\langle (u - \tilde{u}')^2 \rangle_\Omega}. \quad (15)$$

By definition, the true solution u is invariant under the transformation that maps \tilde{u} to \tilde{u}' . Specifically,

$$\tilde{u}(x, y) - u(x, y) = -\tilde{u}'(-x, y) - (-u(-x, y)). \quad (16)$$

By the symmetry of the domain, it follows that

$$\sqrt{\langle (\tilde{u} - u)^2 \rangle_\Omega} = \sqrt{\langle (u - \tilde{u}')^2 \rangle_\Omega}. \quad (17)$$

Combining these results and dividing by $\sqrt{\langle u^2 \rangle_\Omega}$, it follows that

$$R_u[\tilde{u}] \leq 2\delta_u[\tilde{u}], \quad (18)$$

that is, the distance from an approximate potential \tilde{u} to its reflection \tilde{u}' is, at most, twice the distance from \tilde{u} to the true solution u . Very similar reasoning can be applied to an approximate electric field $\tilde{\mathbf{E}}$ to conclude that

$$R_E[\tilde{\mathbf{E}}] \leq 2\delta_E[\tilde{\mathbf{E}}]. \quad (19)$$

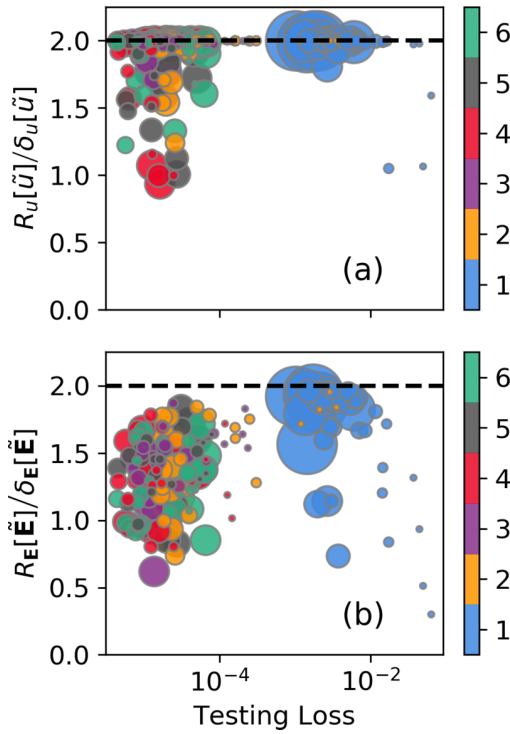


FIG. 6. Relative deviation of symmetry for the NNM solutions normalized by relative error, shown against testing loss. (a) Deviation of symmetry of the NNM electric potentials $R_u[\tilde{u}]$, divided by the relative error $\delta_u[\tilde{u}]$. (b) Deviation of symmetry of the NNM electric fields $R_E[\tilde{E}]$, divided by the relative error $\delta_E[\tilde{E}]$. Marker color indicates the depth of the network, and marker area indicates its width. The dotted lines show the upper bounds given by Eqs. (18) and (19).

Thus, solutions with small error values will inevitably be nearly symmetric, simply by virtue of being nearly equal to a symmetric function. Furthermore, since it was established in Sec. III A that the loss functional provides a reliable estimator of the error, it follows that the loss also provides a reliable estimator of the deviation from symmetry. It remains to be seen, however, whether or not inequalities (18) and (19) are strict in practice. That is, do neural networks learn that symmetry is a desirable feature, or are they only symmetric insofar as they approximate the true solution?

Figure 6 shows $R_u[\tilde{u}]/\delta_u[\tilde{u}]$ and $R_E[\tilde{E}]/\delta_E[\tilde{E}]$ for all networks in the data set, plotted against each network's testing loss. As in Fig. 5, the marker sizes correspond to network widths, and the colors indicate network depth. The dotted lines correspond to the maximum deviation from symmetry permitted for a given error value, according to inequalities (18) and (19).

Most of the data in Fig. 6(a) lie nearly on the dotted line: roughly 90% lie above 1.5, and 75% lie above 1.9. This indicates that most of the electric potentials approximated via the NNM satisfy the target symmetries only to the smallest degree required by virtue of their proximity to the true solution. The data in Fig. 6(b), however, lie somewhat farther from the dotted line. Quite a few of the most symmetric electric field approximations have $R_E[\tilde{E}]/\delta_E[\tilde{E}]$ ratios below 1, indicating

that they are more similar to their own reflections than they are to the true solution. It is important to note, however, that the electric field metrics of error and symmetry are normalized pointwise by the electric field intensity, whereas the electric potential metrics are not normalized pointwise. This distinction may account for some of the apparent differences between Figs. 6(a) and 6(b).

Altogether, the results in this section indicate that the NNM solutions deviate from the symmetries of the true solution by an amount comparable to their error values. Some networks may produce electric field solutions that are more symmetric than required given their error values alone, but most networks only exhibit the minimal degree of symmetry required by the triangle inequality. As discussed in the Introduction, directly constraining the networks to satisfy the symmetries (e.g., by modifying the network architectures, or by adding additional terms to the loss functional) would almost certainly improve the symmetry of the resulting approximations. However, implementing such constraints can be expensive for more complicated invariants, and some problems may exhibit invariants that are unknown *a priori*. These results illustrate that the NNM can still learn to satisfy invariants approximately, even when they are not explicitly enforced. Furthermore, the loss functional may provide a means of empirically estimating the extent to which such invariants are satisfied in practice.

2. Conservation of flux

Another important physical property of the true solution to the target PDE is the conservation of electric flux. In its strong form, conservation states that the true electric field \mathbf{E} must be divergence free at all points in the domain. This is equivalent to the condition that the true electric potential u must satisfy Laplace's equation $\nabla^2 u = 0$ since it can be rewritten as

$$\nabla \cdot (\nabla u) = \nabla \cdot \mathbf{E} = 0. \quad (20)$$

Thus, one could quantify the deviation from conservation of flux of an approximate field $\tilde{\mathbf{E}}$ by computing some error norm of $\nabla \cdot \tilde{\mathbf{E}}$. However, since all the derivatives taken in the NNM are exact (obtained via automatic differentiation), $\nabla \cdot \tilde{\mathbf{E}}$ is exactly equal to $\nabla^2 \tilde{u}$. As a result, the first term of the loss functional [Eq. (1)] is precisely a measure of how well the NNM satisfies the strong form of the conservation of flux.

Nonetheless, the strong form of conservation is insufficient to fully describe the extent to which the electric field conserves flux over extended regions of space within the domain. This is better described using the weak form, which states that the surface integral of the flux into any closed subset of the domain must be zero. Motivated by this, we define the quantity

$$\mathcal{E}(\tilde{u}; \epsilon) = \frac{1}{|\Omega^\epsilon|} \int_{\Omega^\epsilon} \left[\frac{1}{|B_\epsilon|} \int_{\partial B(\mathbf{x}; \epsilon)} \tilde{E}_n ds \right]^2 dA. \quad (21)$$

Here, $B(\mathbf{x}; \epsilon)$ is a ball of radius ϵ centered at a point \mathbf{x} in the domain, $\partial B(\mathbf{x}; \epsilon)$ denotes its boundary, and \tilde{E}_n denotes the outward normal component of the electric field into its surface. The outer integral is taken over Ω^ϵ , by which we denote the set of all points in the domain that are at least a distance ϵ from the boundary. The factors $|\Omega^\epsilon|$ and $|B_\epsilon|$ are the areas of Ω^ϵ and $B(\mathbf{x}; \epsilon)$, respectively. In other words, $\mathcal{E}(\tilde{u}; \epsilon)$ is

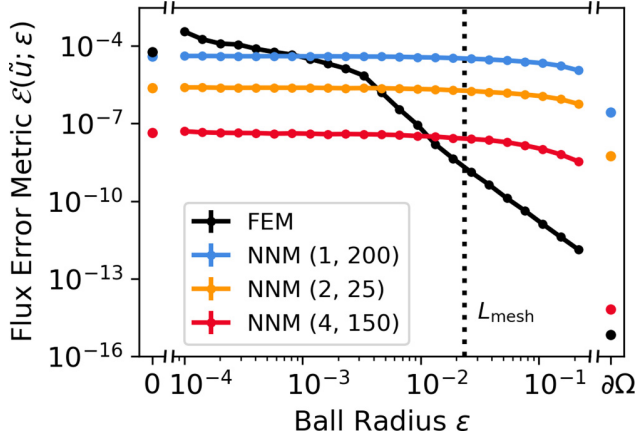


FIG. 7. The flux error metric $\mathcal{E}(\tilde{u}; \epsilon)$ plotted as a function of the ball radius ϵ for three NNM solutions as well as the reference FEM solution. The legend entries for the NNM solutions indicate the architecture (d, w) for each case. The leftmost points show $\mathcal{E}(\tilde{u}; 0)$, and the rightmost show $\mathcal{E}(\tilde{u}; \partial\Omega)$. The dotted vertical line labeled L_{mesh} indicates the mean length scale of the FEM mesh.

the mean-square norm of the flux into all balls of radius ϵ that are entirely contained within Ω , divided by the area of those balls. Because this definition of $\mathcal{E}(\tilde{u}; \epsilon)$ is mesh agnostic, it can also be computed directly for a FEM solution. Numerical calculations of $\mathcal{E}(\tilde{u}; \epsilon)$ and related metrics in this section are somewhat technical, and details are relegated to Appendix B.

Figure 7 shows $\mathcal{E}(\tilde{u}; \epsilon)$ computed for a sample of NNM solutions (colored lines) as well as for the reference FEM solution (black line). The architectures, losses, and relative errors of the three networks shown in Fig. 7 are listed in Table I. The shape of $\mathcal{E}(\tilde{u}; \epsilon)$ measured for the NNM solutions in Fig. 7 is representative of what was measured on several other NNM solutions (not included). In particular, $\mathcal{E}(\tilde{u}; \epsilon)$ was consistently observed to decrease monotonically with increasing ϵ . In Fig. 7, the network with architecture $(d, w) = (2, 25)$ achieved relatively mediocre performance. The $(1, 200)$ network performed fairly poorly overall, but was still among the best performing shallow networks in the data set. As expected, the best of the three networks according to testing loss and the relative error metrics, $(4, 150)$, also performed best in terms of conservation of flux. Similarly, $(2, 25)$ outperformed $(1, 200)$. We emphasize that the $(2, 25)$ network outperforms the $(1, 200)$ network in all metrics, despite having slightly *smaller* capacity. This is reflective of the disproportionately poor performance of shallow architectures noted in Secs. III A and III D.

TABLE I. Summary of the NNM solutions selected for the conservation of flux and particle simulations tests. Columns shown the depth, width, capacity, testing loss, and relative error of the electric potential and electric field, for each network.

d	w	Capacity	$\mathcal{L}[\tilde{u}]$	$\delta_u[\tilde{u}]$	$\delta_E[\tilde{\mathbf{E}}]$
1	200	801	3×10^{-3}	16%	7.4%
2	25	751	2×10^{-4}	1.7%	0.8%
4	150	68551	6×10^{-6}	0.02%	0.08%

The behavior of $\mathcal{E}(\tilde{u}; \epsilon)$ for the FEM solution differs from that of the NNM solutions in some important ways. Whereas, for all three NNM solutions, $\mathcal{E}(\tilde{u}; \epsilon)$ is roughly constant below $\epsilon \approx 10^{-1}$, for the FEM solution $\mathcal{E}(\tilde{u}; \epsilon)$ continues to increase with decreasing ϵ until at least $\epsilon \approx 10^{-4}$. As a result, although the FEM solution achieves better $\mathcal{E}(\tilde{u}; \epsilon)$ than all NNM solutions at long length scales, the converse is true at sufficiently small length scales. The best NNM solution in Fig. 7, $(4, 150)$, exhibits comparable conservation of flux to the FEM solution at length scales near the mean FEM mesh size $L_{\text{mesh}} = \sqrt{|\Omega|/N}$, where N is the number of mesh elements. At length scales below L_{mesh} , the $(4, 150)$ network conserves flux more accurately than the FEM solution. Even the worst of the three NNM solutions shown in Fig. 7 performs comparably to the FEM solution in conservation of flux at length scales below $\epsilon \approx 10^{-3}$. The relative stability of the NNM at small length scales may be attributable to its mesh-free nature, and is an appealing feature for subsequent use in particle simulations. Finally, we recall (see Sec. II C) that the FEM solution was computed in double precision, and suggest that the single precision used for the NNM solutions may be a limiting factor to their performance at large length scales.

For small choices of ϵ , $\mathcal{E}(\tilde{u}; \epsilon)$ converges to a measure of the strong form of conservation of flux. By the divergence theorem, for a continuously differentiable field $\tilde{\mathbf{E}}$, the flux error metric $\mathcal{E}(\tilde{u}; \epsilon)$ can be rewritten as

$$\mathcal{E}(\tilde{u}; \epsilon) = \frac{1}{|\Omega^\epsilon|} \int_{\Omega^\epsilon} \left[\frac{1}{|B_\epsilon|} \int_{B(\mathbf{x}; \epsilon)} \nabla \cdot \tilde{\mathbf{E}} dA' \right]^2 dA \quad (22)$$

$$= \text{mean}_{\Omega^\epsilon} \left[\left(\text{mean}_{B(\mathbf{x}; \epsilon)} (\nabla \cdot \tilde{\mathbf{E}}) \right)^2 \right]. \quad (23)$$

In the remainder of this section, angle brackets $\langle \cdot \rangle_S$ will be used to denote means over any set S . From Eq. (22), it is easy to deduce the limit of $\mathcal{E}(\tilde{u}; \epsilon)$ as $\epsilon \rightarrow 0$, which will be denoted $\mathcal{E}(\tilde{u}; 0)$. Since $\Omega^\epsilon \rightarrow \Omega$ and the mean over $B(\mathbf{x}; \epsilon)$ approaches the identity operator, it follows that

$$\mathcal{E}(\tilde{u}; 0) = \langle (\nabla \cdot \tilde{\mathbf{E}})^2 \rangle_\Omega = \langle (\nabla^2 \tilde{u})^2 \rangle_\Omega. \quad (24)$$

The leftmost points in Fig. 7 illustrate $\mathcal{E}(\tilde{u}; 0)$ for each of the solutions. For the NNM solutions, $\mathcal{E}(\tilde{u}; \epsilon)$ converges to $\mathcal{E}(\tilde{u}; 0)$ as $\epsilon \rightarrow 0$, as expected. This is not the case for the FEM solution, for which $\mathcal{E}(\tilde{u}; \epsilon)$ exceeds $\mathcal{E}(\tilde{u}; 0)$ for small ϵ . However, this is not a contradiction, as Eq. (24) was derived by assuming continuous differentiability.

Equation (24) is precisely the mean of the square deviation of \tilde{u} from the strong form of conservation of flux. For NNM solutions, $\mathcal{E}(\tilde{u}; 0)$ is equal to the first term of the loss functional [Eq. (1)] divided by $|\Omega|$, and is therefore bounded above by the loss. Given that $\mathcal{E}(\tilde{u}; \epsilon)$ was observed to decrease monotonically with ϵ , this suggests that, as for the relative errors and symmetry errors, the loss provides a useful estimator of the error in conservation of flux over any length scale.

However, as ϵ increases, the metric $\mathcal{E}(\tilde{u}; \epsilon)$ becomes increasingly biased because the center of the balls $B(\mathbf{x}; \epsilon)$ cannot be placed within a distance ϵ of the boundaries of the domain. At moderate values of ϵ , this means that errors in flux conservation in the interior of the domain are weighted

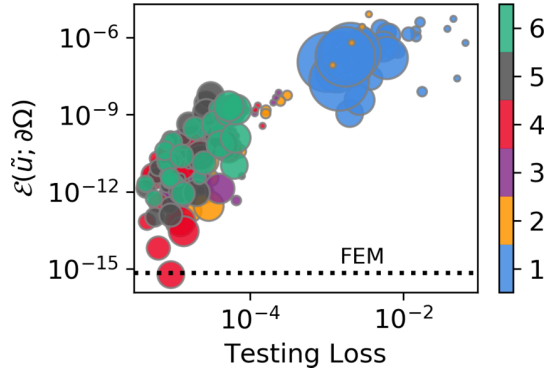


FIG. 8. Error in global flux conservation for all NNM solutions as a function of each network's testing loss. Marker color indicates the depth of the network, and marker area indicates its width. The dotted line indicates the corresponding error in the FEM solution.

more heavily than those near the boundaries of the domain. Eventually, when $\epsilon > 0.6$, the balls are too large to fit inside the slits of the device, so that only errors inside the well contribute to $\mathcal{E}(\tilde{u}; \epsilon)$. For this reason, the data in Fig. 7 are only computed for ϵ values sufficiently below 0.6 that this bias is deemed acceptably small. This biased behavior of $\mathcal{E}(\tilde{u}; \epsilon)$ arises because the inner integral in Eq. (22) is based on circle-shaped test sets. A more meaningful metric of flux conservation over very long length scales can be obtained by replacing $B(\mathbf{x}; \epsilon)$ with $\partial\Omega$ in Eq. (22). This global flux error will be denoted $\mathcal{E}(\tilde{u}; \partial\Omega)$, and satisfies

$$\mathcal{E}(\tilde{u}; \partial\Omega) = \left[\frac{|\partial\Omega|}{|\Omega|} \langle \tilde{E}_n \rangle_{\partial\Omega} \right]^2 = [\langle \nabla^2 \tilde{u} \rangle_{\Omega}]^2. \quad (25)$$

Thus, $\mathcal{E}(\tilde{u}; \partial\Omega)$ is directly connected to $\langle \tilde{E}_n \rangle_{\partial\Omega}$, the net flux through $\partial\Omega$, which is zero for the true solution. Note that the second equality in Eq. (25) follows from the divergence theorem, so it applies to the NNM solutions but not the FEM solution. Together with the second equality of Eq. (24), this means

$$\mathcal{E}(\tilde{u}; 0) - \mathcal{E}(\tilde{u}; \partial\Omega) = \left\langle (\nabla^2 \tilde{u})^2 \right\rangle_{\Omega} - [\langle \nabla^2 \tilde{u} \rangle_{\Omega}]^2, \quad (26)$$

which is the variance of $\nabla^2 u$ over Ω . This is always non-negative, so it follows that

$$\mathcal{E}(\tilde{u}; 0) \geq \mathcal{E}(\tilde{u}; \partial\Omega), \quad (27)$$

for any \tilde{u} satisfying the second inequalities in both Eqs. (24) and (25).

The rightmost points in Fig. 7 illustrate $\mathcal{E}(\tilde{u}; \partial\Omega)$ for each of the four solutions. Figure 8 shows $\mathcal{E}(\tilde{u}; \partial\Omega)$ for all NNM solutions versus each network's testing loss; the dotted line indicates the value for the FEM solution. It is immediately evident that $\mathcal{E}(\tilde{u}; \partial\Omega)$ relates to testing loss in a similar way as do the relative error metrics (Fig. 5). As was the case for the other metrics, $\mathcal{E}(\tilde{u}; \partial\Omega)$ decreases with decreasing testing loss, suggesting that testing loss is a useful estimator of global flux error. Indeed, this is inevitable in the limit of small loss since $\mathcal{E}(\tilde{u}; \partial\Omega)$ is bounded above by $\mathcal{E}(\tilde{u}; 0)$, which is in turn bounded above by the loss. It also appears that the data in Fig. 8 are divided into the same two clusters as the data in

Fig. 5, with the shallow architectures performing worse than nearly all deep architectures.

Somewhat surprisingly, the best of the NNM solutions appear to conserve flux globally to nearly the same degree as the reference FEM solution, despite being computed in single (rather than double) precision. Indeed, one network with architecture (4,200) appears to slightly outperform the FEM solution in this respect. However, it is important to note that $\mathcal{E}(\tilde{u}; \epsilon)$ for this (4,200) network (not shown) exhibits essentially the same behavior as that of the (4,150) network analyzed in Fig. 7. In other words, although that particular network performs very well at global flux conservation, FEM does a significantly better job at conserving flux over intermediate length scales. This suggests that, for the NNM solutions, the error in conservation of flux is heterogeneously distributed throughout the domain, which is consistent with the previous observation that error in the NNM solutions is significantly larger near the reentrant corners.

In summary, the metric $\mathcal{E}(\tilde{u}; \epsilon)$ provides a mesh-agnostic measure of how well an NNM solution conserves flux over a length scale ϵ . As $\epsilon \rightarrow 0$, the limit satisfies Eq. (24), and is bounded above by the loss. Empirically, $\mathcal{E}(\tilde{u}; \epsilon)$ is observed to decrease monotonically with ϵ , so that the loss provides a useful estimator of the error in flux conservation over intermediate length scales, too. Alas, when ϵ is large relative to other length scales in the domain, $\mathcal{E}(\tilde{u}; \epsilon)$ is a biased metric, as it places less weight on flux lost near the boundaries of the domain. However, a related measure of global conservation of flux over the entire domain is given by Eq. (25), which is not biased. This measure, too, is bounded above by the loss. Altogether, the NNM seems capable of reliably producing solutions that conserve flux to an acceptable level of accuracy without the need to explicitly enforce this physical invariant during training. In particular, some of the NNM solutions conserve flux globally roughly as well as the FEM solution. Furthermore, even relatively mediocre NNM solutions conserve flux better than the FEM solution over sufficiently small length scales.

C. Application to particle simulations

Section III A looked directly at error between NNM and FEM, and Sec. III B looked at error metrics motivated by physical invariants. Both suggested that the testing loss provides a reliable estimator of the true performance of the network solutions, and that (with appropriate network architectures) the NNM consistently finds solutions with seemingly small error values. However, the question of what error values are acceptable is subjective, and often depends on the intended application of the numerical solutions. For this reason, this section will consider the performance of the NNM solutions when used as the driving force fields in particle simulations of Brownian motion in the slit-well device (implemented in the C programming language). The simulation scenario is quite similar to those investigated by [38,39].

Simulations of $N = 100\,000$ particles in the slit-well domain were initialized with all the particles located in the middle of the same well. The particle positions \mathbf{x}_i evolved

according to the discretized Brownian equation

$$\frac{\Delta \mathbf{x}_i}{\Delta t} = \sqrt{\frac{2D}{\Delta t}} \mathbf{R}(t) + \frac{q}{\gamma} \tilde{\mathbf{E}}, \quad (28)$$

where the time step was set to $\Delta t = 10^{-4}$, the diffusion coefficient to $D = 1$, and the friction coefficient to $\gamma = 1$. The particle charge q was varied from 1 to 10. The term $\mathbf{R}(t)$ is a random driving force, representing thermal motion of an implicit solvent, and was sampled via the Box-Muller transform from an independent standard Gaussian distribution for each particle at each time step.

The driving electric field $\tilde{\mathbf{E}}$ was obtained from either the reference FEM solution or from one of the NNM solutions. The electric fields were discretized onto a uniform square mesh overlain on $[-L_x, L_x] \times [-L_y, L_y]$, the smallest bounding box containing Ω (see Sec. II B). The side lengths of the mesh elements were set to 0.01. The field experienced by a particle at a given position was approximated by nearest-neighbor interpolation to the mesh. We leave more sophisticated coupling between the particle simulations and the electric fields to future work.

Particles experienced periodic boundary conditions across the left and right sides of the periodic subunit illustrated in Fig. 3, and the boundaries that were insulating in the electric field problem were treated as reflective in the particle simulations. The number of times each particle crossed the domain was tracked, so as to measure its absolute displacement from the original position. After $t_{\max} = 10^6$ time steps, the mean horizontal displacement of the particles from the initial position $\langle x \rangle$ was divided by t_{\max} to obtain an estimate $\langle v_x \rangle$ of the average particle velocity. This average velocity was then divided by particle charge to estimate the effective particle mobility $\mu = \langle v_x \rangle / q$. The statistical error on this mobility measurement was estimated as $s = (\sigma_{v_x} / q) / \sqrt{N}$, where σ_{v_x} is the standard deviation of the particle velocities.

These mobility measurements are shown in Fig. 9(a) for simulations conducted with the same four electric fields investigated in Sec. III B 2: that of the reference FEM solution, and that of the three NNM solutions summarized in Table I. The simulations using the FEM field were conducted twice with different random seeds, shown as the two black lines in Fig. 9(a). The difference between these two sets of measurements provides a means of distinguishing the errors introduced by the electric fields from simple statistical fluctuations on the mobility measurements. In Fig. 9(a), the measurements of μ made using the networks of architectures (2,25) and (4,150) appear fairly similar to those made using the FEM field. Conversely, the measurements using the (1,200) architecture are quite easily distinguished from the FEM data. All simulations recovered effective mobilities that varied with q , induced in the otherwise free-draining particles by their interactions with the slit-well geometry.

The relative error between two mobility measurements μ_1 and μ_2 was quantified as

$$\frac{\mu_1 - \mu_2}{\mu_2}. \quad (29)$$

The colored lines in Fig. 9(b) show the relative errors of the NNM-based mobility measurements in Fig. 9(a) versus

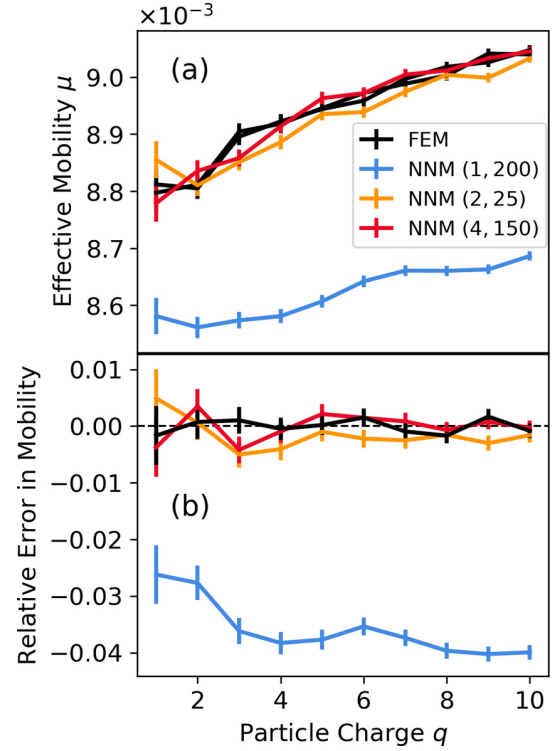


FIG. 9. (a) Lines show the mobility measurements μ made using four different electric field solutions. The two black lines correspond to separate simulations made using the same reference FEM field. The error bars indicate the estimated statistical error of mobility s . (b) The colored lines show the relative errors between the NNM-based measurements and the first set of FEM-based results. The black line shows the relative errors between the two sets of FEM-based measurements. The error bars are obtained from the data in (a) via standard rules for propagation of uncertainty.

the first set of FEM-based measurements. The black line corresponds to the relative errors between the two sets of FEM-based measurements. Error bars were estimated via standard rules for propagation of error.

Unsurprisingly, the errors of the (1,200) architecture are significantly larger than those of the other two architectures, and show a clear bias toward underestimating the mobility. Nonetheless, even this crude solution produces errors smaller in magnitude than 5% of the actual mobility. This suggests that the current particle simulations are relatively insensitive to moderate inaccuracies in the driving electric field.

The relative errors of both the (2,25) and (4,150) architectures are comparable to the relative errors between the two sets of FEM-based measurements, and lie below 1% for all values of q . However, the relative errors for the (2,25) architecture are negative for all q above 2, whereas the relative errors of the (4,150) architecture are roughly evenly distributed about 0. This suggests that the (2,25) architecture introduces a small but detectable systematic bias into the mobility measurements. Conversely, the errors of the better (4,150) architecture are comparable to statistical fluctuations, despite the relatively large number of simulated particles, $N = 100\,000$. These results confirm that the best of the NNM solutions presented in this work are sufficiently accurate for

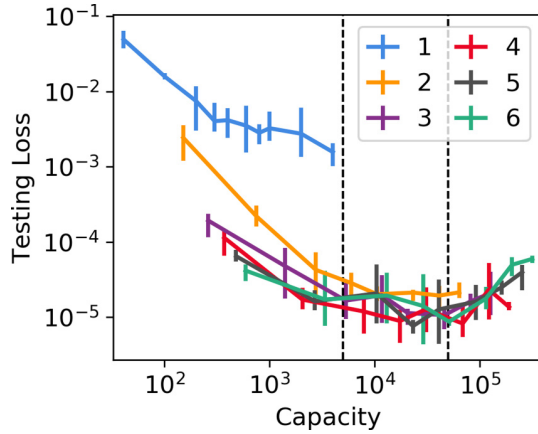


FIG. 10. Testing loss versus network capacity, colored by network depths. The error bars show maxima and minima over four random seeds, and the lines indicate mean performance. The dotted lines at capacities of 5×10^3 and 5×10^4 roughly delineate the three regimes discussed in the text.

use in particle simulation applications. Moreover, the relative performance of the three architectures is consistent with their values of $\mathcal{L}[\tilde{u}]$, $\delta_u[\tilde{u}]$, and $\delta_E[\tilde{E}]$ (Table I).

In Fig. 9, the network with architecture (2,25) significantly outperforms that with architecture (1,200), despite having slightly smaller capacity, reemphasizing the advantages of deep architectures over shallow ones. Conversely, the much larger (4,150) architecture only achieves moderate improvements over the (2,25) architecture, reflecting the diminishing returns associated with increasing network capacity. These subtle impacts of architecture are investigated more closely in Sec. III D.

D. Effect of network architecture

The previous sections have demonstrated that the testing loss is a useful estimator of several independent error metrics. Specifically, the loss functional appears to reliably estimate the error relative to the reference FEM solution; the deviation from symmetry; the deviation from conservation of flux; and the error introduced into subsequent mobility measurements. Thus, the loss is a useful single metric of performance via which to compare different NNM architectures.

In Fig. 10, the testing loss is plotted against the total network capacity. Here, network capacity is measured as the total number of parameters in the network, given in terms of the width w and depth d by

$$(2+1)w + (d-1)(w+1)w + (w+1) \quad (30)$$

since the networks have two inputs and one output. The colored lines in Fig. 10 correspond to different network depths, so that the various capacities within each line identify the network widths. The error bars show maxima and minima over all random seeds, whereas the lines indicate mean performance.

The data in Fig. 10 show that, for network capacities below 5×10^3 , increasing capacity improves testing loss for any choice of depth. This suggests that, for those networks, insufficient capacity is a primary bottleneck toward representing more accurate approximations of the true solution. In

particular, for the networks with two hidden layers, increasing the capacity improves the loss by nearly two orders of magnitude. Furthermore, in this low-capacity regime, increasing depth improves performance for a given capacity. In other words, when insufficient network capacity is the primary barrier to improved performance, deeper networks make more efficient use of that limited resource. Indeed, this is consistent with the effects of architecture observed in Figs. 5, 7, 8, and 9. Specifically, shallow networks perform particularly poorly in all metrics throughout this work, even compared to networks with comparable capacity and as few as two hidden layers.

For deep networks with moderately large capacities (5×10^3 to 5×10^4), testing loss is essentially independent of network architecture (i.e., independent of both depth and capacity/width). This suggests that insufficient network capacity is no longer a primary bottleneck to improving solution accuracy. The investigation by [20] suggested that the internal representations learned by networks in the NNM become essentially independent of width above some critical size, so it is not surprising that loss similarly becomes independent of width. However, it is noteworthy that this limiting loss value is also independent of network depth (among those with two or more hidden layers).

For networks with capacities of 5×10^4 or above, testing loss begins to increase with further increases in capacity. Figure 5 illustrates that these same networks sometimes exhibit relative errors nearly as high as some shallow networks, despite having two orders of magnitude more capacity. Their poor performance can be understood in terms of the difficulties commonly encountered in training very deep, wide neural networks. For instance, Berg and Nyström [16] noted similar loss in performance when training networks with five or more hidden layers, and attributed this to vanishing gradients. Refinements in the network architectures and training algorithms can be expected to alleviate this phenomenon.

Note that the behavior of these networks with very large capacities cannot be described in terms of overfitting, another problem commonly encountered by networks with excessively large capacities. Overfitting is typically defined as a significant gap between the training and testing losses of networks. In the NNM, however, the testing and training sets are drawn from identical distributions. In the implementation used here, in particular, the training set is redrawn regularly throughout training, so that it is fundamentally impossible for the network to be overfitting to a specific set of training samples.

Finally, Fig. 11 shows the total training time of the NNM solutions against testing loss. The same two populations identified in Figs. 5 and 8 are evident again in Fig. 11. The cluster on the right contains all the shallow networks as well as the narrowest of the deep ones. The cluster on the left consists of those networks that attained better than 1% error relative to FEM (Fig. 5). Within each cluster, testing loss and training time are loosely correlated. For all networks, training time was on the order of hours. However, it is important to note that the implementation in this work was not concerned with optimizing the computational efficiency of the NNM, but rather with ensuring that the training process was thoroughly converged (Sec. II B).

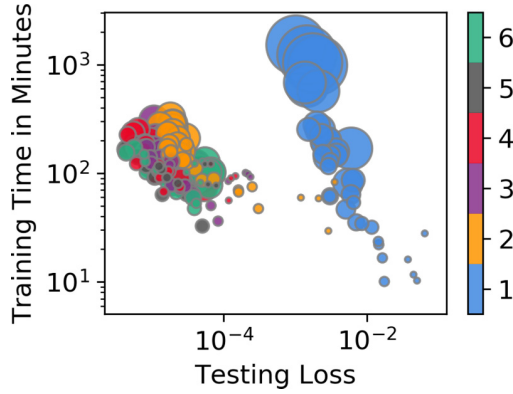


FIG. 11. Total training time and final testing loss of the NNM solutions. Marker color indicates the depth of the network, and marker area indicates its width.

Once again, the networks in the right cluster perform disproportionately poorly, even though many of them have capacities comparable to some of those in the left cluster (Fig. 10). Thus, not only do the networks in the left cluster achieve better accuracies (as measured by testing loss or any of the various error metrics in this paper), but they also finish training far more rapidly. Further, this conclusion is true even between networks of equal capacity. These observations demonstrate many benefits of using deeper architectures in the NNM, and several disadvantages of using shallow architectures.

IV. CONCLUSIONS

This work investigated the performance of the neural network method (NNM) when used to solve the electric potential and field in the slit-well device. This problem features a non-convex geometry, which makes it particularly challenging to solve with the NNM. Performance was quantified in multiple metrics, and compared against a reference FEM solution.

The best network architectures studied here reliably achieved relative errors below 0.1% in both the potential and the field. NNM solutions also recovered spatial symmetries of the true solution to roughly the same extent that they approximated the true solution. Regarding conservation of flux, the NNM solutions performed comparably to the reference FEM solution. Finally, particle simulations conducted using the NNM electric fields yielded mobility measurements consistent with those based on the FEM electric field. In each of these metrics, the testing loss was found to provide a useful estimator of the networks' true performance. That is, networks with smaller losses were found to be closer to the true solution; to more closely approximate the target symmetries; to conserve flux more accurately; and to produce better particle simulations.

These empirical investigations uncovered several valuable insights for practical use of the NNM. Accurate solutions to physical problems can be obtained even without explicitly enforcing known physical invariants of the true problem. The importance of architecture was reemphasized: deep architectures consistently outperformed shallow ones, converging to better solutions in less time and using fewer degrees of

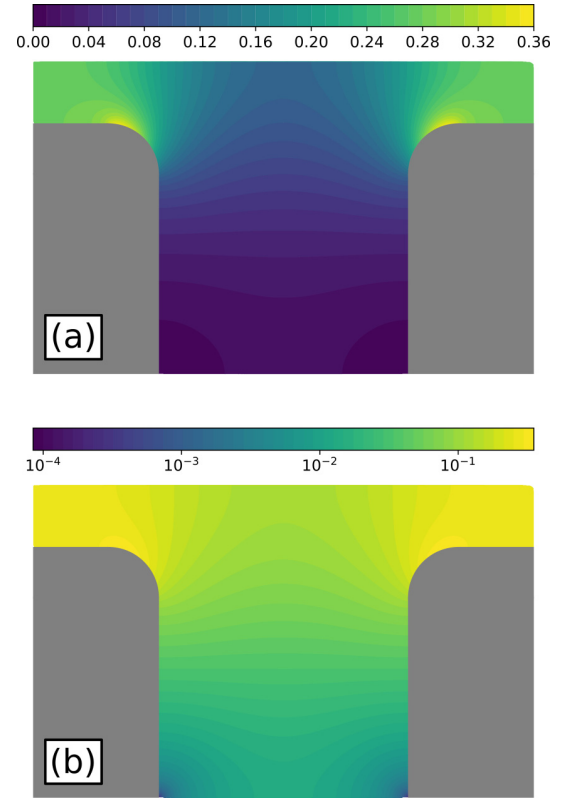


FIG. 12. Electric field intensity of the FEM solution, shown on (a) linear and (b) logarithmic color scales.

freedom. Finally, the testing loss may provide a practical means of gauging a solution's accuracy, even when the ground truth is unknown and convergence is not theoretically guaranteed.

In summary, this work demonstrates that the NNM can successfully solve a problem that is ill conditioned due to the nonconvexity of its domain. The NNM solutions were found to be particularly appropriate for use in subsequent particle simulations. This suggests that it could be a useful tool for the study of microfluidic and nanofluidic devices (MNFDs) and other biophysical systems. Moreover, differential equations in domains with complicated geometries arise throughout physics and other fields. These results support the feasibility of using the NNM to solve this fundamental and ubiquitous class of problems.

ACKNOWLEDGMENT

H.W.d.H. gratefully acknowledges funding from the Natural Sciences and Engineering Research Council (NSERC) in the form of Discovery Grant No. 2014-06091.

APPENDIX A: ADDITIONAL PLOTS OF THE ELECTRIC FIELD SOLUTION

Figure 12 shows the FEM electric field intensity throughout the domain, in both linear and logarithmic color scales. In particular, Fig. 12 illustrates that the peak field intensity occurs near the reentrant corners, with a magnitude of about

0.36. In the bottom corners of the well, the field intensity is over four orders of magnitude weaker. These features contribute to the difficulty of applying the NNM to the slit-well electric field problem since the standard loss functional used during training places equal weight on all regions of Ω and $\partial\Omega$. The regions of very intense electric field near the reentrant corners, specifically, seem to be most difficult to resolve for the NNM, as seen in the error maps shown in Fig. 4.

APPENDIX B: DETAILS OF FLUX LOSS CALCULATIONS

This Appendix contains descriptions of how the metrics shown in Figs. 7 and 8 were computed. For Fig. 7, the integrals in Eq. (21) were computed by sampling 10 000 uniformly spaced points on $\partial B(\mathbf{x}; \epsilon)$ for each choice of the center \mathbf{x} . Candidate samples for the centers were generated according

to the same procedure described in Sec. II B, but with 10 times higher sample density, and all points within a distance ϵ of $\partial\Omega$ were rejected.

The leftmost points in Fig. 7 correspond to Eq. (24). For the NNM solutions, these were computed by Monte Carlo integration over Ω using 10 times higher sampling density than in Sec. II B. The rightmost points in Fig. 7 correspond to Eq. (25). These were not computed using a Monte Carlo integration approach. Because $\langle \tilde{E}_h \rangle_{\partial\Omega}$ is a small number computed by summing many positive and negative terms, it is vulnerable to catastrophic cancellation. For this reason, it was computed using a uniform mesh of points along $\partial\Omega$, sampled with 100 times higher density than in Sec. II B. For the FEM solution, the integrals required for Eqs. (24) and (25) were both computed in FENICS using Gaussian quadrature via the `assemble` command.

- [1] R. D. Cook, M. E. Plesha, D. S. Malkus, and R. J. Witt, *Concepts and Applications of Finite Element Analysis* (Wiley, Hoboken, NJ, 2007).
- [2] M. W. M. G. Dissanayake and N. Phan-Thien, Neural-network-based approximations for solving partial differential equations, *Commun. Numer. Methods Eng.* **10**, 195 (1994).
- [3] J. Han and H. G. Craighead, Entropic trapping and sieving of long DNA molecules in a nanofluidic channel, *J. Vac. Sci. Technol. A* **17**, 2142 (1999).
- [4] J. Han and H. G. Craighead, Separation of long DNA molecules in a microfabricated entropic trap array, *Science* **288**, 1026 (2000).
- [5] S. L. Levy and H. G. Craighead, DNA manipulation, sorting, and mapping in nanofluidic systems, *Chem. Soc. Rev.* **39**, 1133 (2010).
- [6] K. D. Dorfman, DNA electrophoresis in microfabricated devices, *Rev. Mod. Phys.* **82**, 2903 (2010).
- [7] A. J. Meade Jr. and A. A. Fernandez, The numerical solution of linear ordinary differential equations by feedforward neural networks, *Math. Comput. Modell.* **19**, 1 (1994).
- [8] A. J. Meade Jr. and A. A. Fernandez, Solution of nonlinear ordinary differential equations by feedforward neural networks, *Math. Comput. Modell.* **20**, 19 (1994).
- [9] B. Ph. van Milligen, V. Tribaldos, and J. A. Jiménez, Neural Network Differential Equation and Plasma Equilibrium Solver, *Phys. Rev. Lett.* **75**, 3594 (1995).
- [10] I. E. Lagaris, A. Likas, and D. I. Fotiadis, Artificial neural networks for solving ordinary and partial differential equations, *IEEE Trans. Neural Networks* **9**, 987 (1998).
- [11] N. Yadav, A. Yadav, and M. Kumar, *An Introduction to Neural Network Methods for Differential Equations* (Springer, Berlin, 2015).
- [12] I. E. Lagaris, A. Likas, and D. I. Fotiadis, Artificial neural network methods in quantum mechanics, *Comput. Phys. Commun.* **104**, 1 (1997).
- [13] W. E, J. Han, and A. Jentzen, Deep learning-based numerical methods for high-dimensional parabolic partial differential equations and backward stochastic differential equations, *Commun. Math. Stat.* **5**, 349 (2017).
- [14] W. E and B. Yu, The Deep Ritz Method: A deep learning-based numerical algorithm for solving variational problems, *Commun. Math. Stat.* **6**, 1 (2018).
- [15] V. I. Avrutskiy, Neural networks catching up with finite differences in solving partial differential equations in higher dimensions, *Neural Comput. Applic.* (2020), doi: [10.1007/s00521-020-04743-8](https://doi.org/10.1007/s00521-020-04743-8).
- [16] J. Berg and K. Nyström, A unified deep artificial neural network approach to partial differential equations in complex geometries, *Neurocomputing* **317**, 28 (2018).
- [17] J. Sirignano and K. Spiliopoulos, DGM: A deep learning algorithm for solving partial differential equations, *J. Comput. Phys.* **375**, 1339 (2018).
- [18] V. R. Royo and C. Tomlin, Recursive regression with neural networks: Approximating the HJI PDE solution, [arXiv:1611.02739](https://arxiv.org/abs/1611.02739).
- [19] J. Han, A. Jentzen, and Weinan E, Solving high-dimensional partial differential equations using deep learning, *Proc. Natl. Acad. Sci. USA* **115**, 8505 (2018).
- [20] M. Magill, F. Z. Qureshi, and H. W. de Haan, Neural networks trained to solve differential equations learn general representations, in *Advances in Neural Information Processing Systems* (MIT Press, Cambridge, MA, 2018), pp. 4071–4081.
- [21] C. Huré, H. Pham, and X. Warin, Deep backward schemes for high-dimensional nonlinear PDEs, *Math. Comput.* **89**, 1547 (2020).
- [22] S. Karumuri, R. Tripathy, I. Bilonis, and J. Panchal, Simulator-free solution of high-dimensional stochastic elliptic partial differential equations using deep neural networks, *J. Comput. Phys.* **404**, 109120 (2020).
- [23] H. Mutuk, Neural network study of hidden-charm pentaquark resonances, *Chin. Phys. C* **43**, 093103 (2019).
- [24] M. A. Nabian and H. Meidani, A deep learning solution approach for high-dimensional random differential equations, *Probab. Eng. Mech.* **57**, 14 (2019).
- [25] D. Zhang, L. Guo, and G. E. Karniadakis, Learning in modal space: Solving time-dependent stochastic PDEs using physics-informed neural networks, *SIAM J. Sci. Comput.* **42**, A639 (2020).

- [26] C. Beck, Weinan E, and A. Jentzen, Machine learning approximation algorithms for high-dimensional fully nonlinear partial differential equations and second-order backward stochastic differential equations, *J. Nonlinear Sci.* **29**, 1563 (2019).
- [27] Q. Wei, Y. Jiang, and J. Z. Y. Chen, Machine-learning solver for modified diffusion equations, *Phys. Rev. E* **98**, 053304 (2018).
- [28] P. Grohs, F. Hornung, A. Jentzen, and P. von Wurstemberger, A proof that artificial neural networks overcome the curse of dimensionality in the numerical approximation of Black-Scholes partial differential equations, [arXiv:1809.02362](https://arxiv.org/abs/1809.02362).
- [29] A. Jentzen, D. Salimova, and T. Welte, A proof that deep artificial neural networks overcome the curse of dimensionality in the numerical approximation of Kolmogorov partial differential equations with constant diffusion and nonlinear drift coefficients, [arXiv:1809.07321](https://arxiv.org/abs/1809.07321).
- [30] M. Hutzenthaler, A. Jentzen, T. Kruse, and T. A. Nguyen, A proof that rectified deep neural networks overcome the curse of dimensionality in the numerical approximation of semilinear heat equations, *SN Partial Differ. Equ. Appl.* **1**, 10 (2020).
- [31] K. S. McFall and J. R. Mahan, Artificial neural network method for solution of boundary value problems with exact satisfaction of arbitrary boundary conditions, *IEEE Trans. Neural Networks* **20**, 1221 (2009).
- [32] C. Michoski, M. Milosavljević, T. Oliver, and D. R. Hatch, Solving differential equations using deep neural networks, *Neurocomputing* **399**, 193 (2020).
- [33] P. Abgrall and N. T. Nguyen, Nanofluidic devices and their applications, *Anal. Chem.* **80**, 2326 (2008).
- [34] H. Gardeniers and A. V. Den Berg, Micro- and nanofluidic devices for environmental and biomedical applications, *Int. J. Environ. Anal. Chem.* **84**, 809 (2004).
- [35] R. Mulero, A. S. Prabhu, K. J. Freedman, and M. J. Kim, Nanopore-based devices for bioanalytical applications, *JALA: J. Assoc. Lab. Automation* **15**, 243 (2010).
- [36] J. Fu, P. Mao, and J. Han, Nanofilter array chip for fast gel-free biomolecule separation, *Appl. Phys. Lett.* **87**, 263902 (2005).
- [37] J. Fu, J. Yoo, and J. Han, Molecular Sieving in Periodic Free-Energy Landscapes Created by Patterned Nanofilter Arrays, *Phys. Rev. Lett.* **97**, 018103 (2006).
- [38] K.-L. Cheng, Y.-J. Sheng, S. Jiang, and H.-K. Tsao, Electrophoretic size separation of particles in a periodically constricted microchannel, *J. Chem. Phys.* **128**, 101101 (2008).
- [39] H. Wang, H. W. de Haan, and G. W. Slater, Electrophoretic ratcheting of spherical particles in well/channel microfluidic devices: Making particles move against the net field, *Electrophoresis* **41**, 621 (2020).
- [40] M. Langecker, D. Pedone, F. C. Simmel, and U. Rant, Electrophoretic time-of-flight measurements of single DNA molecules with two stacked nanopores, *Nano Lett.* **11**, 5002 (2011).
- [41] B. A. Finlayson, *The Method of Weighted Residuals and Variational Principles*, Vol. 73 (SIAM, Philadelphia, 2013).
- [42] C. Anitescu, E. Atroshchenko, N. Alajlan, and T. Rabczuk, Artificial neural network methods for the solution of second order boundary value problems, *CMC-Comput. Mater. Continua* **59**, 345 (2019).
- [43] A. Al-Arabi, A. Correia, D. Naiff, G. Jardim, and Y. Saporito, Solving nonlinear and high-dimensional partial differential equations via deep learning, [arXiv:1811.08782](https://arxiv.org/abs/1811.08782).
- [44] M. Mattheakis, P. Protopapas, D. Sondak, M. Di Giovanni, and E. Kaxiras, Physical symmetries embedded in neural networks, [arXiv:1904.08991](https://arxiv.org/abs/1904.08991).
- [45] J. Hermann, Z. Schätzle, and F. Noé, Deep neural network solution of the electronic Schrödinger equation, [arXiv:1909.08423](https://arxiv.org/abs/1909.08423).
- [46] F. Moukalled, L. Mangani, and M. Darwish, *The Finite Volume Method in Computational Fluid Dynamics* (Springer, Berlin, 2016), Vol. 113.
- [47] E. Hairer, G. Wanner, and C. Lubich, Symplectic integration of Hamiltonian systems, *Geometric Numerical Integration* (Springer, Berlin, 2006), pp. 179–236.
- [48] S. Zhang, Z. Zhang, and Q. Zou, A postprocessed flux conserving finite element solution, *Numer. Methods Partial Differ. Equ.* **33**, 1859 (2017).
- [49] Z. Cai and S. Kim, A finite element method using singular functions for the Poisson equation: corner singularities, *SIAM J. Numer. Anal.* **39**, 286 (2001).
- [50] M. Dauge, *Elliptic Boundary Value Problems on Corner Domains: Smoothness and Asymptotics of Solutions*, Vol. 1341 (Springer, Berlin, 2006).
- [51] P. Grisvard, *Elliptic Problems in Nonsmooth Domains* (SIAM, 1985).
- [52] D. P. Kingma and J. Ba, Adam: A method for stochastic optimization, [arXiv:1412.6980](https://arxiv.org/abs/1412.6980).
- [53] X. Glorot and Y. Bengio, Understanding the difficulty of training deep feedforward neural networks, in *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics, Chia Laguna Resort, Sardinia, Italy* (PMLR, 2010), Vol. 9, pp. 249–256.
- [54] M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G. S. Corrado, A. Davis, J. Dean, M. Devin *et al.*, TensorFlow: Large-scale machine learning on heterogeneous systems, 2015, <https://www.tensorflow.org/>
- [55] M. S. Alnæs, J. Blechta, J. Hake, A. Johansson, B. Kehlet, A. Logg, C. Richardson, J. Ring, M. E. Rognes, and G. N. Wells, The FEniCS project version 1.5, *Archive Numer. Software* **3**, 9 (2015).

Chapter 5

Neural networks trained to solve differential equations learn general representations

Whereas the study in Chapter 4 studied the NNM from the perspective of classical numerical method analysis, *Neural networks trained to solve differential equations learn general representations* is an effort to understand the NNM using analysis techniques developed by the deep learning research community. The ability of deep learning to learn meaningful and useful representations via its hidden layers is commonly theorized to be responsible for much of its prodigious success. Despite this, this perspective is rarely incorporated into analyses of the NNM using the tools of numerical analysis. Rather, those studies tend to use elaborate constructions to demonstrate that deep learning can, in principle, be used to emulate some well-studied approximation method (e.g., spline interpolation). These neural networks are not reflective of the neural networks that emerge in practice during NNM training, which may explain the

theory-to-practice gap between the astonishing theoretical results predicted for the NNM and its comparatively modest performance in practice (Sec. [A.3.3](#)).

In particular, as seen in Chapter [4](#), the accuracy of the NNM does not appear to converge at the expected rate with respect to the number of degrees of freedom available in the trial function. The study in this chapter may provide some resolution to this fact. Essentially, it appears that each hidden layer in the deep neural network trial function converges to a fixed representation as width is increased. Additional degrees of freedom in that layer have negligible impact on the output of the neural network. The analysis additionally suggests that these learned features may depend continuously on the target function being learned, and even that they may have meaningful interpretations in terms of the geometry of the problem domain. The manuscript is included in Sec. [5.3](#).

5.1 Motivation

Originally, this work was not meant as an investigation of the NNM at all. Rather, the intention was to investigate the internal structure of deep neural networks, with the goal of understanding why they are able to achieve such impressive results on so many applications. The task of solving differential equations was selected as a test problem on which to study the dynamics of deep learning. In comparison to the fairly ill-defined tasks common in machine vision and natural language processing, differential equations are extremely precise mathematical problems. Nonetheless, partial differential equations are far from trivial toy models, and the complexity of the functions they describe can easily be increased. The conjecture was that analysing deep neural networks would be

more fruitful in this setting, as extensive mathematical theory is available, but that nonetheless at least some of the resulting perspectives would be helpful in understanding deep learning in a broader context.

Specifically, the study aimed to reproduce the classical analysis developed by Yosinski et al. [43] for quantifying the generality of neural network layers using transfer learning experiments. Whereas that work was focused on the representations learned by neural networks applied to a machine vision task, the work here is concerned with the representations learned by neural networks trained to solve PDEs. Essentially, a large ensemble of neural networks were trained to solve a family of PDEs related by a problem parameter x' , and the resulting internal representations were analysed as x' was varied. The main result is that, for a given choice of x' , the neural networks always learn essentially the same representations as long as they have sufficient capacity. The analysis of these “general” features provides additional insights into the behaviour of the NNM.

5.2 Results

5.2.1 Intrinsic dimensionality

The main results of the study are presented in Fig. 3 of the manuscript (Sec. 5.3). Three metrics are reported: intrinsic dimensionality, reproducibility, and specificity. All metrics are shown with error bars that indicate maximum and minimum values over all experiments. These errors bars are consistently very small, demonstrating that the NNM is behaving very consistently across different random initializations and different choices of the PDE parameter x' .

Intrinsic dimensionality, as defined more precisely in the manuscript, is approximately equal to the number of significant principal components in the hidden layers of various neural networks. The manuscript argues that this is effectively a measure of how much information is being stored in the hidden layers' representations. For every choice of x' , each hidden layer exhibits three regimes with increasing network width. At small widths, intrinsic dimensionality equals the width: the networks have insufficiently many degrees of freedom, and each new degree of freedom encodes important improvements to the solution. However, at large widths, the intrinsic dimensionality appears to converge to a constant value slightly smaller than its peak value. The most salient result is that the intrinsic dimensionality is far smaller than the total width: very wide neural networks do not use all of their degrees of freedom. The deeper hidden layers appear to utilize increasingly more of their degrees of freedom, with the limiting intrinsic dimensionality growing linearly with depth.

This result provides an interesting perspective on the theory-to-practice gap and the underwhelming convergence of the NNM with respect to its total number of degrees of freedom. It appears that the effective number of degrees of freedom in an NNM solution (i.e., its intrinsic dimensionality) is much smaller than its actual number degrees of freedom (i.e., the total number of weights and biases). Furthermore, it appears that the effective number of degrees of freedom converges to some maximum at large widths. Increasing width beyond this point does not change the function learned by the NNM. In fact, this interpretation is consistent with the measurements of error versus capacity shown in Chapter 4.

5.2.2 Reproducibility and specificity

Reproducibility compares the hidden representations learned by a given neural network architecture (i.e., of fixed width) applied to a given PDE (i.e., fixed x') when training is conducted from different initial random guesses for the network's weights and biases. A high reproducibility means that networks learn the same representations when trained to solve the same problem, regardless of the random state at initialization.

The results in Fig. 3(b) of the manuscript (Sec. 5.3) show that reproducibility increases monotonically with increasing width. Whereas the intrinsic dimensionality analysis indicated that sufficiently wide networks always learn the same number of features, the convergence of reproducibility suggests that they also learn the same specific set of features.

How does this unique reproducible set of features depend on the PDE parameter x' ? This is explored by the metric named specificity (shown in Fig. 3(c) of the manuscript) which is obtained by comparing the representations learned by neural networks with the same architecture applied to different PDE problems (i.e., different values of x'). As shown in Fig. 2, the similarity between hidden layers decreases gradually with the difference in the x' values for which the two solutions were obtained. Specificity summarizes the magnitude of this difference, with a small specificity indicating that the features do not change very much with x' , and vice versa.

Fig. 3(c) reveals a seemingly low specificity for most of the hidden layers investigated. Specifically, the first two hidden layers appear to learn general features (i.e., that change little with x') in all cases. The third (i.e.,

second-to-last) also learns general features, as long as the width of the network is large enough. Conversely, the final layer learns very specific features in all cases.

The fact that many of these features are approximately independent of x' has implications for the use of the NNM on parametrized PDEs. In particular, this suggests that transfer learning protocols may be practically useful if a PDE is to be solved repeatedly for many values of its parameters one after the other. The solution for one choice of the parameters may be a very good initial guess for the solution for another choice of the parameters.

5.2.3 Interpreting the features

The results suggest that the NNM learns a small, reproducible, and general set of features to represent the PDE solution. What information is encoded in these features? Can they be analysed to provide insights into the structure of the PDE solution, perhaps in a manner analogous to the qualitative interpretations assigned to Fourier representations of PDE solutions?

The manuscript (Sec. 5.3) includes some preliminary work in this direction. Visualized as two-dimensional functions of the PDE domain, the features in the first hidden layer appear to coincide with the key geometric features of the domain (corners and walls). Further investigation in this direction might provide insights into how the NNM is able to handle complicated and/or high-dimensional geometries more efficiently than mesh-based methods.

5.3 Manuscript

Neural Networks Trained to Solve Differential Equations Learn General Representations

Martin Magill

U. of Ontario Inst. of Tech.
martin.magill1@uoit.net

Faisal Z. Qureshi

U. of Ontario Inst. of Tech.
faisal.qureshi@uoit.ca

Hendrick W. de Haan

U. of Ontario Inst. of Tech.
hendrick.dehaan@uoit.ca

Abstract

We introduce a technique based on the singular vector canonical correlation analysis (SVCCA) for measuring the generality of neural network layers across a continuously-parametrized set of tasks. We illustrate this method by studying generality in neural networks trained to solve parametrized boundary value problems based on the Poisson partial differential equation. We find that the first hidden layers are general, and that they learn generalized coordinates over the input domain. Deeper layers are successively more specific. Next, we validate our method against an existing technique that measures layer generality using transfer learning experiments. We find excellent agreement between the two methods, and note that our method is much faster, particularly for continuously-parametrized problems. Finally, we also apply our method to networks trained on MNIST, and show it is consistent with, and complimentary to, another study of intrinsic dimensionality.

1 Introduction

Generality of a neural network layer indicates that it can be used successfully in neural networks trained on a variety of tasks [19]. Previously, Yosinski et al. [19] developed a method for measuring layer generality using transfer learning experiments, and used it to compare generality of layers between two image classification tasks. In this work, we will study the generality of layers across a continuously-parametrized set of tasks: a group of similar problems whose details are changed by varying a real number. We found the transfer learning method for measuring generality prohibitively expensive for this task. Instead, by relating generality to similarity, we develop a computationally efficient measure of generality that uses the singular vector canonical correlation analysis (SVCCA).

We demonstrate this method by measuring layer generality in neural networks trained to solve differential equations. We train fully-connected tanh neural networks (NNs) to solve Poisson's equation with a parametrized source term. The parameter of the source defines a family of related boundary value problems (BVPs), and we measure the generality of layers in the trained NNs as the parameter varies. We find the first layers to be general, and deeper layers to be progressively more specific. Using the SVCCA, we are also able to visualize and interpret these general first layers.

We validate our approach by reproducing a subset of our results using the transfer learning experimental protocol of Yosinski et al. [19]. These very different methods produce consistent measurements of generality. Further, our technique is several orders of magnitude faster to compute. Finally, we apply our method to ReLU networks trained on the MNIST dataset [9], and compare to work by Li et al. [11]. We discuss how the two analyses differ, but confirm that our results are consistent with theirs.

The main contributions of this work are:

1. We develop a method for efficiently computing layer generality over a continuously-parametrized family of tasks using the SVCCA.

2. Using this method, we demonstrate generality in the first layers of NNs trained to solve problems from a parametrized family of BVPs. We find that deeper layers become successively more specific to the problem parameter, and that network width can play an important role in determining layer generality.
3. We visualize the principal components of the first layers that were found to be general. We interpret them as generalized coordinates that reflect important subregions of the unit square.
4. We validate our method for measuring layer generality using the transfer learning experimental protocol developed by Yosinski et al. [19]. We find that both approaches identify the same trends in layer generality as network width is varied, but that our approach is significantly more computationally efficient, especially for continuously parametrized tasks.
5. We define a measure of the intrinsic dimensionality of a layer, and contrast it with that of Li et al. [11]. We show the two are consistent for networks trained on the MNIST dataset.

1.1 Neural networks for differential equations

The idea to solve differential equations using neural networks was first proposed by Dissanayake and Phan-Thien [3]. They trained neural networks to minimize the loss function

$$\mathcal{L} = \int_{\Omega} \|G[u](x)\|^2 dV + \int_{\partial\Omega} \|B[u](x)\|^2 dS, \quad (1)$$

where G and B are differential operators on the domain Ω and its boundary $\partial\Omega$ respectively, $G[u] = 0$ is the differential equation, and $B[u] = 0$ describes boundary conditions. Training data consisted of coordinates $x \in \Omega$ sampled from a mesh, used to numerically approximate the integrals in \mathcal{L} at each epoch. Similar methods were proposed by van Milligen et al. [17] and Lagaris et al. [7]. Many innovations have been made since, most of which were reviewed by Schmidhuber [15] and in a book by Yadav et al. [18]. Sirignano and Spiliopoulos [16] as well as Berg and Nyström [2] illustrated that the training points can be obtained by randomly sampling the domain rather than using a mesh, which significantly enhances performance in higher-dimensional problems. In fact, Sirignano and Spiliopoulos [16] and Han et al. [5] have demonstrated that neural networks can be used to solve partial differential equations in hundreds of dimensions, which is a revolutionary result. Traditionally, such problems have often been considered infeasible, since traditional mesh-based solvers suffer from an exponential growth in computational complexity with increasing problem dimensionality.

There are at least two good reasons for studying neural networks that solve differential equations (referred to hereafter as DENNs). The first is their unique advantages over traditional methods for solving differential equations [2–5, 7, 16, 17]. The second is that they offer an opportunity to study the behaviour of neural networks in a well-understood context [2]. Most applications of neural networks, such as machine vision and natural language processing, involve solving problems that are ill-defined or have no known solutions. Conversely, there exists an enormous body of literature on differential equation problems, detailing when solutions exist, when they are unique, and how they will behave. Indeed, in some cases the exact solutions to the problem can be obtained analytically.

1.2 Studying the generality of features with transfer learning

Transfer learning is a major topic in machine learning, reviewed for instance by Pan and Yang [13]. Generally, transfer learning in neural networks entails initializing a recipient neural network using some of the weights from a donor neural network that was previously trained on a related task.

Yosinski et al. [19] developed an experimental protocol for quantifying the generality of neural network layers using transfer learning experiments. They defined generality as the extent to which a layer from a network trained on some task A can be used for another task B. For instance, the first layers of CNNs trained on image data are known to be general: they always converge to the same features, namely Gabor filters (which detect edges) and color blobs (which detect colors) [6, 8, 10].

In the protocol developed by Yosinski et al. [19], the first n layers from a donor network trained on task A are used to initialize the first n layers of a recipient network. The remaining layers of the recipient are randomly initialized, and it is trained on task B. However, the transferred layers are frozen: they are not updated during training on task B. The recipient is expected to perform as well on task B as did the donor on task A if and only if the transferred layers are general.

In practice, however, various other factors can impact the performance of the recipient on task B, so Yosinski et al. [19] also included three control tests. The first control is identical to the actual test, except that the recipient is trained on the original task A; this control identifies any fragile co-adaptation between consecutive layers [19]. The other two controls entail repeating the actual test and the first control, but allowing the transferred layers to be retrained. When the recipient is trained on task A with retraining, performance should return to that of the donor network. When it is trained on task B with retraining, Yosinski et al. [19] found the recipient actually outperformed the donor.

Yosinski et al. [19] successfully used their method to confirm the generality of the first layers of image-based CNNs. Further, they also discovered a previously unknown generality in their second layers. This methodology, however, was constructed for the binary comparison of two tasks A and B. In the present work, we are interested in studying layer generality across a continuously parametrized set of tasks (given by a family of BVPs), and the transfer learning methodology is prohibitively computationally expensive. Instead, we will use a different approach, based on the SVCCA, which we will then validate against the method of Yosinski et al. [19] on a set of test cases.

1.3 SVCCA: Singular Vector Canonical Correlation Analysis

Yosinski et al. [19] defined generality of a layer to mean that it can be used successfully in networks performing a variety of tasks. This definition was motivated, however, by observing that the first layers of image-based CNNs converged upon similar features across many network architectures and applications. We argue that these two concepts are related: if a certain representation leads to good performance across a variety of tasks, then well-trained networks learning any of those tasks will discover similar representations. In this spirit, we define a layer to be general across some group of tasks if similar layers are consistently learned by networks trained on any of those tasks. To use this definition to measure generality, then, we require a quantitative measure of layer similarity.

Recently, the SVCCA was demonstrated by Raghu et al. [14] to be a powerful method for measuring the similarity of neural network layers [14]. The SVCCA considers the activation functions of a layer’s neurons evaluated at points sampled throughout the network’s input domain. In this way it incorporates problem-specific information, and as a result it outperforms older metrics of layer similarity that only consider the weights and biases of a layer. For instance, Li et al. [12] proposed measuring layer similarity by finding neuron permutations that maximized correlation between networks. As a linear algebraic algorithm, however, the SVCCA is more computationally efficient than permutation-based methods. Similarly, Berg and Nyström [2] have concurrently attempted to study the structure of DENNs by analyzing weights and biases directly, but found the results to be too sensitive to the local minima into which their networks converged.

Following Raghu et al. [14], we will use the SVCCA to define a scalar measure of the similarity of two layers. The SVCCA returns canonical directions in which two layers are maximally correlated. They defined the SVCCA similarity ρ of two layers as the average of these optimal correlation values. However, this quantity depends explicitly on the layers’ widths, independently of the functions the layers represent. Here, instead of the mean, we will define ρ as the sum of these correlations. Since we typically found that the majority of the correlations were nearly 1.0 or nearly 0.0, this SVCCA similarity roughly measures the number of significant dimensions shared by two layers. In particular, since the SVCCA between a layer and itself is equivalent to a principal component analysis, we will use the SVCCA self-similarity as an approximate measure of a layer’s intrinsic dimensionality.

This concept of intrinsic dimensionality differs from that recently proposed by Li et al. [11]. They constrained network weights during training to a random d -dimensional subspace for various values of d , and defined the intrinsic dimensionality of a given network on a given task as the smallest d for which good performance is achieved. This metric differs from ours in two important ways. First, their algorithm finds the *smallest representation* required to solve a problem, whereas our definition directly analyses the *actual representations* learned in practice. Specifically, they consider a strongly regularized auxiliary problem, whereas we examine given solutions directly. Second, their measure is based on the *performance* of representations, whereas ours measures *structure*. Indeed, since Raghu et al. [14] were able to compress models with little loss of performance by keeping only the first few important SVCCA directions, the remaining important SVCCA directions must describe structures present in layers’ representations that do not directly influence network performance. Our method is complimentary to that of Li et al. [11]: theirs finds compact solutions that perform well, which is of practical value, but ours can examine any given network without altering its properties.

2 Methodology

2.1 Problem definition

Following concurrent work by Berg and Nyström [2], we will study the structure of DENNs on a parametrized family of PDEs. Berg and Nyström [2] used a family of Poisson equations on a deformable domain. They attempted to characterize the properties of the DENN solutions by studying the variances of their weights and biases. However, they reported that their metrics of study were too sensitive to the local minima into which their solutions converged for them to draw conclusions [2].

In this work, we have repeated this experiment, but using the SVCCA as a more robust tool for studying the structure of the solutions. The family of PDEs considered here was

$$\nabla^2 u(x, y) = s(x, y) \quad \text{for } (x, y) \in \Omega, \quad (2)$$

$$u(x, y) = 0, \quad \text{for } (x, y) \in \partial\Omega, \quad (3)$$

where $\Omega = [-1, 1] \times [-1, 1]$ is the domain and $-s(x, y)$ is a nascent delta function given by

$$s(x, y) = -\delta_r(x, y; x', y') = -\frac{\exp\left(-\frac{(x-x')^2 + (y-y')^2}{2r^2}\right)}{2\pi r^2}, \quad (4)$$

which satisfies $\lim_{r \rightarrow 0} \delta_r(x, y; x', y') = \delta(x - x')\delta(y - y')$, where δ is the Dirac delta function. For the present work, we will fix $y' = 0$ and $r = 0.1$, and vary only x' . Thus, the BVPs describe the electric potential produced by a localized charge distribution on a square domain with grounded edges. The problems are parametrized by x' . We relegate deformable domains to future work.

2.2 Implementation details

The networks used in this work were all fully-connected with 4 hidden layers of equal width, implemented in TensorFlow [1]. Activation functions were tanh, except in Section 3.4, where ReLU was used. Given inputs x and y , the network was trained to directly approximate $u(x, y)$, the solution to a BVP from the family of BVPs described above. Training followed the DGM methodology of Sirignano and Spiliopoulos [16]. More implementation details are discussed in the supplemental material. Since this work was not focused on optimization of performance, we used relatively generic hyperparameters whenever possible to ensure that our results are reasonably general.

3 Results

3.1 Quantifying layer generality in DENNs using SVCCA

In this section, we use the SVCCA to study the generality of layers in DENNs trained to solve our family of BVPs. We train DENNs to solve the BVPs for a range of x' values, each from four different random initializations per x' value. We will refer to the different random initializations as the first through fourth random seeds for each x' value (see the supplemental material for details about the random seed construction). First, we present results for networks of width 20. We condense our analysis into three metrics, and then study how those metrics vary with network width.

Figure 1 shows the SVCCA similarities computed between the first, third, and fourth hidden layers of networks of width 20. The matrix for the second hidden layer is omitted, but closely resembles that for the first hidden layer. The (i, j) th element of the matrices show the SVCCA similarity computed between the given layers of the i th and j th networks in our dataset. Since the SVCCA similarity does not depend on the order in which the layers are compared, the matrices are symmetric. The black grid lines of the matrices separate layers by the x' values on which they were trained, and the four seeds for each x' are grouped between the black grid lines.

The matrices evidently exhibit a lot of symmetry, and can be decomposed into subregions. The first is the diagonal of the matrices, which contains the self-similarities of the layers, denoted ρ_{self}^l in the l th layer. The second region contains the matrix elements that lie inside the block diagonal formed by the black grid lines, but that are off the main diagonal. These indicate the similarities between layers trained on the same x' values, but from different random seeds, and will be denoted $\rho_{\Delta x'=0}^l$. The remaining matrix elements were found to be equivalent along the block-off-diagonals. These

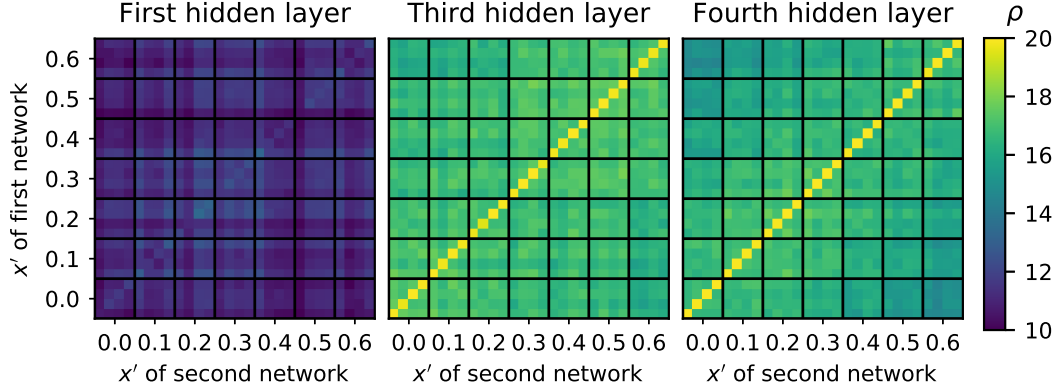


Figure 1: Matrices of layer-wise SVCCA similarities between the first, third, and fourth hidden layers of networks of width 20 trained at various x' values, with four random seeds per position. The black lines group layers on each axis by the x' values at which they were trained. For each x' value, the four entries correspond to four distinct random seeds. Thus the matrix diagonals contain self-similarities, the block diagonals formed by black lines contain similarities across random seeds at a fixed x' , and the remaining entries correspond to comparisons between distinct x' values.

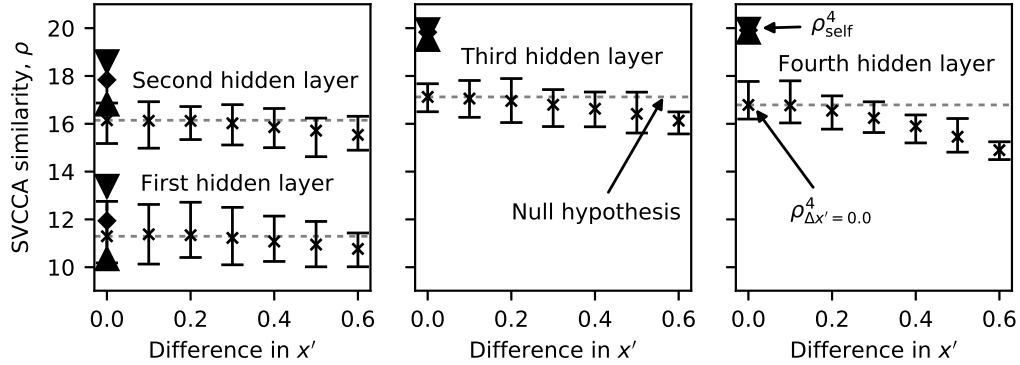


Figure 2: For each layer, crosses show mean similarities between distinct layers as a function of the difference in the x' values at which they were trained. Diamonds show mean self-similarities. For both, error bars indicate maximum and minimum values. The gray lines show the null hypothesis described in the text, namely that the representations are independent of x' .

correspond to all similarities computed between l th layers from networks trained on x' values that differ by $\Delta x'$, which we will denote $\rho_{\Delta x'}^l$.

With this decomposition in mind, the matrices can be represented more succinctly as the plots shown in Figure 2. The diamonds and their error bars show the mean, minima, and maxima of ρ_{self}^l in each layer l . The crosses and their error bars show the means, minima, and maxima of $\rho_{\Delta x'}^l$ for varying source-to-source distances $\Delta x'$. As described above, the statistics of $\rho_{\Delta x'=0}^l$ were computed excluding the self-similarities ρ_{self}^l . The dashed gray lines show $\langle \rho_{\Delta x'=0}^l \rangle$ for each layer, and are used below to quantify specificity. We show the minima and maxima of the data in order to emphasize that our decomposition of the matrices in Figure 1 accurately reflects the structure of the data.

In the plots of Figure 2, the gap between ρ_{self}^l and $\rho_{\Delta x'=0}^l$ indicates the extent to which different random initializations trained on the same value of x' converge to the same representations. For this reason, we define the ratio $\langle \rho_{\Delta x'=0}^l \rangle / \langle \rho_{\text{self}}^l \rangle$ as the reproducibility. It measures what fraction of a layer's intrinsic dimensionality is consistently reproduced across different random seeds. We see that, for networks of width 20, the first layer is highly reproducible, and the second is mostly

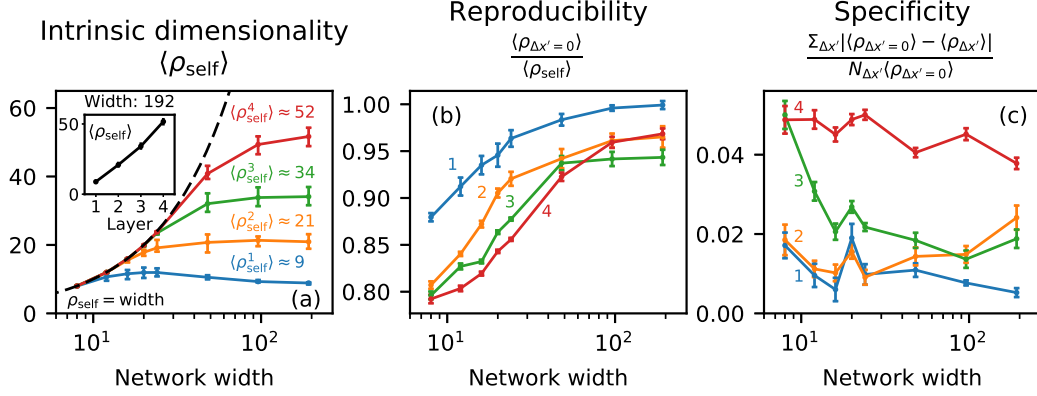


Figure 3: The intrinsic dimensionality, reproducibility, and specificity of the four layers at varying width. The lines indicate mean values. The error bars on intrinsic dimensionality indicate maxima and minima, whereas the error bars on reproducibility and specificity indicate estimated uncertainty on the means (discussed in the supplemental material). Numbers indicate layer numbers. The inset in (a) shows the limiting dimensionalities of the four layers at width 192.

reproducible. Conversely, the third and fourth layers in Figure 2 have a gap of roughly 3 out of 20 between $\langle \rho_{\Delta x'=0}^l \rangle$ and $\langle \rho_{\text{self}}^l \rangle$: networks from different random seeds at the same x' value are consistently dissimilar in about 15% of their canonical components.

We can use the plots of Figure 2 to quantify the generality of the layers. When a layer is general across x' , the similarity between layers should not depend on the x' values at which they were trained. Thus the $\rho_{\Delta x'}^l$ values should be distributed no differently than $\rho_{\Delta x'=0}^l$. Visually, when a layer is general, the crosses in Figure 2 should be within error of the dashed grey lines. Similarly, the distance between the crosses and the dashed line is proportional to the specificity of a layer. Thus we can see in Figure 2 that, for networks of width 20, the first and second layers appear to be general, whereas the third and fourth are progressively more specific.

To quantify this, we will define a layer's specificity as the average over $\Delta x'$ of $|\langle \rho_{\Delta x'=0}^l \rangle - \langle \rho_{\Delta x'}^l \rangle| / \langle \rho_{\Delta x'=0}^l \rangle$. In Figure 2, this is equivalent to the mean distance from the crosses to the dashed grey line, normalized by the height of the dashed grey line. Equivalently, it is the ratio of the area delimited by the crosses and the dashed line to the area under the dashed line. It can also be interpreted as a numerical estimation of the normalized L_1 norm of the difference between the measured $\langle \rho_{\Delta x'=x}^l \rangle$ and the null hypothesis of a perfectly general layer. By this definition, a layer will have a specificity of 0 if and only if it has similar representations across all values of $\Delta x'$. Furthermore, the specificity is proportional to how much $\langle \rho_{\Delta x'}^l \rangle$ varies with $\Delta x'$. Thus the specificity metric we defined here is indeed consistent with the accepted definitions of generality and specificity.

The same experiments described above for networks of width 20 were repeated for widths of 8, 12, 16, 24, 48, 96, and 192. Figure 3 shows the measured intrinsic dimensionalities, reproducibilities, and specificities of the four layers. The error bars on the intrinsic dimensionalities show minima and maxima, emphasizing that these measurements were consistent across different values of x' and different random seeds. The error bars on the reproducibility and specificity show the estimated uncertainty on the means, as discussed in the supplemental material.

In narrow networks, the layers' intrinsic dimensionalities (Fig. 3(a)) equal the network width. As the network width increases, these dimensionalities drop below the width, and appear to converge to finite values. We suggest that, for a fixed x' value, there are finite-dimensional representations to which the layers will consistently converge, so long as the networks are wide enough to support those representations. If the networks are too narrow, they converge to some smaller-dimensional projections of those representations. The reproducibility plots (Fig. 3(b)) support this interpretation, as the reproducibilities grow with network width. Furthermore, they are smaller for deeper layers, except in very wide networks where the fourth layer becomes more reproducible than the second and third. This could be related to convergence issues in very wide networks, as discussed below.

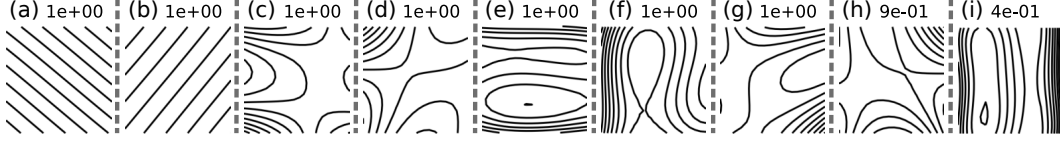


Figure 4: Plots of the first nine principal components of the first layer of a network trained on $x' = 0.6$ (obtained by self-SVCCA). The numbers show the SVCCA correlations of each component.

The limiting dimensionalities increase nearly linearly with layer depth, as shown in the inset of Figure 3(a). This has implications for sparsification of DENNs, as Raghu et al. [14] showed successful sparsification by eliminating low-correlation components of the SVCCA. Similarly, DENN architectures that widen with depth may be more optimal than fixed-width architectures.

The specificity (Fig. 3(c)) varies more richly with network width. Overall, the first layer is most general, and successive layers are progressively more specific. Over small to medium widths, the second layer is nearly as general as the first layer; the third layer transitions from highly specific to quite general; and the fourth layer remains consistently specific. In very wide networks, however, the second and third layers appear to become more specific, whereas the fourth layer becomes somewhat more general. Future work should explore the behaviour at large widths, but we speculate that it may be related to changes in training dynamics at large widths. As discussed in the supplemental material, very wide networks seemed to experience very broad minima in the loss landscape, so our training protocol may have terminated before the layers converged to optimal and general representations.

The overall trends in specificity discussed above are interrupted near widths of 16, 20, and 24. All four layers appear somewhat more general than expected at width 16, and then more specific than expected at width 20. By width 24 and above they resume a more gradual variation with width. This is a surprising result that future work should explore more carefully. It occurs as the network width exceeds the limiting dimensionality of the second layer, which may play a role in this phenomenon.

3.2 Visualizing and interpreting the canonical directions

We have shown that the first layers of the DENNs studied here converge to general 9-dimensional representations independent of the parameter x' . Figure 4 shows a visualization of the first 9 principal components (obtained by self-SVCCA) of the first layer of a network of width 192 trained at $x' = 0.6$, shown as contour maps. We interpret these as generalized coordinates. The contours are densest where the corresponding coordinates are most sensitive. It is clear that the first 2 of these 9 components capture precisely the same information as x and y , but rotated. The remaining components act together to identify 9 regions of interest in the domain: the 4 corners, the 4 walls, and the center. For instance, component (e) describes the distance from the top and bottom walls; component (i) does the same for the left and right walls; and component (d) describes distance to the upper-left corner. We found the first layers could be interpreted this way at any x' and whether we found components by self-SVCCA or cross-SVCCA, and have included examples of this in the supplemental material. The components are always some linear combination of x , y , and the 9 regions described above.

Surprisingly, we found that the SVCCA was numerically unstable. Repeated analyses of the same networks produced slightly different components, although the correlation vectors were very stable. We see two factors contributing to this problem. Firstly, the first 7 or 8 correlation values of the first layer are all extremely close to 1 and, therefore, to one another. Thus the task of sorting the corresponding components is inevitably ill-conditioned. Second, the components appear to be paired into subspaces, such as the first two in Figure 4. Thus the task of splitting these subspaces into one-dimensional components is also ill-conditioned. We propose that future work should explore component analyses that search for closely-coupled components. This could resolve the numerical stability while also extracting even more structure about the layer representations.

3.3 Confirming generality by transfer learning experiments

In this section, we validate the method used to measure generality in Section 3.1 by repeating a subset of our measurements using the transfer learning technique established by Yosinski et al. [19]. We restricted our validation to a subset of cases because the transfer learning technique is significantly

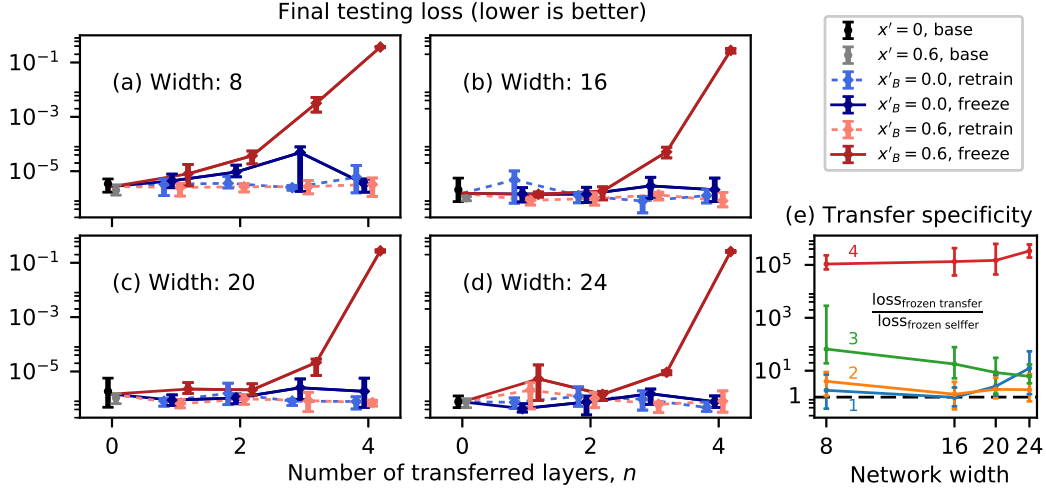


Figure 5: (a-d): Results of the transfer learning experiments conducted on networks of four different widths. Markers indicate means, and error bars indicated maxima and minima. At $n = 0$, the lines pass through the average of the two base cases. Donors were trained at $x'_A = 0$, and recipients at x'_B . (e): Measured transfer specificity as a function of network width. Numbers indicate layer number. Markers show the ratio of the mean losses, and error bars show the maximum and minimum ratios over all 16 combinations of the two losses. The dashed line shows a transfer specificity of 1.

more computationally expensive. To this end, we only trained donor networks at $x'_A = 0$ and measured generality towards $x'_B = 0.6$. Following Yosinski et al. [19], we will call the control cases with $x'_B = 0$ the selfer cases, and the experimental cases with $x'_B = 0.6$ the transfer cases. We show the results for widths of 8, 16, 20, and 24 in Figure 5(a-d). Throughout this section, we will refer to the measure of layer specificity we defined in Section 3.1 as the SVCCA specificity, to distinguish it from the measure of layer specificity obtained from the transfer learning experiments, which we call the transfer specificity. In Figure 5(a-d), the transfer specificity is given by the difference between the losses of the frozen transfer group (solid, dark red points) and those of the frozen selfer group (solid, dark blue points). It is immediately clear that the third and fourth layers are much more specific than the first and second at all widths. The specificities of the first, second, and fourth layers do not change very much with width, whereas the third layer appears to become more general with increasing width. These results are in agreement with those found with the SVCCA specificity.

To quantify these differences, we define a transfer specificity metric given by the ratio of the losses between the two frozen groups. This is shown in Figure 5(e), and it can be compared to the SVCCA specificities for the same widths, which lie in the leftmost third of Figure 3(c). The dashed line in Figure 5(e) shows a transfer specificity of 1, corresponding to a perfectly general layer. The first and second layers have transfer specificities of roughly 5, and are general (within error) at all widths. The fourth layer, on the other hand, has a transfer specificity of roughly 10^5 , and is highly specific at all widths. Whereas those layers' transfer specificities do not change significantly with width, the third layer becomes increasingly general as the width increases. Its transfer specificity decreases by roughly a factor of 4 from roughly 64 at width 8 to 18 at width 16. In Figure 3(c), by comparison, its SVCCA specificity drops from roughly 5% at width 8 to 2% at width 16. Thus the transfer specificity metric agrees with the main results of the SVCCA specificity: at all four widths, the first two layers are general, and the fourth is very specific; the third layer is specific, albeit much less so than the fourth, and becomes more general as the width increases.

Returning to Figure 5(a-d), recall that the remaining control groups also contain information about network structure. Any difference between the two selfer groups (the two blue series) indicates fragile co-adaptation. We note possible fragile co-adaptation at a width of 8, especially at $n = 3$. Future work should try measuring co-adaptation using the SVCCA, perhaps by measuring the similarities of different layers within the same network, as done by Raghu et al. [14]. Finally, any significant difference between the two retrained groups (the two dashed series) was meant to check if retraining transferred layers boosted recipient performance; however, this was not seen in any of our cases.

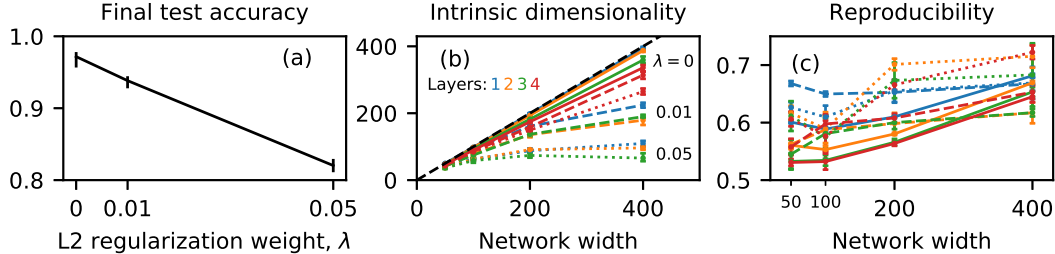


Figure 6: Test accuracies, intrinsic dimensionalities and reproducibilities of networks trained on the MNIST dataset for various L2 regularization weights λ and network widths. The error bars in (a) and (b) show maxima and minima; those in (c) show the estimated standard error.

Overall, the transfer specificity used by Yosinski et al. [19] shows good agreement with the SVCCA specificity we defined. We note however that the SVCCA specificity is much faster to compute. Both methods require the training of the original set of networks without transfer learning, which took about 2 hours per network using our methodology and hardware. We could then compute all the SVCCA specificities for this work in roughly 15 minutes. On the other hand, Figure 5 required hundreds of extra hours to compute, and only considers four widths and two x' values. That method would be prohibitively expensive for measuring generality in any continuously-parametrized problem.

3.4 Intrinsic dimensionality and reproducibility on MNIST

We also applied our metrics to the same networks trained instead on the MNIST dataset [9], and with ReLU activation functions rather than tanh. The networks were trained to minimize the classification cross entropy plus an L2 regularization term with weight λ . We used widths of 50, 100, 200, and 400; λ values of 0, 0.01, and 0.05; and four random seeds per combination. Li et al. [11] measured the intrinsic dimensionalities of such networks, and found them to be vastly overparametrized.

Figure 6(a) shows, for each λ , the range of test accuracies over all widths after training on 2000 batches of 100 images. Figures 6(b) and 6(c) show the intrinsic dimensionalities and reproducibilities, respectively, by width, layer number, and λ . Without regularization (i.e. with $\lambda = 0$), the intrinsic dimensionalities of all four layers are nearly equal to their widths. This apparent contradiction to Li et al. [11] arises because their method is itself strongly regularizing. As we increase λ , the intrinsic dimensionalities decrease more rapidly than performance, which is consistent with the results of Li et al. [11]. We find low reproducibility in all experiments, even though the accuracies and intrinsic dimensionalities are quite consistent across seeds. This suggests that, at fixed width and regularization strength, although the networks consistently converge to representations of the same dimension, and although they exhibit comparable accuracy, the details of the learned representations vary significantly across seeds. In other words, the optimal representations in this experiment are non-unique. Since our metrics can be computed efficiently, future work should explore how this conclusion evolves during training. This experiment illustrates how our first two metrics, developed for DENNs, can also be applied more broadly. Our metric of specificity is based on a continuously-changing task; extending this to MNIST could be done, for instance, by varying the relative sampling of the target classes.

4 Conclusion

In this paper, we presented a method for measuring layer generality over a continuously-parametrized set of problems using the SVCCA. Using this method, we studied the generality of layers in DENNs over a parametrized family of BVPs. We found that the first layer is general; the second is somewhat less so; the third is general in wide networks but specific in narrow ones; and the fourth is specific for widths up to 192. We visualized the general components identified in the first layers and interpreted them as generalized coordinates capturing features of interest in the input domain. We validated our method against the transfer learning protocol of Yosinski et al. [19]. The methods show good agreement, but our method is much faster, especially on continuously-parametrized problems. Finally, we contrasted our intrinsic dimensionality with that used by Li et al. [11]. The two are distinct but complimentary, and produce consistent results for networks trained on the MNIST dataset [9].

Acknowledgements

MM gratefully acknowledges funding from the Ontario Graduate Scholarship (OGS). FZQ gratefully acknowledges funding from the Natural Sciences and Engineering Research Council (NSERC) in the form of Discovery Grant 2015-04533. HWdH gratefully acknowledges funding from the Natural Sciences and Engineering Research Council (NSERC) in the form of Discovery Grant 2014-06091.

References

- [1] Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S. Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Ian Goodfellow, Andrew Harp, Geoffrey Irving, Michael Isard, Yangqing Jia, Rafal Jozefowicz, Lukasz Kaiser, Manjunath Kudlur, Josh Levenberg, Dan Mané, Rajat Monga, Sherry Moore, Derek Murray, Chris Olah, Mike Schuster, Jonathon Shlens, Benoit Steiner, Ilya Sutskever, Kunal Talwar, Paul Tucker, Vincent Vanhoucke, Vijay Vasudevan, Fernanda Viégas, Oriol Vinyals, Pete Warden, Martin Wattenberg, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng. TensorFlow: Large-scale machine learning on heterogeneous systems, 2015. URL <https://www.tensorflow.org/>. Software available from tensorflow.org.
- [2] Jens Berg and Kaj Nyström. A unified deep artificial neural network approach to partial differential equations in complex geometries. *arXiv preprint arXiv:1711.06464*, 2017.
- [3] M. W. M. G. Dissanayake and N. Phan-Thien. Neural-network-based approximations for solving partial differential equations. *Communications in Numerical Methods in Engineering*, 10(3):195–201, 1994. doi: 10.1002/cnm.1640100303. URL <https://onlinelibrary.wiley.com/doi/abs/10.1002/cnm.1640100303>.
- [4] Philipp Grohs, Fabian Hornung, Arnulf Jentzen, and Philippe von Wurstemberger. A proof that artificial neural networks overcome the curse of dimensionality in the numerical approximation of Black-Scholes partial differential equations. *arXiv preprint arXiv:1809.02362*, 2018.
- [5] Jiequn Han, Arnulf Jentzen, and E Weinan. Solving high-dimensional partial differential equations using deep learning. *Proceedings of the National Academy of Sciences*, 115(34):8505–8510, 2018.
- [6] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. ImageNet Classification with Deep Convolutional Neural Networks. In *Advances in neural information processing systems*, pages 1097–1105, 2012.
- [7] Isaac E Lagaris, Aristidis Likas, and Dimitrios I Fotiadis. Artificial neural networks for solving ordinary and partial differential equations. *IEEE Transactions on Neural Networks*, 9(5):987–1000, 1998.
- [8] Quoc V Le, Alexandre Karpenko, Jiquan Ngiam, and Andrew Y Ng. ICA with reconstruction cost for efficient overcomplete feature learning. In *Advances in neural information processing systems*, pages 1017–1025, 2011.
- [9] Yann LeCun. The MNIST database of handwritten digits. <http://yann.lecun.com/exdb/mnist/>, 1998.
- [10] Honglak Lee, Roger Grosse, Rajesh Ranganath, and Andrew Y Ng. Convolutional deep belief networks for scalable unsupervised learning of hierarchical representations. In *Proceedings of the 26th annual international conference on machine learning*, pages 609–616. ACM, 2009.
- [11] Chunyuan Li, Heerad Farkhor, Rosanne Liu, and Jason Yosinski. Measuring the Intrinsic Dimension of Objective Landscapes. *arXiv preprint arXiv:1804.08838*, 2018.
- [12] Yixuan Li, Jason Yosinski, Jeff Clune, Hod Lipson, and John Hopcroft. Convergent learning: Do different neural networks learn the same representations? In *Feature Extraction: Modern Questions and Challenges*, pages 196–212, 2015.
- [13] Sinno Jialin Pan and Qiang Yang. A survey on transfer learning. *IEEE Transactions on knowledge and data engineering*, 22(10):1345–1359, 2010.
- [14] Maithra Raghu, Justin Gilmer, Jason Yosinski, and Jascha Sohl-Dickstein. SVCCA: Singular Vector Canonical Correlation Analysis for Deep Learning Dynamics and Interpretability. In *Advances in Neural Information Processing Systems*, pages 6078–6087, 2017.
- [15] Jürgen Schmidhuber. Deep learning in neural networks: An overview. *Neural networks*, 61:85–117, 2015.
- [16] Justin Sirignano and Konstantinos Spiliopoulos. DGM: A deep learning algorithm for solving partial differential equations. *arXiv preprint arXiv:1708.07469*, 2017.

- [17] B Ph van Milligen, V Tribaldos, and J A Jiménez. Neural Network Differential Equation and Plasma Equilibrium Solver. *Physical Review Letters*, 75(20):3594, 1995.
- [18] Neha Yadav, Anupam Yadav, and Manoj Kumar. *An introduction to neural network methods for differential equations*. Springer, 2015.
- [19] Jason Yosinski, Jeff Clune, Yoshua Bengio, and Hod Lipson. How transferable are features in deep neural networks? In *Advances in neural information processing systems*, pages 3320–3328, 2014.

Neural Networks Trained to Solve Differential Equations Learn General Representations:

Supplemental Material

Martin Magill
U. of Ontario Inst. of Tech.
martin.magill1@uoit.net

Faisal Z. Qureshi
U. of Ontario Inst. of Tech.
faisal.qureshi@uoit.ca

Hendrick W. de Haan
U. of Ontario Inst. of Tech.
hendrick.dehaan@uoit.ca

A Additional implementation details

We used the tanh activation function for the DENNs in our work, and chose $\Omega = [-1, 1] \times [-1, 1]$ so that the input neurons had the same range as the hidden neurons. We used the tanh activation function as it outperformed the sigmoid, which was used in many previous works on DENNs (e.g. [2, 5, 6]). This is consistent with the general guidelines on efficient backpropagation offered by LeCun et al. [7]. LeCun et al. [7] also propose a rescaling of the tanh activation function that might improve performance when used with correspondingly rescaled inputs. Implementing this function using the python interface of TensorFlow [1] did not improve performance significantly, although a lower-level implementation might do better. Since this work was not performance-oriented, this is relegated to future work.

Piecewise-linear activations functions like ReLU were found to be incompatible with DENNs; the PDEs are defined in terms of the network’s derivatives with respect to its inputs, and piecewise-linear activations functions produce solutions that are locally flat. Although such activation functions could still be used approximate the solution functions in theory (since Sonoda and Murata [10] showed that they still lead to universal approximation theorems), any function represented by them has no higher derivatives at any point in the domain, and so cannot learn by backpropagation from the loss functions used with DENNs.

Following Sirignano and Spiliopoulos [9], the loss function defined as

$$\mathcal{L}(x, y) = (\nabla^2 u - s)^2 (1 - I_{\delta\Omega}) + \eta u^2 I_{\delta\Omega} \quad (1)$$

where

$$I_{\delta\Omega}(x, y) = \begin{cases} 1, & (x, y) \in \partial\Omega \\ 0, & (x, y) \notin \partial\Omega \end{cases} \quad (2)$$

is the indicator function for the boundary of the domain. We chose $\eta = 1$, assigning equal weight to the PDE and loss terms. Since only one term is non-zero for any given point (x, y) , the relative importance of the PDE and BC are therefore controlled directly by the relative sampling of the interior and boundary of the domain.

In defining the loss function, the L_2 norm was consistently found to lead to better training performance than the L_1 norm. This was not clear *a priori*, as the problem studied here is essentially one of approximating a specific function, rather than learning from a statistical process. In this case, then, one might expect the L_1 norm to converge to sharper minima in the loss landscape, in much the same way that L_1 regularization encourages sparsification more readily than L_2 regularization. Future work should explore why training seemed to be less efficient with this loss norm.

During training, batches of training points were randomly sampled from the domain. Specifically, 10^4 points were randomly drawn in the interior of Ω , and then 10^4 more points were drawn on *each* of

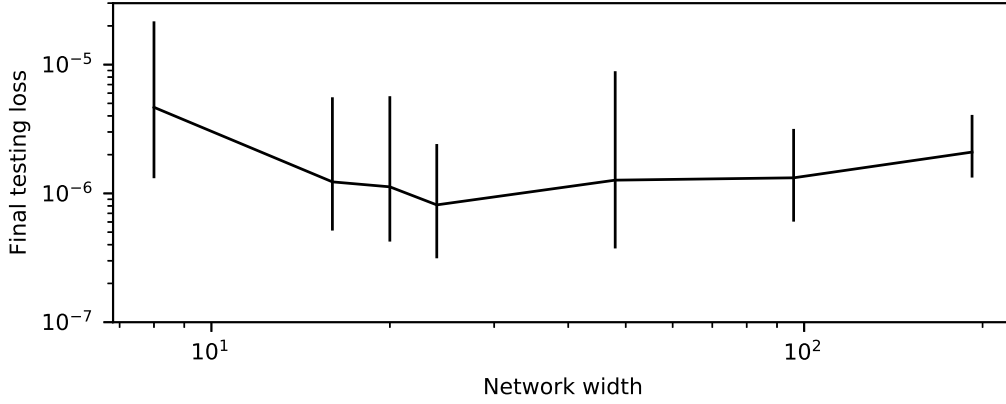


Figure 1: Final testing losses after training of all the networks used in the SVCCA-based generality measurements. Error bars show maxima and minima.

the four edges of the domain. Thus, although the loss function assigned equal weight on the PDE and BC terms, the data sampling favoured the boundaries significantly. The size of the training set was selected to optimally utilize the available GPU resources (NVIDIA GTX 1080). Since resampling the data set was computationally expensive, it was only changed every 100 training epochs.

The loss function was evaluated over a testing set of points, which was randomly generated in the same manner as the training set, but with ten times more points from each region of the domain (for a total of 5×10^5 points). This size was deemed to be more than large enough to fully resolve all features of the problem. As such, the testing set was only generated once for each experiment. The loss was computed over the testing set every 1000 epochs. Training proceeded until the testing loss failed to improve after five consecutive evaluations. This was found to reliably produce thoroughly-converged solutions in early tests, although the number of epochs before convergence varied significantly across different random initializations for the same experiments. As discussed below, this training protocol may have encountered issues for very wide networks.

Weights were randomly initialized according to the Tensorflow implementation of the Glorot uniform initializer [1, 3]. Optimization was conducted using the default TensorFlow implementation of the Adam optimizer [1, 4].

Because networks were fast to train (taking at most a few hours to converge), many instances of training (starting from different random seeds) were conducted for each experiment. In order to reduce the chance of artifacts arising from random seed correlations, the seeds were set according to the formula

```
seed = int(str(nxp+1) +
           "%02d"%(seed_core) +
           str(np.abs(nr)+1) +
           str(n_layers) +
           "%03d"%(neurons_per_layer))
```

where n_{xp} indicates the number of increments of 0.1 by which the source has been translated in x' , nr indicates number of times the effective width of the source was increased by a factor of 2 from $r = 0.1$, n_{layers} is the number of hidden layers in the network and $neurons_per_layer$ is the number of neurons in each hidden layer. Finally, $seed_core$ is a number use to distinguish between different repetitions of the same experiment. Note that several of these parameters were not varied in the current work, but this convention was selected for compatibility with future work.

Figure 1 shows the final testing losses achieved by all the networks trained for the SVCCA-based generality measurements. In other words, these are the losses for all networks trained for this work, except those trained using transfer learning. Performance improved with width until a width of 24. Wider networks achieved somewhat worse performance. We observed very wide networks during

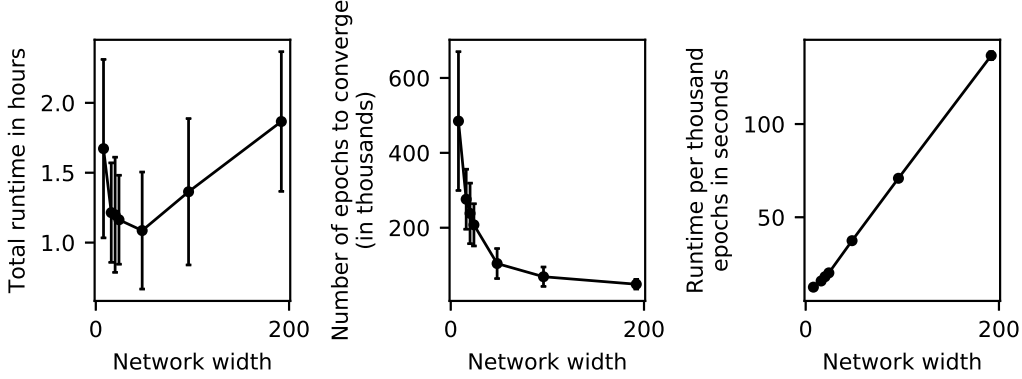


Figure 2: Training statistics for all the networks used in the SVCCA-based generality measurements. Error bars show standard deviations.

training and noticed that they made incremental (in the third decimal place) improvements in testing loss for many testing periods before converging. We believe this to be due to very broad, flat regions near the minima of the loss landscapes of these networks. As discussed in the text, our training protocol may have terminated for these networks before they attained the minima of these plateaus. As a result, this may have interfered with the networks discovering general representations in the second and third layers. Similarly, if this coincided with co-adaptation among the second, third, and fourth layers, then some of the generality observed in the second and third layers at moderate widths may have been shared with the fourth layer in the under-converged very wide networks. In other words, the increase in the SVCCA specificities of the second and third layers at very large widths could be related to the slight decrease in the SVCCA specificity of the fourth layer in the same networks. Certainly, future work should explore this issue more carefully. At a practical level, a different training protocol than that used here might be beneficial for training very wide DENNs in performance-oriented settings.

Figure 2 summarizes the runtime performance of all the networks except those trained with transfer learning. Despite converging in fewer epochs, very wide networks were slowest to train, as they took the longest to train per epoch. Very small networks took longer to train because they took many more epochs to converge. Again, these behaviours might be of interest to future performance-oriented work.

B Additional analysis details

We used the SVCCA code provided by Raghu et al. [8] on the Google github repository. Their implementation included various threshold values used to remove small values from the data, as these are expected to correspond to noise. Because of the nature of DENNs, training is conducted with an unlimited amount of training data and without any noise in the data. As such, we did not use these thresholding operations.

The error bars of reproducibility and specificity in Figures 3 and 6 of the main text were obtained by treating the distributions of each of ρ_{self}^l , $\rho_{\Delta x'=0}^l$, and $\rho_{\Delta x'=X}^l$ as uncorrelated samples and applying standard rules for the propagation of uncertainty. In reality, these values are in fact somewhat correlated, so the error bars should be taken only as approximate uncertainties. However, since the reproducibility varies quite smoothly with network width and the specificity agrees quite well with the validation tests, we deem the metrics to be sufficiently well-resolved for the current work, and properly accounting for error correlations is relegated to future work.

C Additional visualizations of first layers

Figure 3 shows the first 9 components obtained by SVCCA with five different pairs of networks' first layers. All these networks had widths of 192. The numbers on the left indicate which networks were compared: x'_1 and x'_2 are the respective x' values on which they were trained, and s_1 and s_2 are their

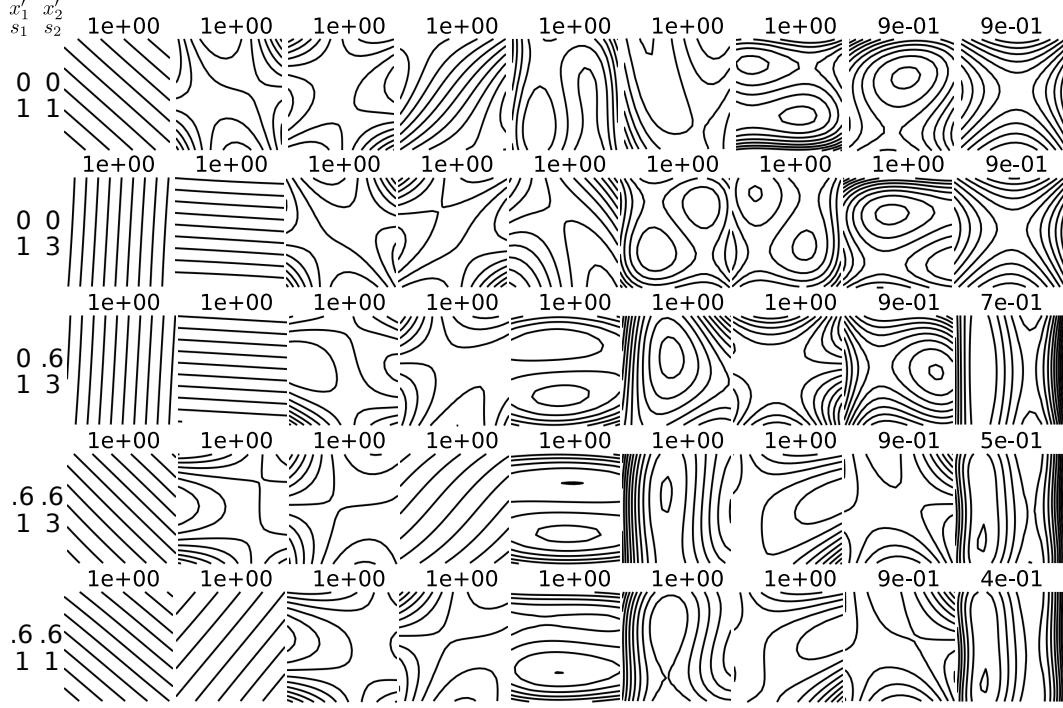


Figure 3: Each row shows plots of the first nine canonical components found by applying the SVCCA between the first layer of a network trained on $x' = x'_1$ and the first layer of a network trained on a different random seed at $x' = x'_2$, as indexed to the left of the plots. The number above each plot shows the correlation between the layers in the direction corresponding to that component.

respective random seeds. Thus the first and last rows show self-SVCCAs, which are equivalent to singular value decompositions. The numbers above each component show the canonical correlation computed by the SVCCA for that component. The last row of Figure 3 contains the same components shown in Figure 5 of the main text.

In all five cases, the leading 9 components have the same general structure. All rows contain a pair of components that capture the same information as the original inputs x and y :

1. In row 1, components 1 and 4, although component 4 is slightly distorted by mixing with another component.
2. In row 2, components 1 and 2.
3. In row 3, components 1 and 2.
4. In row 4, components 1 and 4, although component 4 is slightly distorted by mixing with another component.
5. In row 5, components 1 and 2.

The remaining components highlight the 9 regions of interest in the domain, as discussed in the main text. For instance, the top-left corner is present in the following components:

1. In row 1, components 3 and 8.
2. In row 2, component 4.
3. In row 3, component 4.
4. In row 4, components 2 and 3.
5. In row 5, components 3 and 4.

Figure 4 shows the first 27 components and their correlations for the layer shown in the first row of Figure 3. We note two things here. First, as mentioned in the first text, the correlation values drop

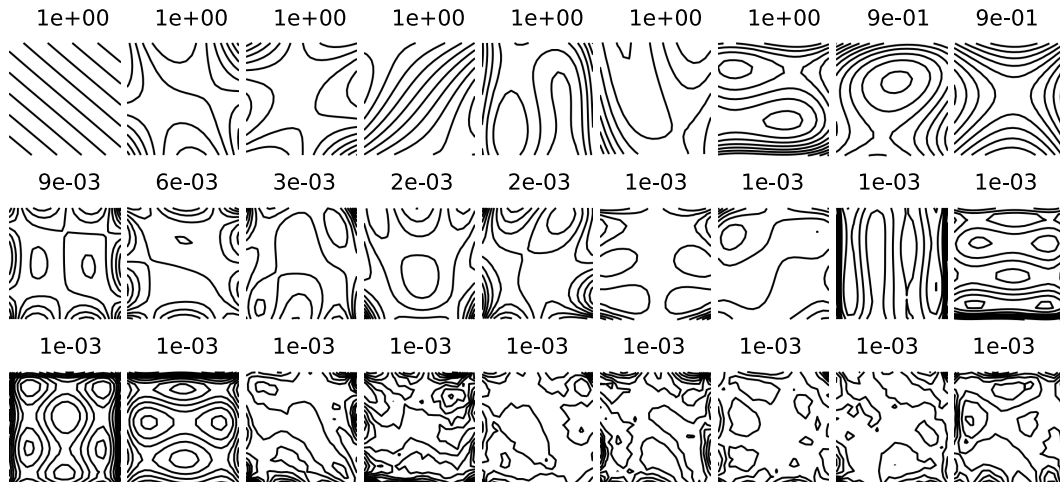


Figure 4: Plots of the first 27 components of the first layer shown in the first row of Figure 3.

drastically after the ninth component. This was the basis for our use of the self-SVCCA as a measure of intrinsic dimensionality. Second, we note that the 11 components following the first 9 still seem to capture coherent features over the input domain. Indeed, they appear analogous to higher-frequency Fourier modes found in spectral analysis. In contrast, components 21 and higher of the remaining 192 components are quite incoherent.

References

- [1] Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S. Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Ian Goodfellow, Andrew Harp, Geoffrey Irving, Michael Isard, Yangqing Jia, Rafal Jozefowicz, Lukasz Kaiser, Manjunath Kudlur, Josh Levenberg, Dan Mané, Rajat Monga, Sherry Moore, Derek Murray, Chris Olah, Mike Schuster, Jonathon Shlens, Benoit Steiner, Ilya Sutskever, Kunal Talwar, Paul Tucker, Vincent Vanhoucke, Vijay Vasudevan, Fernanda Viégas, Oriol Vinyals, Pete Warden, Martin Wattenberg, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng. TensorFlow: Large-scale machine learning on heterogeneous systems, 2015. URL <https://www.tensorflow.org/>. Software available from tensorflow.org.
- [2] Jens Berg and Kaj Nyström. A unified deep artificial neural network approach to partial differential equations in complex geometries. *arXiv preprint arXiv:1711.06464*, 2017.
- [3] Xavier Glorot and Yoshua Bengio. Understanding the difficulty of training deep feedforward neural networks. In *Proceedings of the thirteenth international conference on artificial intelligence and statistics*, pages 249–256, 2010.
- [4] Diederik Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.
- [5] Isaac E Lagaris, Aristidis Likas, and Dimitrios I Fotiadis. Artificial neural networks for solving ordinary and partial differential equations. *IEEE Transactions on Neural Networks*, 9(5):987–1000, 1998.
- [6] Isaac E Lagaris, Aristidis C Likas, and Dimitris G Papageorgiou. Neural-network methods for boundary value problems with irregular boundaries. *IEEE Transactions on Neural Networks*, 11(5):1041–1049, 2000.
- [7] Yann LeCun, Léon Bottou, Genevieve B Orr, and Klaus-Robert Müller. Efficient backprop. In *Neural networks: Tricks of the trade*, pages 9–50. Springer, 1998.
- [8] Maithra Raghu, Justin Gilmer, Jason Yosinski, and Jascha Sohl-Dickstein. SVCCA: Singular Vector Canonical Correlation Analysis for Deep Learning Dynamics and Interpretability. In *Advances in Neural Information Processing Systems*, pages 6078–6087, 2017.
- [9] Justin Sirignano and Konstantinos Spiliopoulos. DGM: A deep learning algorithm for solving partial differential equations. *arXiv preprint arXiv:1708.07469*, 2017.
- [10] Sho Sonoda and Noboru Murata. Neural network with unbounded activation functions is universal approximator. *Applied and Computational Harmonic Analysis*, 2015.

Chapter 6

On parallel computing for mobilities in periodic geometries

On Parallel Computing for Mobilities in Periodic Geometries examines a formulation of effective mobility (Sec. 2.3.1) in terms of a first passage problem. It extends the analysis from Chapter 3 to a much more general context, proving that the effective mobility of a biomolecule travelling through a periodic geometry can be expressed in terms of its first passage time through a single period of the geometry. The equivalence holds only for a specific choice of the initial conditions of the first passage problem.

This work was originally motivated by the results of *Studying First Passage Problems using Neural Networks: A Case Study in the Slit-Well Microfluidic Device*, attached in App. D and discussed below. Nagel, Andrew M. and Magill, Martin and de Haan, Hendrick W. [32] demonstrated an approach to studying periodic MNFDs using the neural network method to solve the parameterized time-integrated Smoluchowski equation. Their approach hinged on the assumption that the effective mobility of biomolecules through periodic

geometries could be expressed in terms of the first passage across a single period. The results *On Parallel Computing for Mobilities in Periodic Geometries* confirm that this is true and explain the technical requirements for correctly relating the method of Nagel, Andrew M. and Magill, Martin and de Haan, Hendrick W. [32] to real MNFDs.

Surprisingly, the first passage time formulation of the mobility was also found to have great promise for use with traditional particle simulations. *On Parallel Computing for Mobilities in Periodic Geometries* illustrates that, when ample parallel computing hardware is available, particle-based calculations of effective mobilities can be conducted far more efficiently using the first passage formulation. Given the rapid growth of parallel computing resources, this result may be of interest in its own right.

6.1 Motivation

As discussed in Sec. A.3, one of the great appeals of the neural network method (NNM) of solving partial differential equations (PDEs) is that it can readily be extended to solving parameterized PDEs. That is, whereas traditional numerical methods for PDEs typically produce solutions for one choice of model parameters at a time, the NNM can solve parameterized PDEs directly to obtain parameterized solutions. By comparison, using the finite element method (FEM), parameterized solutions to parameterized PDEs can be obtained using model order reduction (MOR) techniques (Sec. A.2.1.4). This is typically a two-stage process: first, a database of FEM solutions are produced at some reference choices of the parameter values; and second, the MOR techniques are trained on

this database to interpolate to new parameter values. The NNM approach essentially combines these two steps into one.

In *Studying First Passage Problems using Neural Networks: A Case Study in the Slit-Well Microfluidic Device* (App. D), Nagel, Andrew M. and Magill, Martin and de Haan, Hendrick W. [32] apply this method to a parameterized time-integrated Smoluchowski equation. The equation in question is a simplified model of the first passage of nanoparticles across a single period of the slit-well MNFD. For each choice of system parameters, the PDE describes a function $g_0(x, y)$, which is the time-integrated probability density function of nanoparticle center-of-mass position inside the slit-well device. The model was constructed in terms of two problem parameters: the applied field strength λ and the nanoparticle radius σ . The nanoparticles were modelled as hard spheres experiencing repulsive interactions with the walls of the slit-well device. As such, the parameter σ varied the volume accessible to the center-of-mass nanoparticle positions, and the domain geometry varied in shape and size as a function of σ .

Thus, the overall parameterized PDE described the function $g_0(x, y; \lambda, \sigma)$. As explained in Sec. 2.5, the solution g_0 to the time-integrated Smoluchowski equation has the property that the volume integral of g_0 over the PDE domain Ω is the mean first passage time τ from the initial position distribution (encoded in the source term of PDE) to the absorbing boundary condition of the model. For the specific model explored by Nagel, Andrew M. and Magill, Martin and de Haan, Hendrick W. [32], this was approximately the mean time taken by nanoparticles to cross a single period of the slit-well geometry.

Nagel, Andrew M. and Magill, Martin and de Haan, Hendrick W. [32]

demonstrated that the NNM could accurately approximate the four-dimensional function $g_0(x, y; \lambda, \sigma)$ over a physically relevant range of parameter choices. Implicitly, this function also encodes the dependence of τ on λ and σ , which can be computed by integration:

$$\tau(\lambda, \sigma) = \int_{\Omega} g_0(x, y; \lambda, \sigma) dx dy. \quad (6.1)$$

As demonstrated in their manuscript, this parameterized approximation to τ is valuable as a tool for visualizing and analysing the physical behaviour of the nanoparticles in the slit-well system. Moreover, because the standard NNM formulation produces continuously differentiable approximations to g_0 , the derivatives of τ with respect to λ and σ can be computed by integrating the appropriate derivatives of g_0 .

In contrast, traditional PDE solvers like FEM produce only point estimates of τ at fixed choices of λ and σ . Traditional MOR methods can also be used to learn $\tau(\lambda, \sigma)$, but these methods suffer from their own technical challenges. Most of these methods amount to approximating $g_0(x, y; \lambda, \sigma)$ by some linear function of the reference FEM solutions at fixed λ and σ . In the current problem, a major challenge to such an approach is presented by the parameter σ . Because changing σ modifies the domain of the PDE, it is not clear how to linearly combine FEM solutions obtained at different σ values in a meaningful way. In particular, it is not clear that it is possible to linearly combine them in such a way as to satisfy the boundary conditions for all intermediary values of σ .

Nagel, Andrew M. and Magill, Martin and de Haan, Hendrick W. [32] show that the basic NNM implementation is capable of handling the

geometry-modifying parameter σ without any need for algorithmic refinements. Moreover, by comparing the performance of the NNM when trained with fixed values of σ to its performance when trained with fixed values of λ , Nagel, Andrew M. and Magill, Martin and de Haan, Hendrick W. [32] conclude that the geometry-modifying parameter σ does not appear to present noticeably greater computational complexity than the parameter λ , which does not modify geometry. These results demonstrates that the NNM may be an appealing method for computational studies of first passage problems in domains with complicated geometries. More broadly, the implications are potentially relevant to any model where physical observables of interest can be expressed as functionals of the solutions to some parameterized PDEs.

In the specific context of researching and developing periodic MNFDs, however, the mean crossing time τ is typically not the physical observable of primary interest. More commonly, the effective mobility (Sec. 2.3.1), normally defined as

$$\mu_{\text{direct}} = \lim_{t \rightarrow \infty} \frac{\langle x/t \rangle_t}{\lambda}, \quad (6.2)$$

is of greater interest. Nagel, Andrew M. and Magill, Martin and de Haan, Hendrick W. [32] analyse instead the quantity

$$\mu_{\text{indirect}} \equiv \frac{L/\tau}{\lambda} \quad (6.3)$$

as a reasonable proxy for the effective mobility. Here L is the distance in x between the mean initial position of the nanoparticles and the absorbing boundary to which the mean first passage time τ is measured. The intuition

behind this definition is that L/τ is at least qualitatively similar to $\lim_{t \rightarrow \infty} \langle x/t \rangle_t$, as both describe some typical rate at which nanoparticles travel through the system. The names direct and indirect mobility are used in this chapter to distinguish between the two quantities. Nagel, Andrew M. and Magill, Martin and de Haan, Hendrick W. [32] found that the behaviour of μ_{indirect} was in general agreement with the known behaviour of effective mobility for nanoparticles in the slit-well (as studied by Cheng et al. [6]), and it provided a more interesting observable than τ for their analysis of the NNM's performance. The assumption was that, at worst, this quantity μ_{indirect} might serve as an acceptable alternative to μ_{direct} in periodic MNFD research.

The main motivation for the work in this chapter was, simply, to clarify the relationship between this quantity μ_{indirect} studied by Nagel, Andrew M. and Magill, Martin and de Haan, Hendrick W. [32] and the commonly accepted definition of the effective mobility μ_{direct} . As discussed in the next section, the analysis in the manuscript (Sec. 6.3) extends the derivations used in Chapter 3 to show that, in fact, the two equations can be made to match exactly. The subtle technical requirement is that the mean first passage time τ must be defined in the appropriate domain and with a specific initial condition. Similar results were available in a few related branches of the biophysical literature, but seemed to fall short of the generality and precision required to utilize this connection in the way proposed by Nagel, Andrew M. and Magill, Martin and de Haan, Hendrick W. [32]. *On Parallel Computing for Mobilities in Periodic Geometries* provides a firm foundation for extending the proof-of-concept technique of Nagel, Andrew M. and Magill, Martin and de Haan, Hendrick W. [32] to any molecules driven by

any fields through any geometry, with the only constraint being that the system must be Markovian on the timescale of transport across periods.

6.2 Results

6.2.1 Direct and indirect mobilities are equivalent

The main result of *On Parallel Computing for Mobilities in Periodic Geometries* is that the direct and indirect mobilities (Eqns 6.2 and 6.3) are, in fact, describing the same physical observable. This equivalence is, however, conditional. The indirect mobility must be computed using a particular choice of initial conditions. The details are explained in the manuscript; essentially, the initial distribution of molecular degrees of freedom must be chosen such that the resulting final distribution (at first contact with the next period) is itself equal to the initial distribution.

This self-consistency condition corresponds to the stationary distribution, $\pi(\phi)$, of an appropriately defined Markov chain model, $\rho(\phi_i) \rightarrow \rho(\phi_{i+1})$. It is argued in the manuscript that this stationary distribution should almost certainly exist for most realistic biophysical systems. Moreover, it should also be unique, and repeated calculations of the mapping $\rho(\phi_i) \rightarrow \rho(\phi_{i+1})$ should converge exponentially fast to $\pi(\phi)$ for any initial choice of $\rho(\phi_0)$. Numerical analyses are provided to illustrate that this convergence is indeed extremely rapid, at least in the case of nanoparticles traversing the slit-well MNFD. This enables efficient sampling of the stationary distribution using a Markov chain Monte Carlo method.

6.2.2 Indirect mobility better exploits parallel computing

After proving the equivalence of the indirect and direct mobilities, *On Parallel Computing for Mobilities in Periodic Geometries* had essentially accomplished its initial purpose of clarifying the modelling requirements for connecting the NNM-based methodology of Nagel, Andrew M. and Magill, Martin and de Haan, Hendrick W. [32] to the study of arbitrary periodic MNFD systems. However, a second interesting result emerged as a by-product of that derivation. The analysis that underpins the proof of equivalence allows one to compare the computational cost of estimating the direct and indirect mobilities using particle simulations.

Although a couple of simplifying approximations are made in the theoretical comparison of the two mobilities, the analysis uncovers one unambiguous conclusion: Given access to a sufficient amount of parallel computing hardware, simulation-based estimates of the indirect mobility converge exponentially faster than estimates of the direct mobility based on the same particle simulations. When access to parallel hardware is restricted, the advantage of the indirect mobility formulation is most pronounced at moderate levels of target accuracy. When very high levels of accuracy are required, the analysis suggests the two algorithms are roughly equivalent.

In practice, moderate levels of accuracy are often all that is required in computational nanobiophysics. The complexity of biophysical systems means that most mathematical models studied numerically correspond to modelling errors greater than 1%, so numerical solutions of comparable accuracy are sufficient to guide research and development efforts. Furthermore, the availability of parallel computing hardware is rapidly increasing. A single consumer-grade GPU can

readily provide tens of thousands of parallel threads for molecular dynamics simulations, and accessing multiple GPUs for scientific computing is increasingly feasible. The simplified theoretical analysis in the manuscript suggests that, under such conditions, the indirect mobility can be substantially more efficient than the direct mobility.

Ultimately, a complete prediction of the two algorithms' computational performance essentially requires knowledge of the entire time-dependent transport of molecules through the MNFD system in question. Thus, we turn once more to the example of nanoparticles in the slit-well device to provide numerical measurements of their behaviour. As confirmed in the manuscript, the theoretical analyses of error versus runtime for both algorithms match reality quite well in scenarios with large Péclet numbers (i.e., where the electric field is strong relative to the diffusive motion of the particles). In this case, as predicted, the indirect mobility estimates can converge to relative errors below 1% up to 5-6 times faster when a single GPU is utilized; with 10 GPUs, the indirect mobility converges 10-20 times faster down to accuracies on the order of 0.1%.

However, the numerical analyses of the algorithms were found to break down in the regime of low Péclet number. Here, diffusive effects were found to greatly accelerate the convergence of the direct mobility, likely because the system converges more rapidly to its steady-state behaviour. Conversely, the runtime of the indirect mobility becomes dominated by the very large first passage times of particles that diffuse backwards through several periods before absorption. As a result, the indirect mobility in fact converges much more slowly than the direct mobility for low Péclet numbers and access to only a single GPU. This holds until

fairly low errors (e.g., on the order of 0.1% in one example of nanoparticles in the slit-well), below which the two algorithms are essentially equivalent again. On the other hand, given access to sufficient parallelism the indirect mobility recovers the advantage. In the slit-well example, the indirect mobility is roughly an order of magnitude more efficient than the direct mobility for target errors below roughly 0.5% even for very small Péclet numbers (see Fig. 9 in the manuscript and the corresponding text). This acceleration corresponds to relatively modest hardware; current top-of-the-line consumer-grade GPUs should favour the indirect mobility even more significantly, as discussed in the manuscript.

The above discussion neglects a crucial component of the computational cost of the indirect mobility: the calculation of the stationary distribution. Luckily, it appears that this cost may be quite minimal for many periodic MNFDs of interest. In the slit-well example, the stationary distribution was found to be essentially uniform when the delineation between periods of the device was placed in the middle of the slits. This was conjectured to occur because of the geometric bottleneck in the system near these points. To investigate this, the system was shifted by half a period, so that the threshold between periods was located in the middle of the wells. Indeed, the stationary distribution became unambiguously non-uniform. Even still, relaxing the distribution by computing trajectories through a few periods of the device was found to converge to the stationary distribution very rapidly. In fact, even relaxation through a single period appeared to eliminate virtually all error in the mobility due to the the initial conditions.

In practice, many periodic MNFDs do exhibit geometric bottlenecks; in this

case, the stationary distribution may turn out to be approximately equal to a very simple distribution, trajectory-based sampling may be found to converge very quickly, and/or the error in mobility measurements due to imperfect sampling of the stationary distribution may turn out to be small. These conclusions are certainly system dependent, and this will hopefully be considered by future work. In any case, for the example of nanoparticles traversing the slit-well, properly sampling the stationary distribution increases the total computational cost of the indirect mobility by at most a factor of 2 or 3. As discussed in the manuscript, this cost can be reduced even further by recycling samples (i.e., generating only 100,000 samples of the stationary distribution and re-sampling these to initialize 10,000,000 simulated trajectories); this greatly reduces the relative cost of sampling, and does not appear to dominate the error at the 0.1-1% range.

6.3 Manuscript

On Parallel Computing for Mobilities in Periodic Geometries

Martin Magill, Andrew M. Nagel, and Hendrick W. de Haan^{a)}

Faculty of Science, University of Ontario Institute of Technology, 2000 Simcoe St N, Oshawa, Ontario L1H7K4, Canada

(Dated: 2 June 2022)

We examine methods for calculating the effective mobilities of molecules driven through periodic geometries in the context of particle-based simulation. The standard formulation of the mobility, based on the long-time limit of the mean drift velocity, is compared to a formulation based on the mean first passage time of molecules crossing a single period of the system geometry. The equivalence of the two definitions is derived under weaker assumptions than similar conclusions obtained previously, requiring only that the state of the system at subsequent period crossings satisfy the Markov property. Approximate theoretical analyses of the computational costs of estimating each of these two mobility formulations via particle simulations suggest that the definition based on first passage times may be substantially better suited to exploiting parallel computation hardware. This claim is investigated numerically on an example system modelling the passage of nanoparticles through the slit-well device. In this case, the traditional mobility formulation is found to perform best when the Péclet number is small, whereas the mean first passage time formulation is found to converge much more quickly when the Péclet number is moderate or large. The results suggest that, given fairly modest access to modern GPU hardware, this alternative mobility formulation may be an order of magnitude faster than the standard technique for computing effective mobilities of biomolecules through periodic geometries.

I. INTRODUCTION

Microfluidic and nanofluidic devices (MNFDs) are an emerging class of biotechnologies with various promising applications in the biological and medical sciences^{1–4}. Of these devices, an important subclass of periodic MNFDs exploits the motion of molecules driven through a periodic arrangement of geometric features (e.g., by an electric field) to induce separation by size or other chemical properties. For instance, some of the first MNFDs used for biomolecular separation consisted of periodic arrays of micron-scale posts⁵, and work on this type of MNFDs remains an area of active research and development^{6–13}. Variants of the post-array design with asymmetric obstacle shapes form the basis of so-called Brownian ratchet devices^{14–17}. The slit-well motif is another important MNFD design, consisting of a planar confinement with alternating deeper well regions and shallower slit regions. First pioneered by Han and Craighead¹⁸ and elaborated upon in a series of subsequent studies^{19–21}, the slit-well device has stimulated ongoing research interest^{22–30}. A related MNFD design, the capillary-well motif, has been developed more recently^{31,32}. The dynamics of biomolecules have also been studied experimentally, theoretically, and numerically in a variety of other periodic geometries, such as: one-, two-, or three-dimensional arrays of spherical cavities^{33–37}, channels with periodic bands of attractive and repulsive zones on their walls^{38–41}, a network of interconnected channels named the railroad switch motif⁴², planar confinement with an array of nanopits⁴³, a series of nanopores connected by

microchannels⁴⁴, and a periodic sheet of graphene alternating with boron nitride⁴⁵. In fact, even MNFDs whose geometries are uniform in the net direction of motion, such as those used for microcapillary hydrodynamic chromatography^{46–49}, are (trivially) periodic in this direction.

When periodic MNFDs are used to separate molecules according to size or some other chemical property of interest, this is accomplished by coupling that property to the molecule’s net speed through the device. Specifically, the transport rate of analytes is usually characterized by the effective mobility, which is the mean velocity on long timescales normalized by the magnitude of the applied force-generating field. In fact, many molecular mixtures of interest (e.g., DNA, nanoparticles, etc.) exhibit little to no variation in mobility when driven through free solution as the net force and net friction scale in direct proportion to one another. To enable molecular separation, MNFDs break this symmetry by exploiting the interplay of drift and diffusion in non-trivial geometries.

In practice, the design of such devices can be challenging. One aims to control the coupling between mobility and molecular characteristics by optimizing design parameters (such as applied voltage or pressure, solvent composition, and device geometry) in order to produce the desired profile of effective mobilities. Simulations are often a valuable aid in elucidating the influence of the many design parameters on molecular transport properties such as mobility.

The most common definition of mobility is

$$\mu_{\text{direct}} = \lim_{t \rightarrow \infty} \frac{\langle x(t) \rangle / t}{\Phi}, \quad (1)$$

where $\langle x(t) \rangle$ is the ensemble mean at time t of the center-of-mass position in the net direction of motion. Here Φ

^{a)}Electronic mail: Hendrick.deHaan@uoit.ca

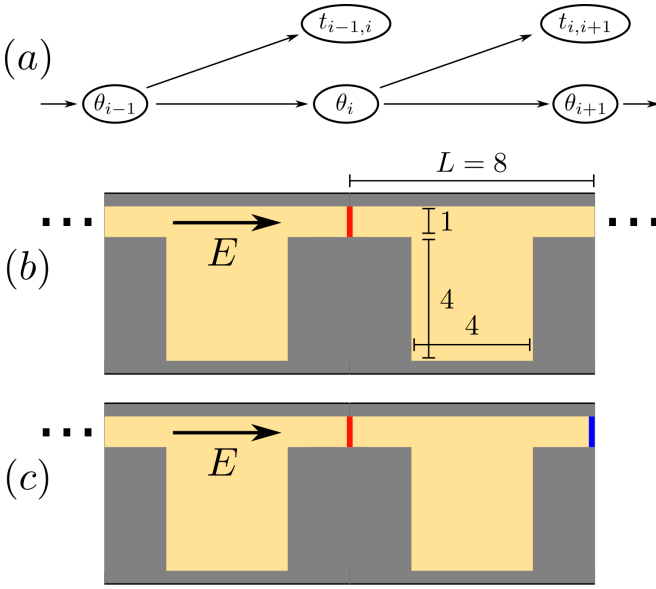


FIG. 1. Schematic illustrating the model of nanoparticles in the slit-well device. (a) A probabilistic graphical model of the Markov chain model from App. B. For the problem described in Sec. II B, the only auxiliary coordinate is $\theta_i = y_i$, the y coordinate of the nanoparticle at its first passage to each new period. The distributions of y_{i+1} and $t_{i,i+1}$ are each determined entirely by y_i . (b) In the direct mobility formulation, particles are initialized at the midpoint of a slit (red line) and evolve forward in time until many periods have been crossed. (c) In the indirect mobility formulation, particles are initialized at the midpoint of a slit (red line) and evolve forward in time until a single period has been crossed (blue line). The arrow indicates the direction of the applied electric field. Diffusive motion against the direction of E is possible in both mobility formulations.

is a scalar characterising the magnitude of the field that is generating the force driving molecular motion. The choice of Φ is context-dependent: for electrically driven motion, $\Phi \sim \Delta V$ must characterize the gradient of the applied electric potential V (see, e.g., Cheng et al.²³); for pressure-driven motion, $\Phi \sim \Delta p$ should indicate the gradient of the applied pressure p (see, e.g., Ollila et al.²⁷); and so on. Equation 1 will be referred to here as the direct mobility, because it is defined directly in terms of the physical observable it is used to study: the long-term drift velocity of molecules through the system.

In contrast, the focus of this work is the quantity defined as

$$\mu_{\text{indirect}} = \frac{L}{\Phi \langle \tau_1 \rangle}, \quad (2)$$

where L is the period of the system geometry and $\langle \tau_1 \rangle$ denotes the ensemble mean of the first passage time across one such period. Equation 2 will be referred to here as the indirect mobility, because it is formulated in terms of observables that can be measured without directly examining the long-term motion across many periods. This

manuscript includes a careful comparison of the direct and indirect mobility formulations.

In fact, the two mobility definitions are equivalent under certain circumstances. Indeed, several classical theoretical frameworks imply that $\lim_{t \rightarrow \infty} \langle x(t) \rangle / t = L / \langle \tau_1 \rangle$. However, as reviewed in App. A, these results are derived under fairly strong assumptions. Fick-Jacobs theory (App. A 1) assumes that motion can be reduced to an effective one-dimensional system, and this approach is typically limited to weakly driven motion and/or slowly varying geometries. Kramers theory (App. A 2) and related reaction rate theories assume that most degrees of freedom of the system relax very quickly on the timescale over which the molecule traverses the distance L along the device.

More generally, one can argue for the equivalence of the two mobility definitions on the basis of ergodicity. A single molecule that has crossed a large number k of periods at time t will have sampled the crossing time for a single period k times. The long-time mean of this one particle's k crossing times will be, by ergodicity, equal to the ensemble mean of the time to cross a single period, so that $t \approx k \langle \tau_1 \rangle$. Conversely, its position will be roughly $x \approx kL$, since it has crossed k periods. Thus, its mean velocity will be

$$\frac{x}{t} \approx \frac{kL}{k \langle \tau_1 \rangle} = \frac{L}{\langle \tau_1 \rangle}, \quad (3)$$

from which the equivalence of Eqns. 1 and 2 follow.

A more detailed derivation of this result is included in App. B and demonstrated numerically on a test problem in Sec. III A. The equivalence of the two definitions is proven under the simple hypothesis that the system satisfies the Markov property on the timescale of crossing from one period to the next. Specifically, the dynamics of the analyte between the time it first enters the k th period and the time it first enters period $k+1$ are assumed to depend only on the state of the analyte at the moment that it first entered period k . Under this assumption, the limiting form for the ensemble distribution of x positions can be deduced in closed form. Correlations between the crossing times in consecutive periods are properly taken into account, and these are seen to have a direct effect on the effective diffusion coefficient of the analytes on long timescales.

The equivalence of direct and indirect mobility depends on one important technical requirement, which is that the mean first passage time $\langle \tau_1 \rangle$ in Eqn. 2 must be defined with respect to a certain stationary distribution. It is argued in App. B 2 that under typical conditions this distribution should exist and be unique. Moreover, a simple Markov chain Monte Carlo algorithm for estimating this distribution numerically is described in App. C and tested in Sec. III B.

The limiting behaviour of the transport dynamics deduced in App. B enables an approximate convergence rate analysis of the two mobility formulations, included in App. C. It appears that the indirect mobility is better

suit for exploiting the massive parallelization afforded by modern hardware. Given reasonable access to such hardware, the analysis suggests that standard computational studies of mobilities through periodic geometries (such as those conducted in Refs. 8,13,22,23,27,37,38,40,41,43,45,49) may be made to converge up to an order of magnitude more quickly with very little modification to the underlying simulation algorithms. The computational advantage of the indirect mobility in the test case from Sec. II B is verified numerically in Sec. III B.

II. PROBLEM DEFINITION

A. The general case: Transport through a periodic geometry

The physical systems under consideration are those in which a single molecule is driven through a periodic MNFD. The molecular motion is stochastic, such that the physical observables of interest are ensemble averages. Some external force field (generated, e.g., by an applied voltage) biases the stochastic motion of the molecule. The mean direction of the molecule's center-of-mass motion over long timescales will be called the \hat{x} direction. The geometry of the periodic MNFD and the external force field are both taken to be periodic in the direction of \hat{x} , with a period of length L . Every interval of length L in the \hat{x} direction will be called one period of the device.

The molecule travelling through the system will be represented by a finite number of degrees of freedom N , which together specify all information about the state of the system. These would typically be the positions of all the atoms in the molecule. In the event of a time-periodic force field, the phase of the molecule with respect to the period of the force field should also be considered an auxiliary coordinate. In particular, we make the assumption that the dynamics of these N degrees of freedom are well-approximated as Markovian, at least on the timescale over which the molecule crosses a period of the device. As reviewed, for instance, by Hänggi et al.⁵⁰, coarse-grained representations can exhibit non-Markovian dynamics (i.e. memory) even when the underlying system is actually Markovian at the finest scale. Nonetheless, for the models commonly used to study periodic MNFDs, this Markovian assumption is either exactly true or a good approximation on the timescales of interest^{8,13,22,23,27,37,38,40,41,43,45,49}.

Of the degrees of freedom, the position of the center of mass in the \hat{x} direction at time t will be denoted by the random variable x or $x(t)$ to make the time dependency explicit. The remaining degrees of freedom will be called auxiliary coordinates, and denoted collectively by the random vector θ or $\theta(t)$. The original value of x will be fixed to some $x_0 = 0$ at time $t = 0$, and the thresholds between consecutive periods will be located at $x_i = x_0 + iL$ for each i . The time elapsed between the molecule's first arrival at x_i and its subsequent first arrival at x_{i+1}

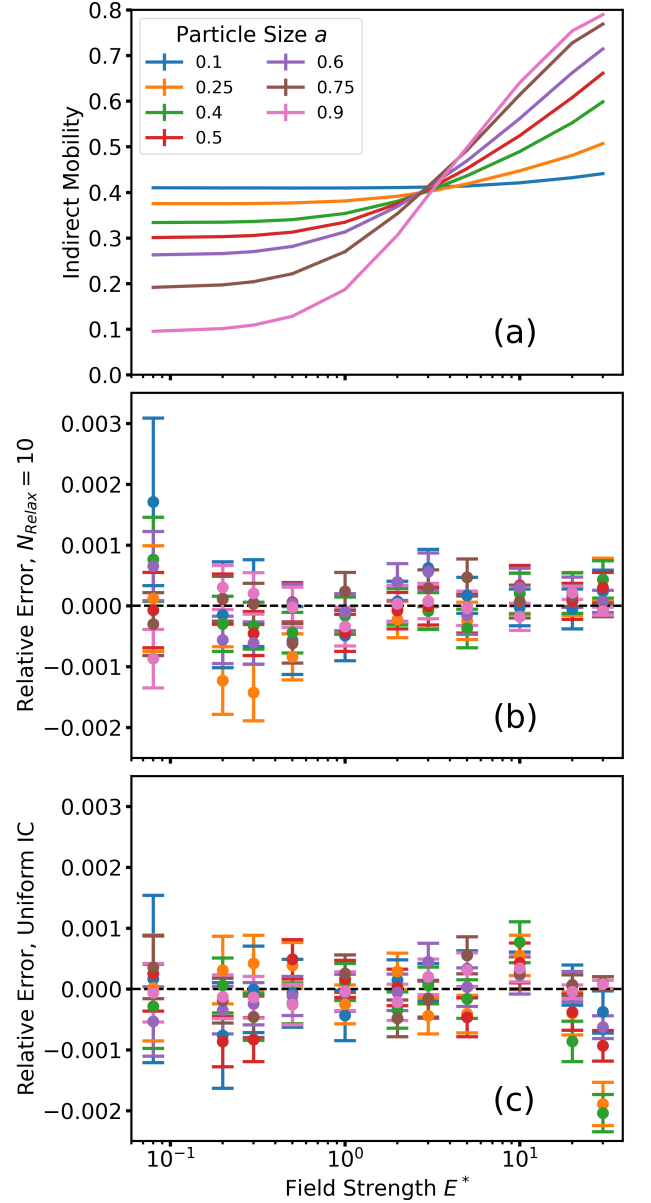


FIG. 2. (a) Measured indirect mobility values as a function of field strength E^* for various particle diameters a . (b,c) Relative error of simulated indirect mobility values compared to simulated direct mobility values when indirect mobilities are calculated using initial x values in the middle of a slit and (b) initial y values sampled using MCMC with $N_{\text{relax}} = 10$ or (c) initial y values distributed uniformly. Error bars correspond to one standard error.

is denoted $t_{i,i+1}$. The value of the auxiliary coordinates at the moment of first contact with x_i will be denoted θ_i .

The proof in App. B is based on a Markov chain model of this general physical scenario. If the underlying molecular dynamics are Markovian, then the states of the system at first contact with each x_i form a Markov chain. The Markov chain is written in Eqn. B2 and illustrated

in Fig 1(a) as a probabilistic graphical model (i.e., arrows show statistical dependencies). In particular, the value of the auxiliary coordinates θ_i at first contact with x_i entirely determines the distributions of the first passage time $t_{i,i+1}$ and of the auxiliary coordinates θ_{i+1} at first passage into the next period.

Based on an application of the Markov chain central limit theorem to this Markov chain, the limiting form of the position distribution $\rho(x(t))$ at long times t can be deduced. From there, the equivalence of the direct (Eqn. 1) and indirect (Eqn. 2) mobility formulations follows readily. This use of the Markov chain central limit, however, requires that the initial auxiliary coordinates θ_0 be initialized according to a specific initial distribution. In particular, the initial distribution of auxiliary coordinates $\rho(\theta_0)$ must be chosen such that the auxiliary coordinates θ_1 measured at the first passage threshold x_1 are distributed according to the same distribution: $\rho(\theta_1) = \rho(\theta_0)$. This choice corresponds to the stationary distribution of the Markov chain in Eqn. B2; Sec. B2 includes discussion regarding its existence and uniqueness.

In principle, the choice of the initial position x_0 in the above description is arbitrary and should have no impact on the mobility. In practice, however, shifting x_0 can have subtle but important consequences for the numerical determination of the indirect mobility, as will be shown in Sec. III B. In particular, x_0 affects the nature of the stationary distribution, and thereby controls the computational cost of sampling the initial values of θ for the indirect mobility calculation.

B. Guiding example: Particles in the slit-well device

As a specific illustration of the general circumstance described in Sec. II A, this section presents a model of free-draining nanoparticles traversing the slit-well MNFD under the influence of an applied electric force^{18–30}. In particular, we will study the same model analysed by Cheng et al.²³. Whereas the slit-well has primarily been studied in the context of polymer analytes (and especially DNA), we will focus on the simpler case of nanoparticle mobilities as it facilitates a more comprehensive numerical exploration. The equivalence of the direct and indirect mobilities is demonstrated numerically for this system in Sec. III A, the task of sampling the correct stationary distribution is explored in Sec. III B, and the computational advantages of the indirect mobility are illustrated in Sec. III C.

The geometry of the system is illustrated in Fig. 1(b,c). The dimensions are indicated in Fig. 1(b); the period length $L = 8$ and the aspect ratios of the slit and well regions are set to match Cheng et al.²³. The nanoparticles are modelled as hard spherical particles of diameter a having only two degrees of freedom: the x and y coordinates of their centers of mass. The z coordinate of the center of mass is omitted under the assumption of symmetry in the z direction, and rotational degrees of

freedom are also assumed to be negligible. The applied force field will be held constant in time, so that the only auxiliary coordinate in this case is $\theta = y$.

Particle motion will be governed by Brownian dynamics, i.e., the overdamped Langevin equation

$$\frac{d\vec{x}}{dt} = -\mu_0\lambda\nabla U + \sqrt{2D}R(t), \quad (4)$$

where $\vec{x} = (x, y)$, μ_0 is the free-solution mobility of the nanoparticles, D is the free-solution diffusion coefficient, R is a stationary delta-correlated stochastic force with mean 0 and variance 1, λ is a scalar controlling the magnitude of the applied force, and U is the baseline electrostatic potential energy of the particle. The free-solution mobilities will be fixed at $\mu_0 = 1$, while the free-solution diffusion coefficients will scale as $D = 1/a$ in line with Stokes' law. The walls are treated as purely reflective conditions applied when the center of the nanoparticles is a distance $a/2$ from the nominal dimensions listed in Fig. 1(b). Hydrodynamic effects are neglected, and electrohydrodynamic phenomena (such as the particle's charge and ζ potential) are subsumed into μ_0 .

The baseline electrostatic potential in Eqn. 4 is modelled simply by Laplace's equation,

$$\nabla^2 U = 0. \quad (5)$$

The walls of the slit-well device are treated as perfectly insulating boundary conditions. A unit voltage drop is imposed across one period of the device measured from the middle of two consecutive slits. By linearity, the field $-\lambda\nabla U$ corresponds to an applied voltage drop of λ per period. To match Cheng et al.²³, we define the quantity $E^* = \lambda/L$ as the characteristic field strength. This characteristic field strength E^* is also the correct choice of Φ for computing mobilities in this system (i.e., in Eqns. 1, 2). Finally, it will also be useful to discuss the behaviour of the system in terms of the Péclet number $\text{Pé} = E^*a$, which is proportional to the drift-diffusion ratio in the system.

The direct and indirect mobilities for this model system were computed using particle simulations under a variety of conditions. Equation 4 was discretized using the common Euler-Maruyama scheme⁵¹ to

$$\vec{x}(t_{j+1}) = \vec{x}(t_j) - \mu_0\lambda\nabla U(\vec{x}(t_j))\Delta t + \sqrt{2D\Delta t}R_j. \quad (6)$$

Here $\vec{x}(t_j)$ is the position of the particle at time t_j , each R_j is an independent standard normal random variable drawn at each timestep, and Δt is the discrete timestep. A value of $\Delta t = 10^{-3}$ was used for all simulations. The baseline electrostatic potential U and the corresponding baseline field $-\nabla U$ were approximated using a mixed finite element method formulation according to the methodology described in Nagel et al.⁵².

Particle simulations were always initialized with constant x positions but random distributed initial y values. The initial x value was generally placed in the middle of a slit (Fig. 1(b,c)). For direct mobility calculations, the

initial values of y were uniformly distributed. For indirect mobility calculations, the initial values of y were sampled from a precomputed database of 10^5 samples obtained by Markov chain Monte Carlo (MCMC). Specifically, these samples correspond to the final y positions of trajectories initialized with uniform y positions and simulated until a total of $N_{\text{relax}} = 10$ periods were crossed; this ensemble of samples from the stationary distribution was computed once for each choice of E^* and a and reused for all corresponding simulations. A total of 10^7 trajectories were used for indirect mobility calculations. Direct mobilities were computed by simulating 10^4 trajectories until the mean position in the \hat{x} direction exceeded $5000L$, which is roughly the same methodology used by Cheng et al.²³. The above is the default simulation protocol throughout this paper, but variations of some of these parameters are investigated in Sec. III.

III. NUMERICAL DEMONSTRATIONS

A. Equivalence of direct and indirect mobilities

Figure 2(a,b) provides numerical verification that the indirect and direct mobilities are equivalent for the model of free-draining nanoparticles of diameter a through the slit-well MNFD described in Sec. II B. Figure 2(a) shows the indirect mobility as a function of the normalized field strength E^* for nanoparticles of various sizes a . This can be compared directly to Fig. 2 of Cheng et al.²³, where the same measurements were computed using a direct mobility formulation. The direct mobility measurements were reproduced for the present work as well, but they are not shown as they are visually indistinguishable from the indirect mobilities in Fig. 2(a). Rather, Fig. 2(b) shows the relative error of the indirect mobility values relative to the direct mobility measurements. The indirect and direct mobilities were all computed using the default simulation protocol described in Sec. II B. As anticipated, the direct and indirect mobilities are in excellent agreement for all cases in Fig. 2(a,b): the relative errors are all of the order of 0.1% or better, and all points lie within two standard errors of 0.

B. Sampling the stationary distribution

The results in Fig. 2(a,b) are based on indirect mobilities calculated by sampling the stationary distribution for y using the MCMC protocol described in Sec. II B. For each choice of E^* and a , particle trajectories are evolved until they traverse $N_{\text{relax}} = 10$ periods, and their final y values form initial conditions for subsequent indirect mobility estimations. That simulation protocol appears to be sufficient to recover approximate equivalence of the direct and indirect mobilities. However, because the simulations used for the MCMC algorithm are essentially

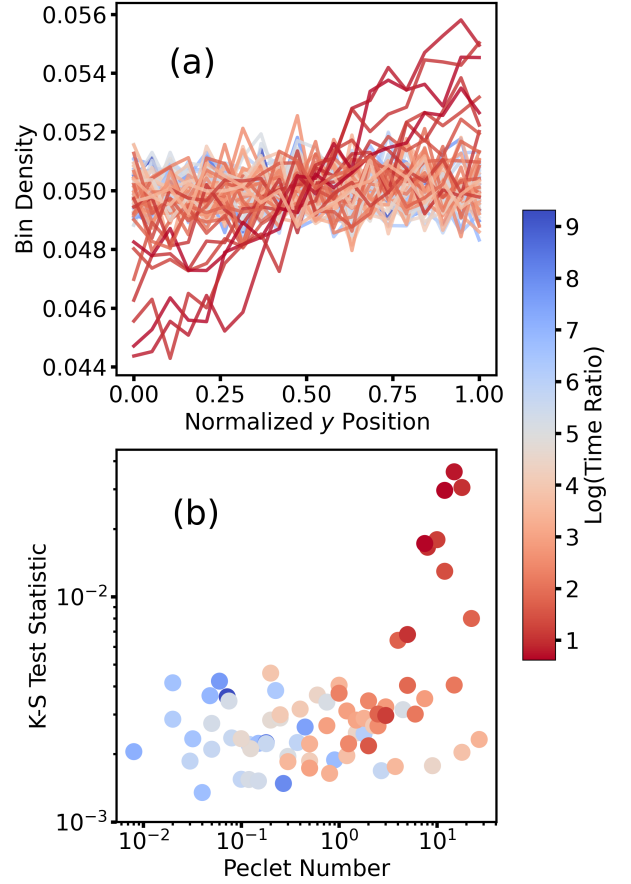


FIG. 3. (a) Normalized histograms (with 20 bins) of the stationary distributions obtained with $N_{\text{relax}} = 10$ for the standard protocol. The normalized y position spans the available y coordinates in the slit, which depends on a through the reflective boundary conditions. (b) Kolmogorov-Smirnov test statistic between the sampled distributions and the uniform distribution (higher value indicates less uniform behaviour), as a function of Péclet number. Colours in (a) and (b) show $\log(\tau_{\text{drift,slit}}/\tau_{\text{diff,slit}})$ as described in the text.

identical to the simulations used to measure both the direct and indirect mobilities, the MCMC algorithm nominally multiplies the computational cost of the indirect mobility calculation by a factor of N_{relax} . For excessively large values of N_{relax} , any computational advantage of the indirect mobility will be lost.

Luckily, it appears to be possible to dramatically reduce this overhead cost. For example, the protocol in Sec. II B reduces this cost by recycling 10^5 MCMC samples across the 10^7 trajectories used for the indirect mobility calculation. This reduced the runtime of the MCMC algorithm by roughly a factor of 100, rendering it a negligible fraction of the total runtime. The magnitude of the error imparted by this technique will depend on the details of the system being studied. Systems for which the true stationary distribution is more intricate and/or for which the first passage time depends strongly

on the initial values of the auxiliary coordinates should incur more error from recycling MCMC samples.

In the current system, however, the stationary distribution for almost all of the physical parameter combinations was found to be very nearly uniform. Figure 3(a) shows histograms of the sampled stationary distributions for all values of E^* and a . The lines are coloured according to a ratio of drift and diffusion times described below. It is clear that most cases are nearly uniform, and even the few that deviate noticeably from uniform do not deviate very much in absolute terms. Figure 3(b) shows the Kolmogorov-Smirnov test statistics of these distributions with respect to the uniform distribution. This is the maximum distance between the empirical cumulative distribution of the MCMC samples of y against the cumulative distribution of a uniform distribution; essentially, a larger value indicates that the samples likely come from a more non-uniform distribution. Here, it is clear that although all of the most non-uniform distributions correspond to large Péclet numbers, not all cases with large Péclet numbers exhibit significantly non-uniform stationary distributions.

In fact, this behaviour is not so surprising. The slits of the slit-well device are fairly narrow and long. The local dynamics within each slit likely satisfy the condition assumed in Fick-Jacobs theory (App. A 1): diffusive relaxation of y coordinates occurs much more quickly than translation in the x direction. This explains why not all cases with large Péclet numbers in Fig. 3 exhibit large Kolmogorov-Smirnov test statistics relative to the uniform distribution: the Péclet number $Pé = E^*a$ is a global Péclet number, and does not account for the local drift-diffusion ratio within the slit, which is more strongly affected by the excluded volume effects due to the particle diameter a .

A local measure of the drift-diffusion balance can be obtained by comparing the drift time along the slit in the x direction,

$$\tau_{\text{drift,slit}} \sim \frac{L/2}{\mu_0 E^*}, \quad (7)$$

to the diffusion time across the slit in the y direction,

$$\tau_{\text{diff,slit}} \sim \frac{(\ell_{\text{slit}} - a)^2}{2D} \sim \frac{a(\ell_{\text{slit}} - a)^2}{2}. \quad (8)$$

Here $L = 8$ is the period length, $\ell_{\text{slit}} = 1$ is the nominal width of the slit, a is the particle diameter, and $D = 1/a$ is the particle diffusion coefficient. The plots in Fig. 3(a,b) are coloured in proportion to the logarithm of the ratio of these times, $\log(\tau_{\text{drift,slit}}/\tau_{\text{diff,slit}})$. A large ratio corresponds to the Fick-Jacobs regime. Figure 3(a,b) clearly shows that the most non-uniform stationary distributions are those for which the global Péclet number is high and the drift-diffusion time ratio in the slit is smallest.

In fact, it is even feasible in this case to forego the MCMC sampling step altogether. Figure 2(c) shows relative errors of mobility measurements made with $N_{\text{relax}} =$

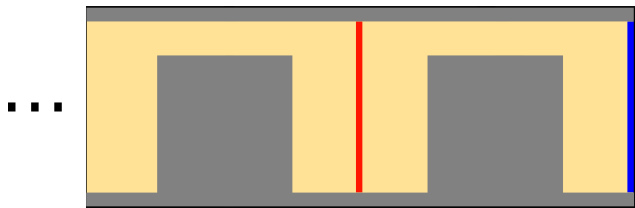


FIG. 4. Schematic of the well-to-well configuration for indirect mobility measurements. Particles are initialized on the red line, which is in the middle of a well. Mean first passage times are computed to the blue line.

0, i.e., a uniform distribution of initial y values. The error relative to the calculated direct mobilities is statistically indistinguishable at all but the largest field strengths. More specifically, as predicted above based on the ratio of drift to diffusion times in the slit, only for very large values of E^* and moderate values of a is the error from using a uniform initialization statistically significant in Fig. 2(c). Even at these choices of E^* and a , the indirect mobility still only has errors on the order of 0.02%; the true stationary distributions are still nearly uniform (Fig. 3(a)). Extending the above reasoning, nearly-uniform stationary distributions may be expected to arise in other periodic MNFDs featuring geometric bottlenecks.

It is nevertheless important to note that the use of the correct stationary distribution is indeed an essential condition for the equivalence of the direct and indirect mobilities (App. B). Figure 5 illustrates the consequences that can arise if this condition is neglected inappropriately. In this case, indirect mobilities were once again measured using uniform initial conditions for y , but now with the mean first passage time computed from an initial x position set in the middle of a well to the middle of the next well (Fig. 4), rather than from the middle of a slit to the middle of the next slit (Fig. 1(c)).

Figure 5(a) shows the indirect mobilities computed based on the well-to-well mean first passage process with uniform initial conditions, and Figure 5(b) shows the corresponding relative errors. At low field strengths, this algorithm still produces acceptably small relative errors. However, at higher field strengths the indirect mobilities are entirely incorrect, both quantitatively and qualitatively. This is in stark contrast to the results of Fig. 2(c), which showed that uniform initial conditions were an acceptable approximation for all cases in the slit-to-slit configuration.

Indeed, the correct stationary distribution in the well-to-well configuration is substantially non-uniform. Figure 6(a) shows all the distributions for the well-to-well configuration measured with $N_{\text{relax}} = 1$, and Fig. 6(b) shows the corresponding Kolmogorov-Smirnov test statistics relative to the uniform distribution. As in Fig. 3, colours are based on the ratio of the drift timescale to the diffusion timescale; in the well-to-well configura-

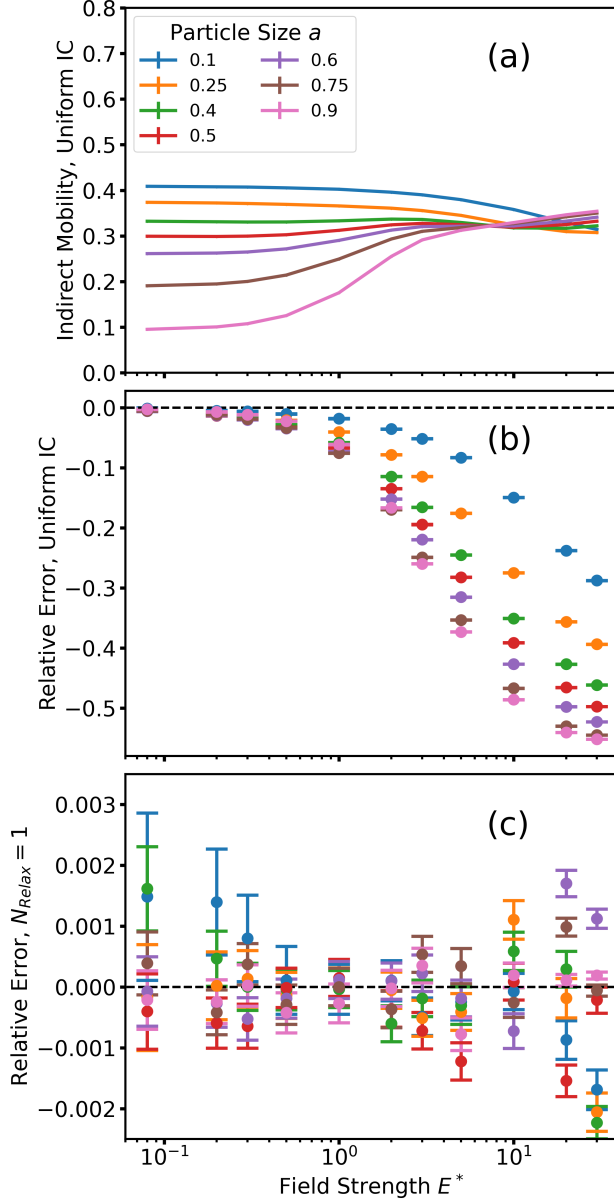


FIG. 5. (a) Incorrect indirect mobility values measured using the well-to-well configuration with uniform initial conditions for y shown as a function of field strength E^* for various particle diameters a . (b,c) Relative error of simulated indirect mobility values compared to simulated direct mobility values when indirect mobilities are calculated using initial x values in the middle of a well and (b) initial y values are distributed uniformly or (c) initial y values are sampled using MCMC with $N_{\text{relax}} = 1$. Error bars correspond to one standard error.

tion, these are

$$\tau_{\text{drift,well}} \sim \frac{L/2}{\mu_0 E^*}, \quad (9)$$

$$\tau_{\text{diff,well}} \sim \frac{a(\ell_{\text{well}} - a)^2}{2}. \quad (10)$$

In particular, note that the nominal size of the well is $\ell_{\text{well}} = 5$, which is much larger than the nominal size of the slit, $\ell_{\text{slit}} = 1$. Contrasting with the stationary distributions in the slit-to-slit configuration (Fig. 3), the well-to-well distributions are clearly far less uniform. With increasing Péclet number, the stationary distribution appears to favour y positions near the top of the well. This is consistent with the known physics of the system: when the applied field is strong, the larger particles are less likely to diffuse into the bottom of the well, and have larger mobilities as a result²³.

As was the case in the slit-to-slit configuration, non-uniformity is greatest when the Péclet number is large and the diffusion timescale in y is large relative to the transit timescale in x . However, whereas the geometric bottleneck in the slit limits the importance of non-uniformity for large a , in the well-to-well configuration the accessible range in y is large even at the largest a values. This result highlights a subtle but important aspect of utilizing the indirect mobility formulation in practice: the behaviour depends on the choice of location for the period-to-period threshold. This choice affects the complexity of the true stationary distribution, and thus affects the possibility of approximating it analytically (e.g., using a uniform distribution) or the computational cost of sampling it numerically.

Nonetheless, even the intentionally sub-optimal choice of this threshold in the well-to-well configuration can be handled very efficiently in this case. The analysis in App. B suggests that the MCMC algorithm should converge exponentially with increasing N_{relax} , suggesting that perhaps large values are not necessary. Figure 5(c) shows the relative error of the well-to-well indirect mobility calculation when the initial conditions are sampled using the MCMC algorithm with $N_{\text{relax}} = 1$. Even with relaxation through only a single period, the relative error has become essentially negligible—although errors are statistically discernible at high E^* , these relative errors are on the order of 0.1% or less.

The extremely fast convergence of the MCMC sampling protocol can once again be attributed to the geometric bottleneck in the slits. In the well-to-well configuration, the distribution of y values after crossing one period are entirely specified by the intermediate y values in the slit. Because of the bottleneck, the system becomes thoroughly mixed at this location. It thus appears inevitable that the MCMC algorithm will converge very rapidly with N_{relax} whenever such a bottleneck is present.

Future work might explore other options for sampling from the stationary for indirect mobility calculations. Finite samples from the MCMC algorithm presented above might be smoothed by fitting to a histogram or using kernel density estimation, for instance. Alternative, generative modelling techniques (e.g., generative adversarial networks) might be of interest. Furthermore, the indirect mobility can also be applied to solutions of the time-integrated Smoluchowski PDE, as explored in Nagel

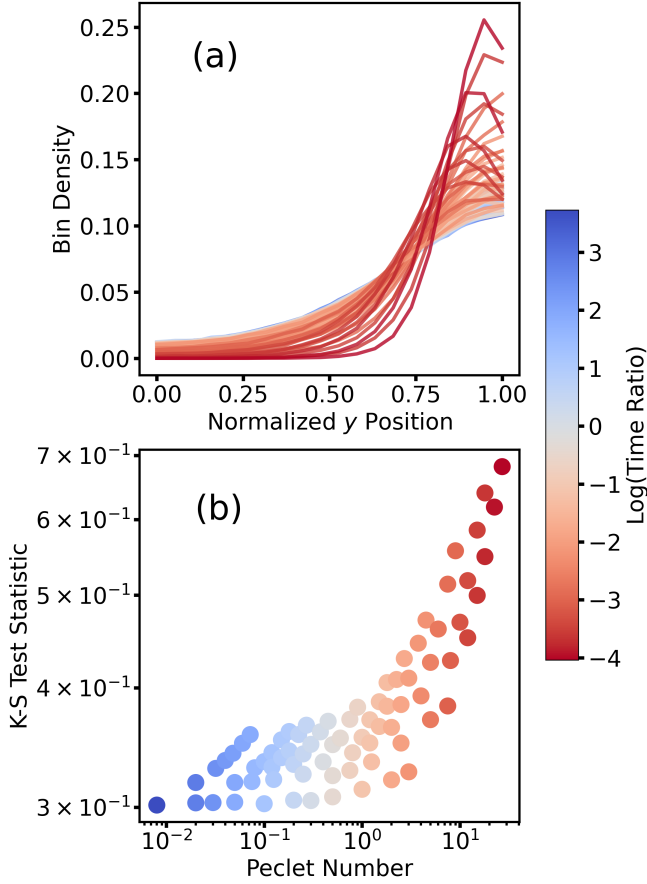


FIG. 6. (a) Normalized histograms (with 20 bins) of the stationary distributions obtained with $N_{\text{relax}} = 1$ for the well-to-well configuration. Normalized y position spans the available y coordinates in the well, which depend on a via the reflective boundary conditions. (b) Kolmogorov-Smirnov test statistic between the sampled distributions and the uniform distribution (higher value indicates less uniform behaviour), as a function of Péclet number. Colours in (a) and (b) show $\log(\tau_{\text{drift,well}}/\tau_{\text{diff,well}})$ as described in the text.

et al.⁵² (see Sec. A 3). That application actually requires the estimation of the stationary distribution’s probability distribution function, rather than only requiring samples drawn from that distribution. In the context of the algorithm of Nagel et al.⁵², this could be accomplished for instance by using an auxiliary neural network to represent the stationary distribution and imposing a self-consistency condition on the distribution of y values at the absorbing boundary condition.

C. Computational cost comparison

The demonstration in Sec. III A confirmed that the direct and indirect mobilities are equivalent, as derived in App. B. Those simulations used an MCMC algorithm to sample the stationary distribution required for the indi-

rect mobility calculations. As discussed, if every trajectory used to calculate indirect mobility requires an independent MCMC sample, and if the MCMC samples require a large value of N_{relax} , then the cost of the MCMC sampling protocol will dominate the cost of the indirect mobility calculation. Luckily, as explored in Sec. III B, it appears possible to greatly reduce this overhead cost. At least for the model described in Sec. II B, a small number of MCMC samples can be recycled across many trajectories without introducing substantial error. Convergence of the MCMC sampler is expected to be exponential in N_{relax} in general (App. B 2), but geometric bottlenecks were argued in Sec. II B to produce particularly fast convergence. Altogether, it appears that the cost of sampling the stationary distribution by MCMC can be made a negligible fraction of the computational cost of the indirect mobility calculation in many cases.

With this established, the current section provides a direct comparison of the computational costs of the direct and indirect mobility formulations. It is based on the theoretical cost analysis included in App. C, which is briefly recounted below. The predicted convergence rates of the two mobilities are tested against their actual performance on the problem of nanoparticles in the slit-well from Sec. II B. The analysis ignores the cost of the MCMC sampling protocol, based on the arguments above that this is likely a small addition to the overall cost of the indirect mobility.

Appendix C contains an analysis of the approximate computational cost of measuring the mobility to a target level of relative error ϵ using either the direct or indirect mobility formulations. The convergence rates are estimated by leveraging the detailed prediction of the limiting x position distribution obtained in App. B. Specifically, the limiting distribution is predicted to be a normal distribution with mean and variance given by Eqns. B15 and B16, reproduced here for convenience:

$$\langle x(t) \rangle = L \frac{t}{\langle \tau_1 \rangle} + L \frac{1}{2} \frac{\sigma^2}{\langle \tau_1 \rangle^2} - \langle \delta_x \rangle, \quad (11)$$

$$\text{var}(x(t)) = L^2 \frac{\sigma^2}{\langle \tau_1 \rangle^2} \frac{t}{\langle \tau_1 \rangle} + L^2 \frac{5}{4} \frac{\sigma^4}{\langle \tau_1 \rangle^4} + \text{var}(\delta_x). \quad (12)$$

Here L is the period length of the system, $\langle \tau_1 \rangle$ is the mean first passage time across each period (assuming the stationarity condition for auxiliary coordinates), and σ is a correlation-adjusted standard deviation of the first passage time across each period (Eqn. B7).

The quantity δ_x is an additional random variable introduced in Sec. B 4 to account for the motion of analytes against the net long-time direction of motion. In particular, $\langle \delta_x \rangle$ is a measure of the mean fluctuation of the analyte’s x position between its first contact with period k and period $k + 1$. Because of the stationarity condition, the statistics of δ_x do not depend on k , and thus δ_x plays no role in determining the mobility. However, as discussed below, it does play a very important role in the rate of convergence of the direct mobility calculation.

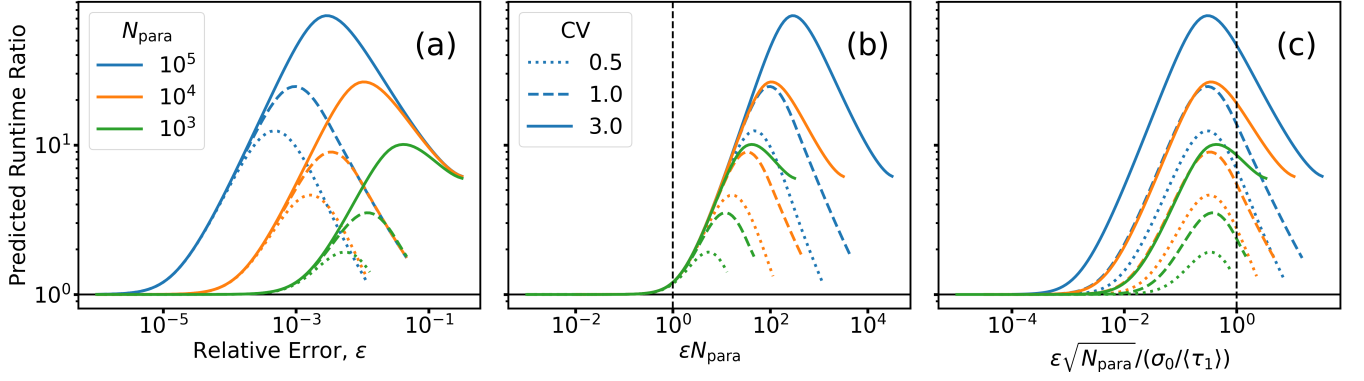


FIG. 7. Ratio of the predicted runtimes for the direct and indirect mobility estimators, for various choices of N_{para} (indicated by line color) and $\text{CV} = \sigma_0/\langle\tau_1\rangle$ (indicated by line style). The predicted runtime ratio is plotted against (a) the target relative error ϵ , (b) ϵN_{para} , and (c) $\epsilon\sqrt{N_{\text{para}}} / (\sigma_0/\langle\tau_1\rangle)$. The black vertical line in (b) indicates the predicted transition of the direct mobility estimator from order $\mathcal{O}(1/T_{\text{direct}})$ to order $\mathcal{O}(1/\sqrt{T_{\text{direct}}})$, while the black vertical line in (c) indicates the predicted transition of the indirect mobility estimator from exponential convergence to order $\mathcal{O}(1/\sqrt{T_{\text{indirect}}})$. All plots neglect the cost of MCMC sampling and the effect of correlations between first passage times (i.e., assume $\psi = \sigma/\sigma_0 = 1$; see App. C).

A measurement of the direct mobility using particle simulations essentially amounts to generating a sample of $x(t)$ values in order to estimate $\langle x(t) \rangle$ in the direct mobility definition (Eqn. 1). Appendix C1 uses the predicted mean and variance of $x(t)$ to deduce the mean relative error (Eqn. C2) and standard relative error (Eqn. C3) of the direct mobility estimator in terms of L , $\langle\tau_1\rangle$, σ , and $\langle\delta_x\rangle$. These errors are expressed as functions of the number of independent trajectories sampled and the total runtime for which the trajectories are evolved. The total relative error of the direct mobility estimator is obtained by adding the mean and standard error in quadrature.

Equation C2 states that the mean relative error of the direct mobility estimator is proportional to

$$\frac{1}{2} \frac{\sigma^2}{\langle\tau_1\rangle^2} - \frac{\langle\delta_x\rangle}{L} \quad (13)$$

divided by the total runtime of the simulated trajectories. This prefactor can potentially become very small if its two terms are comparable in magnitude. However, understanding the behaviour of δ_x was deemed beyond the scope of the analysis in App. C. The comparison of direct and indirect mobility convergence rates was conducted with the approximation $\delta_x \approx 0$. This is one of the major limitations of that analysis, and the numerical demonstrations below will investigate the implications on convergence rates obtained in practice.

The error convergence of the indirect mobility estimator was approximated by assuming that first passage times across any given period have exponentially decaying tails. Specifically, the probability density function $\rho(\tau_1)$ of the first passage time is assumed to be of the form (Eqn. C6)

$$\rho(\tau_1) \approx \frac{1}{\tau^*} \exp\left(-\frac{\tau_1}{\tau^*}\right) \quad (14)$$

at large τ_1 , where τ^* is some constant. This is generally a fair assumption, since the tails of the τ_1 distribution are

generated by those stochastic trajectories that happen to remain trapped for long periods of time (see the theoretical frameworks in App. A2). However, it is not generally the case that τ^* is equal to the mean first passage time $\langle\tau_1\rangle$. Nonetheless, the simplifying assumption $\tau^* \approx \langle\tau_1\rangle$ was made in parts of App. C, as a proper characterization of τ^* is difficult in general. This is the second major limitation in the theoretical comparison between the direct and indirect mobility convergence rates, and will also be addressed in the numerical demonstrations below.

The analysis in App. C culminates in predictions for the total computation time necessary to achieve a relative error of ϵ using either method when a total of N_{para} parallel threads are available. Figure 7 summarizes the main results of the analysis. Figure 7(a) shows the predicted ratio of the runtimes for the direct and indirect mobility estimators. Results are shown for $N_{\text{para}} = 10^3, 10^4, 10^5$ and with the assumption that the coefficient of variation $\sigma_0/\langle\tau_1\rangle$ of the first passage time across a single period is 0.5, 1.0, or 3.0. Note that here σ_0 is the actual standard deviation of the first passage time, which is assumed to be similar to the correlation-adjusted standard deviation σ ; see App. C for details.

The general conclusion is that, given sufficient access to parallel computing hardware, the indirect mobility appears to be a more efficient choice. Figure 7(b) illustrates that for target errors below $\epsilon \approx 1/N_{\text{para}}$, the two mobility formulations have roughly the same runtime. Conversely, the maximum advantage of using the indirect mobility occurs for target errors close to $\frac{\sigma_0/\langle\tau_1\rangle}{\sqrt{N_{\text{para}}}}$, as indicated in Fig. 7(c). As N_{para} increases, the relative cost of the indirect mobility to the direct mobility decreases at all values of ϵ , but the ϵ at which the ratio is maximized shifts to lower values. In practice, ϵ values near 0.1-1% are commonly used in MNFD research, and the N_{para} values listed in Fig. 7 are increasingly affordable thanks to GPU acceleration. Thus, Fig. 7 shows that

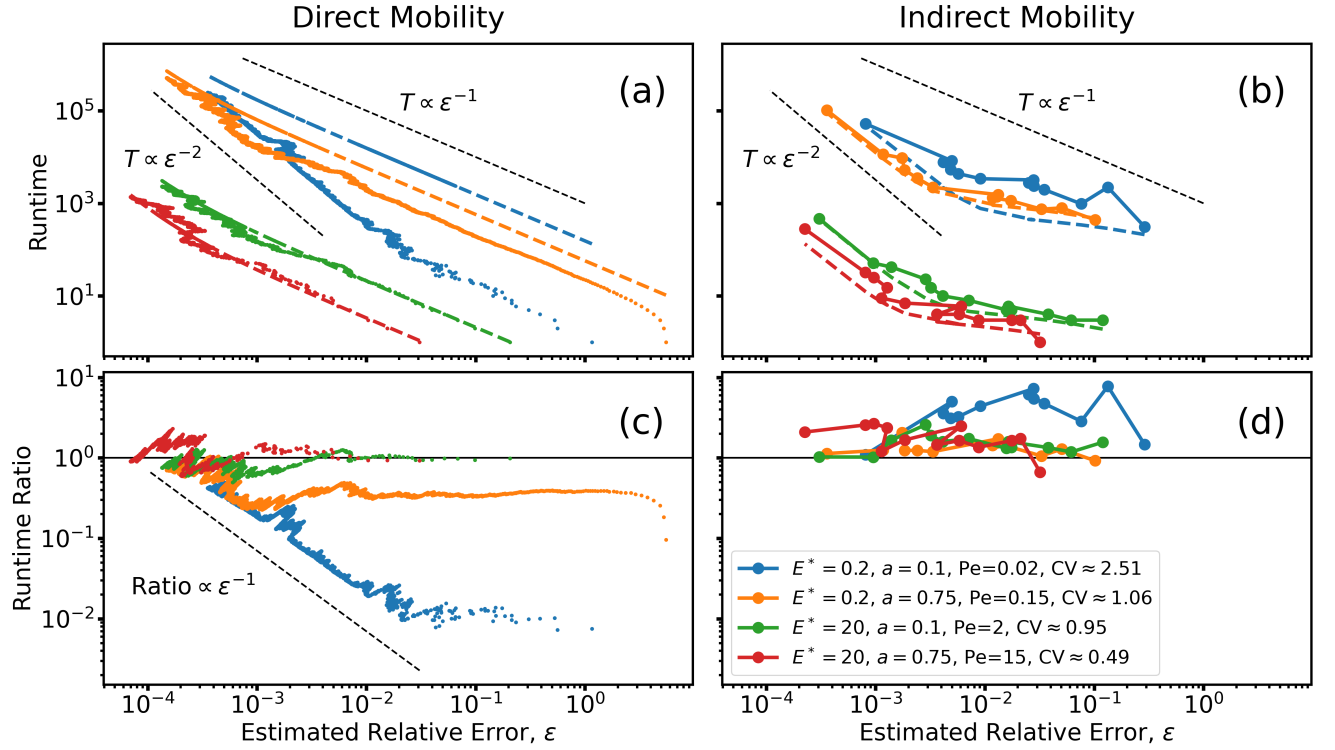


FIG. 8. (a,b): Measured runtimes for (a) direct mobility estimators and (b) indirect mobility estimators shown as a function of estimated relative error, for four physical scenarios indicated in the legend. Pe indicates the Péclet number $Pe = E^*a$ and CV stands for the coefficients of variation $\sigma_0/\langle\tau_1\rangle$. (c,d) Deviation of the measured (c) direct mobility and (d) indirect mobility behaviour from the predicted behaviour. All results for $N_{para} = 10^4$.

the theoretical analysis of App. C predicts accelerations of an order of magnitude or more by switching to the indirect mobility formulation under practically relevant conditions.

However, as noted above, the analysis in App. C and the results in Fig. 7 are based on two questionable approximations. First, it neglects the effect of the quantity $\langle\delta_x\rangle$ in Eqn C1, which characterizes the motion of analytes against the net force between the first passage to the k th period and the first passage to period $k+1$. Second, it assumes that the first passage times are exponentially distributed with a time constant τ^* that is similar in magnitude to the mean first passage time $\langle\tau_1\rangle$. If $\langle\delta_x\rangle$ is large or $\tau^* \gg \langle\tau_1\rangle$, the predicted computational advantages of the indirect mobility over the direct mobility may be smaller than expected.

Figure 8 presents numerical measurements of the runtimes and estimated relative errors for the direct and indirect mobility estimators, computed from simulations of nanoparticles in the slit-well device (Sec. IIB). In each subplot of Fig. 8, four physical scenarios are considered: all combinations of $E^* = 0.2, 20$ and $a = 0.1, 0.75$. The simulation protocol is the same one that was used in Sec. III A, with calculations parallelized across $N_{para} = 10^4$ threads. The direct mobility curves in Figs. 8(a,c) each correspond to measurements taken throughout a

single long simulation trajectory. Each data point for the indirect mobility estimators in Figs 8(b,d) is sampled independently using varying number of trajectories (although the same MCMC samples of the stationary distributions are recycled for each physical scenario). Runtimes are reported in units of simulation time, and relative errors are estimated against the final direct mobility values for each of the four physical scenarios.

In Figs. 8(a,b), the coloured dotted lines correspond to the predicted runtime necessary to achieve a given relative error ϵ (Eqns. C23 and C30). The black dotted lines denote $T \sim 1/\epsilon$ and $T \sim 1/\epsilon^2$ scaling, corresponding to the expected limiting behaviour of the direct mobility estimator. In Figs. 8(c,d), the measured mobility results are divided by the predicted behaviours; the solid black line indicates where the simulation results and the theoretical predictions are in agreement. The dotted black line indicates a scaling of $1/\epsilon$, as discussed below.

In Fig. 8(a), the red and green lines match the theoretical predictions quite well, but it is clear that the blue and orange series are performing significantly better than expected from the analysis. Specifically, Fig. 8(c) shows that for sufficiently large values of ϵ the orange line is about twice as fast as expected and the blue line is roughly two orders of magnitude faster than expected. The deviation by this constant factor persists until a cer-

tain level of relative error is achieved, below which the ratio of measured to predicted runtimes decays to 1 at a rate of $1/\epsilon$ (indicated by the dotted line in Fig. 8(c)). These transition points in Fig. 8(c) coincide with the similar transition points in Fig. 8(a) at which the corresponding runtimes change from scaling as $1/\epsilon$ to scaling as $1/\epsilon^2$.

It appears that the disagreement between theory and reality for the convergence of the direct mobility estimator is larger for smaller Péclet numbers. This behaviour can be tentatively attributed to the omitted term $\langle \delta_x \rangle / L$ acting to reduce the prefactor of the mean relative error by a constant amount. As explained in App. C1, the mean relative error of the direct mobility estimator decays as $\epsilon \sim 1/T$ with runtime T , whereas its standard relative error decays as $\epsilon \sim 1/\sqrt{T}$. At large ϵ , the total error of the direct mobility estimator is dominated by the mean relative error. Since the prefactor of the mean relative error is missing $\langle \delta_x \rangle / L$, the measured runtime disagrees with the prediction by a constant factor. Conversely, at sufficiently small ϵ the total error becomes dominated by the standard relative error, which does not depend on $\langle \delta_x \rangle / L$. In the small ϵ regime, the disagreement with the theory decays at the same rate as the mean relative error, i.e. $1/\epsilon$. At large Péclet numbers, however, it appears that indeed $\langle \delta_x \rangle \approx 0$, since the measured runtime versus error agrees well with the prediction from App. C.

Whereas the direct mobility is performing much better than predicted in some cases, Figs. 8(b,d) show that the indirect mobility is performing somewhat less well than expected. The case with the lowest Péclet number (blue) exhibits runtimes nearly an order of magnitude larger than expected over much of the ϵ range. The other three cases (orange, green, red) have runtimes that only exceed the predicted runtime by a factor of 2-3 or less at all values of ϵ . These observations can be attributed to the difference between τ^* and $\langle \tau_1 \rangle$, which were assumed to be equal in the theoretical predictions.

The difference between τ^* and $\langle \tau_1 \rangle$ is better understood by considering the coefficient of variation than the Péclet alone. Indeed, the orange and green lines have very similar coefficients of variation but very different Péclet numbers; whereas only green agrees with theory in the direct mobility case (Fig. 8(a,c)), both agree comparably well with theory in the indirect mobility case (Fig. 8(b,d)). The coefficient of variation is equal to 1 for an exponential distribution, and the case of exponentially distributed first passage times corresponds to $\tau^* = \langle \tau_1 \rangle$. More generally, the coefficient of variation of a distribution is a common metric for the relative importance of the tails of the distribution. In any case, the magnitude of the gap between theory and practice for the indirect mobility appears less significant than that observed for the direct mobility.

In summary, the theoretical convergence analysis conducted in App. C and illustrated in Fig. 7 overpredicts the advantage of the indirect mobility in two ways. When

the Péclet number is low, the direct mobility performs better than expected, likely because of the action of δ_x to reduce the mean relative error at large ϵ . When the coefficient of variation is large, the indirect mobility performs worse than expected, likely because τ^* is significantly larger than $\langle \tau_1 \rangle$. Note that in the case of nanoparticles traversing the slit-well device, the Péclet number correlates very strongly with the coefficient of variation in the diffusive regime²³. Both of these effects tend to diminish the computational advantage of the indirect mobility over the direct mobility.

Regardless, the predicted computational advantage of the indirect mobility is still discernible in this system. Figure 9(b) is a plot of the ratio of the measured runtimes for the direct and indirect mobility estimators shown in Fig. 8(a,b), obtained by linearly interpolating the direct mobility curves in Fig. 8(a). Also included (dotted lines) are the theoretical predictions of the ratio based on Eqns. C23 and C30 (as shown in Fig. 7).

As expected from the discussion of Fig. 8, the measured runtime ratios match the theoretical prediction at high Péclet numbers, but significantly deviate at lower Péclet numbers. Nonetheless, the indirect mobility estimator is found to consistently converge faster than the direct mobility estimator for the three largest Péclet numbers for ϵ in the range of 0.1-1%. The largest increase in speed is observed for the green line, which converges roughly 6 times faster to an error of approximately 0.5%.

Figures 9(a,c) show how the measured runtime ratios change when these experiments are repeated with fewer parallel threads ($N_{\text{para}} = 10^3$) or more parallel threads ($N_{\text{para}} = 10^5$), respectively. For $N_{\text{para}} = 10^3$, the difference between the two estimators is difficult to resolve at any ϵ value. As noted in App. C3, the two algorithms are expected to have roughly identical convergence rates for small values of N_{para} . In practice, the direct mobility appears slightly more efficient, especially given that this plot omits the cost of sampling the stationary distribution for the indirect mobility estimator.

Conversely, for $N_{\text{para}} = 10^5$, the advantage of the indirect mobility is quite clear (Fig. 9(c)). In this case, observations are much better described by the theory from App. C. Even for the blue line, where the low Péclet number and large coefficient of variation were previously identified as substantially favouring the direct mobility, the indirect mobility calculation is several times faster at errors near 0.1%. The green and red lines, corresponding to the larger Péclet numbers, are very well-described by the theory, and are 10-20 times faster to compute at errors near 0.1%.

These results demonstrate that the use of the indirect mobility formulation may indeed be significantly faster under practical conditions. As noted earlier, target errors of 0.1-1% are typically appropriate for simulation studies of periodic MNFDs. The effective values of N_{para} in practice depend somewhat on implementation details, but can be estimated (at least for some implementations on certain NVIDIA GPUs) as 16 times the number of

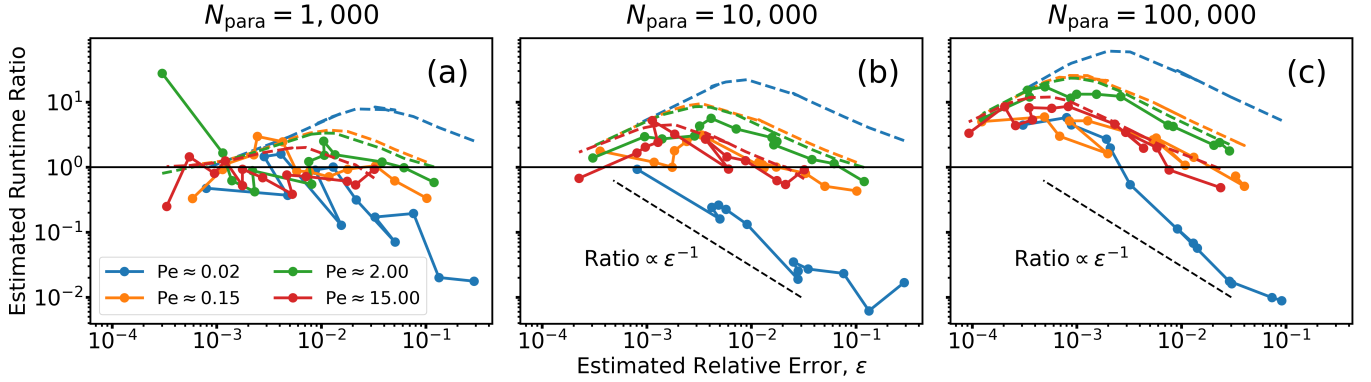


FIG. 9. Ratio of the measured runtimes for direct and indirect mobility estimators as a function of estimated relative error ϵ . Values for the direct mobilities were interpolated/extrapolated linearly to the required values of ϵ . The coloured dashed lines show the predicted ratio of runtimes for each case based on Eqs. C23 and C30. The solid black line shows a ratio of 1, whereas the dashed black line shows a scaling of $1/\epsilon$. Results shown for (a) $N_{\text{para}} = 10^3$, (b) $N_{\text{para}} = 10^4$, (c) $N_{\text{para}} = 10^5$.

CUDA cores⁵³. For our implementation, perfect parallelization to the level of $N_{\text{para}} \approx 10^4$ was easily achieved on a GTX 1650, a very modest consumer-grade GPU with 896 CUDA cores. The case of $N_{\text{para}} = 10^5$ was meant to illustrate what could be achieved by parallelizing across roughly ten such GPUs, a practice which is increasingly becoming commonplace. Experiments on a GTX 1080 Ti suggested that a single card of that variety could deliver $N_{\text{para}} \approx 5 \times 10^4$, consistent with it having 3584 CUDA cores. Although its performance was not tested with our implemented simulations, the RTX 3090 is listed as having 10496 CUDA cores, which may correspond to $N_{\text{para}} \approx 1.5 \times 10^5$ on a single card.

Depending on implementation details, simulations of M -body molecules may effectively be limited to roughly M times smaller values of N_{para} on the same hardware. This would be the case, for instance, if roughly M threads were allocated to accelerate each independent trajectory. This may or may not be advantageous compared to running serial M -body simulations on each of the N_{para} threads; such considerations likely depend on the physics of the system being simulated and are beyond the present scope. Regardless, the increasing affordability of massively parallel computing resources will enable larger N_{para} values for increasingly complex molecules. For instance, a set of 10 RTX 3090 cards could enable $N_{\text{para}} = 10^5$ with M as large as 150, in which case the indirect mobility formulation may feasibly be roughly five times faster than the standard direct mobility formulation, especially for systems with moderate-to-large Péclet numbers and/or geometric bottlenecks.

IV. CONCLUSION

The theoretical and empirical results presented in this work support the claim that the indirect mobility formulation (Eqn. 2) may be a more efficient option for comput-

ing the effective mobility of biomolecules driven through periodic geometries than the traditional direct mobility formulation (Eqn. 1). In the limit of unlimited parallel computing capacity and arbitrarily small target errors, the indirect mobility formulation leads to exponentially faster convergence. Given the growing importance and availability of parallel computing hardware for computational science, the relevance of this result is likely to increase in the future.

Even under realistic conditions of finite parallel computing capacity and target errors near or slightly below 1%, the indirect mobility can still be a substantially more efficient approach. In general, the relative performance of the two approaches appears to depend on a few key physical parameters of the system under study, and especially on the balance of drift to diffusion. In the example model of nanoparticles traversing the slit-well device, the indirect mobility was demonstrated to converge up to an order of magnitude faster in some circumstances (specifically, when the Péclet number is moderate or large), even using quite modest computing hardware.

Future work is needed to assess the relative merit of the indirect mobility formulation in simulations of other biophysical systems. The theoretical discussions in the appendices are applicable to a fairly general model of the transport of biomolecules through periodic geometries. However, the theoretical analyses of computational cost in App. C are limited by the approximations of nearly-exponential first passage time distributions and of nearly-negligible analyte motion against the direction of the applied force. The empirical results reveal that, in some cases, the direct mobility is actually a substantially more efficient estimator than the indirect mobility. Nonetheless, this only appears to be a practical concern in very weakly driven systems. Highly driven systems, where the indirect mobility is most useful, are likely to be of more practical relevance to the design of MNFDs.

- ¹Kevin D Dorfman. DNA electrophoresis in microfabricated devices. *Reviews of Modern Physics*, 82(4):2903, 2010.
- ²Stephen L Levy and Harold G Craighead. DNA manipulation, sorting, and mapping in nanofluidic systems. *Chemical Society Reviews*, 39(3):1133–1152, 2010.
- ³Kevin D Dorfman, Scott B King, Daniel W Olson, Joel DP Thomas, and Douglas R Tree. Beyond gel electrophoresis: microfluidic separations, fluorescence burst analysis, and DNA stretching. *Chemical Reviews*, 113(4):2584–2667, 2013.
- ⁴Mukul Sonker, Daihyun Kim, Ana Egatz-Gomez, and Alexandra Ros. Separation phenomena in tailored micro-and nanofluidic environments. *Annual Review of Analytical Chemistry*, 12:475–500, 2019.
- ⁵WD Volkmuth and RH Austin. DNA electrophoresis in microlithographic arrays. *Nature*, 358(6387):600–602, 1992.
- ⁶Kevin D Dorfman. DNA electrophoresis in microfluidic post arrays under moderate electric fields. *Physical Review E*, 73(6):061922, 2006.
- ⁷Jaeseol Cho, Satish Kumar, and Kevin D Dorfman. Electrophoretic collision of a DNA molecule with a small elliptical obstacle. *Electrophoresis*, 31(5):860–867, 2010.
- ⁸Pulak K Ghosh, Peter Hänggi, Fabio Marchesoni, S Martens, Franco Nori, Lutz Schimansky-Geier, and Gerhard Schmid. Driven Brownian transport through arrays of symmetric obstacles. *Physical Review E*, 85(1):011101, 2012.
- ⁹Zuzana Benková, Lucia Rišpanová, and Peter Cifra. Effect of chain stiffness for semiflexible macromolecules in array of cylindrical nanoposts. *The Journal of Chemical Physics*, 147(13):134907, 2017.
- ¹⁰Matthew C VandeSande, Daniel J Pasut, and Hendrick W de Haan. Sorting polymers by size via an array of viscous posts. *Electrophoresis*, 38(19):2488–2497, 2017.
- ¹¹Qiong Wu, Noritada Kaji, Takao Yasui, Sakon Rahong, Takeshi Yanagida, Masaki Kanai, Kazuki Nagashima, Manabu Tokeshi, Tomoji Kawai, and Yoshinobu Baba. A millisecond micro-RNA separation technique by a hybrid structure of nanopillars and nanoslits. *Scientific Reports*, 7(1):1–7, 2017.
- ¹²Taiga Ajiri, Takao Yasui, Masatoshi Maeki, Akihiko Ishida, Hirofumi Tani, Junji Nishii, Yoshinobu Baba, and Manabu Tokeshi. Silica nanopillar arrays for monitoring diffraction-based label-free biomolecule separation. *ACS Applied Nano Materials*, 3(9):8810–8816, 2020.
- ¹³Brato Chakrabarti, Charles Gaillard, and David Saintillan. Trapping, gliding, vaulting: transport of semiflexible polymers in periodic post arrays. *Soft Matter*, 16(23):5534–5544, 2020.
- ¹⁴TAJ Duke and RH Austin. Microfabricated sieve for the continuous sorting of macromolecules. *Physical Review Letters*, 80(7):1552, 1998.
- ¹⁵Deniz Ertaş. Lateral separation of macromolecules and polyelectrolytes in microlithographic arrays. *Physical Review Letters*, 80(7):1548, 1998.
- ¹⁶Chia-Fu Chou, Olgica Bakajin, Stephen WP Turner, Thomas AJ Duke, Shirley S Chan, Edward C Cox, Harold G Craighead, and Robert H Austin. Sorting by diffusion: An asymmetric obstacle course for continuous molecular separation. *Proceedings of the National Academy of Sciences*, 96(24):13762–13765, 1999.
- ¹⁷Alexander Van Oudenaarden and Steven G Boxer. Brownian ratchets: Molecular separations in lipid bilayers supported on patterned arrays. *Science*, 285(5430):1046–1048, 1999.
- ¹⁸Jongyoon Han and Harold G Craighead. Entropic trapping and sieving of long DNA molecules in a nanofluidic channel. *Journal of Vacuum Science & Technology A: Vacuum, Surfaces, and Films*, 17(4):2142–2147, 1999.
- ¹⁹Jongyoon Han, SW Turner, and Harold G Craighead. Entropic trapping and escape of long DNA molecules at submicron size constriction. *Physical Review Letters*, 83(8):1688, 1999.
- ²⁰Jongyoon Han and Harold G Craighead. Separation of long DNA molecules in a microfabricated entropic trap array. *Science*, 288(5468):1026–1029, 2000.
- ²¹Jongyoon Han and Harold G Craighead. Characterization and optimization of an entropic trap for DNA separation. *Analytical Chemistry*, 74(2):394–401, 2002.
- ²²Nabil Laachi, Carmelo Declet, Christina Matson, and Kevin D Dorfman. Nonequilibrium transport of rigid macromolecules in periodically constricted geometries. *Physical Review Letters*, 98(9):098106, 2007.
- ²³Kuang-Ling Cheng, Yu-Jane Sheng, Shaoyi Jiang, and Heng-Kwong Tsao. Electrophoretic size separation of particles in a periodically constricted microchannel. *The Journal of Chemical Physics*, 128(10):101101, 2008.
- ²⁴Elizabeth A Strychalski, Henry W Lau, and Lynden A Archer. Nonequilibrium separation of short DNA using nanoslit arrays. *Journal of Applied Physics*, 106(2):024915, 2009.
- ²⁵Zi Rui Li, Gui Rong Liu, Nicolas G Hadjicostantinou, Jongyoon Han, Jian-Sheng Wang, and Yu Zong Chen. Dispersive transport of biomolecules in periodic energy landscapes with application to nanofilter sieving arrays. *Electrophoresis*, 32(5):506–517, 2011.
- ²⁶Morten Bo Mikkelsen, Walter Reisner, Henrik Flyvbjerg, and Anders Kristensen. Pressure-driven DNA in nanogroove arrays: complex dynamics leads to length-and topology-dependent separation. *Nano Letters*, 11(4):1598–1602, 2011.
- ²⁷Santtu TT Ollila, Colin Denniston, Mikko Karttunen, and Tapio Ala-Nissila. Biopolymer filtration in corrugated nanochannels. *Physical Review Letters*, 112(11):118301, 2014.
- ²⁸Lingling Wu and Stephen Levy. Fluctuations of DNA mobility in nanofluidic entropic traps. *Biomicrofluidics*, 8(4):044103, 2014.
- ²⁹Marzieh Alishahi, Reza Kamali, and Omid Abouali. Numerical investigation of molecular nano-array in potential-energy profile for a single dsDNA. *The European Physical Journal E*, 39(4):1–7, 2016.
- ³⁰Sung Hee Ko, Divya Chandra, Wei Ouyang, Taehong Kwon, Pankaj Karande, and Jongyoon Han. Nanofluidic device for continuous multiparameter quality assurance of biologics. *Nature Nanotechnology*, 12(8):804–812, 2017.
- ³¹Zhen Cao and Levent Yobas. Fast DNA sieving through submicrometer cylindrical glass capillary matrix. *Analytical Chemistry*, 86(1):737–743, 2014.

- ³²Zhen Cao and Levent Yobas. Gel-free electrophoresis of DNA and proteins on chips featuring a 70 nm capillary-well motif. *ACS Nano*, 9(1):427–435, 2015.
- ³³Dmytro Nykypanchuk, Helmut H Strey, and David A Hoagland. Brownian motion of DNA confined within a two-dimensional array. *Science*, 297(5583):987–990, 2002.
- ³⁴Alexander M Berezhkovskii, Vladimir Yu Zitserman, and Stanislav Y Shvartsman. Diffusivity in periodic arrays of spherical cavities. *The Journal of Chemical Physics*, 118(15):7146–7147, 2003.
- ³⁵Alexander M Berezhkovskii, Vladimir Yu Zitserman, and Stanislav Y Shvartsman. Effective diffusivity in periodic porous materials. *The Journal of Chemical Physics*, 119(14):6991–6993, 2003.
- ³⁶Erica J Saltzman and Murugappan Muthukumar. Conformation and dynamics of model polymer in connected chamber-pore system. *The Journal of Chemical Physics*, 131(21):12B605, 2009.
- ³⁷Karthik Nagarajan and Shing Bor Chen. Driven transport of dilute polymer solutions through porous media comprising interconnected cavities. *Colloids and Interfaces*, 5(2):22, 2021.
- ³⁸Timo Ikonen. Driven polymer transport through a periodically patterned channel. *The Journal of Chemical Physics*, 140(23):234906, 2014.
- ³⁹Ying-Cai Chen, Yan-Li Zhou, and Chao Wang. Monte Carlo simulation on the diffusion of polymer in narrow periodical channels. *International Journal of Modern Physics B*, 31(21):1750144, 2017.
- ⁴⁰Lisa B Weiss, Arash Nikoubashman, and Christos N Likos. Topology-sensitive microfluidic filter for polymers of varying stiffness. *ACS Macro Letters*, 6(12):1426–1431, 2017.
- ⁴¹Chao Wang, Yan-Li Zhou, Li-Zhen Sun, Ying-Cai Chen, and Meng-Bo Luo. Simulation study on the migration of diblock copolymers in periodically patterned slits. *The Journal of Chemical Physics*, 150(16):164904, 2019.
- ⁴²Robert Riehn, Robert H Austin, and James C Sturm. A nanofluidic railroad switch for DNA. *Nano Letters*, 6(9):1973–1976, 2006.
- ⁴³Daniel Kim, Clark Bowman, Jackson T Del Bonis-ODonnell, Anastasios Matzavinos, and Derek Stein. Giant acceleration of DNA diffusion in an array of entropic barriers. *Physical Review Letters*, 118(4):048002, 2017.
- ⁴⁴Martin Magill, Ed Waller, and Hendrick W de Haan. A sequential nanopore-channel device for polymer separation. *The Journal of Chemical Physics*, 149(17):174903, 2018.
- ⁴⁵Zhi He and Ruhong Zhou. Exploring an in-plane graphene and hexagonal boron nitride array for separation of single nucleotides. *ACS Nano*, 15(7):11704–11710, 2021.
- ⁴⁶Emil Chmela, Robert Tijssen, Marko T Blom, Han JGE Gardener, and Albert van den Berg. A chip system for size separation of macromolecules and particles by hydrodynamic chromatography. *Analytical Chemistry*, 74(14):3470–3475, 2002.
- ⁴⁷Dongun Huh, Joong Hwan Bahng, Yibo Ling, Hsien-Hung Wei, Oliver D Kripfgans, J Brian Fowlkes, James B Grotberg, and Shuichi Takayama. Gravity-driven microfluidic particle sorting device with hydrodynamic separation amplification. *Analytical Chemistry*, 79(4):1369–1376, 2007.
- ⁴⁸Laurens-Jan C Jellema, Anton P Markesteijn, Jerry Westerweel, and Elisabeth Verpoorte. Tunable hydrodynamic chromatography of microparticles localized in short microchannels. *Analytical Chemistry*, 82(10):4027–4035, 2010.
- ⁴⁹Valentina Biagioni, Alpha L Sow, Alessandra Adrover, and Stefano Cerbelli. Brownian sieving effect for boosting the performance of microcapillary hydrodynamic chromatography. proof of concept. *Analytical Chemistry*, 93(17):6808–6816, 2021.
- ⁵⁰Peter Hänggi, Peter Talkner, and Michal Borkovec. Reaction-rate theory: fifty years after Kramers. *Reviews of Modern Physics*, 62(2):251, 1990.
- ⁵¹Peter E. Kloeden and Eckhard Platen. *Numerical solution of stochastic differential equations*. Springer-Verlag Berlin ; New York, 1992.
- ⁵²Andrew M Nagel, Martin Magill, and Hendrick W de Haan. Studying first passage problems using neural networks: A case study in the slit-well microfluidic device. *arXiv preprint arXiv:2204.12479*, 2022.
- ⁵³How many concurrent threads are running on my GeForce GTX 1080 Ti? NVIDIA Developer Forum. Online; accessed 2 June 2022.
- ⁵⁴Merkel Henry Jacobs. Diffusion processes. In *Diffusion Processes*, pages 1–145. Springer, 1967.
- ⁵⁵Robert Zwanzig. Diffusion past an entropy barrier. *The Journal of Physical Chemistry*, 96(10):3926–3930, 1992.
- ⁵⁶David Reguera and JM Rubi. Kinetic equations for diffusion in the presence of entropic barriers. *Physical Review E*, 64(6):061106, 2001.
- ⁵⁷P Kalinay and JK Percus. Corrections to the Fick-Jacobs equation. *Physical Review E*, 74(4):041203, 2006.
- ⁵⁸David Reguera, Gerhard Schmid, Poornachandra Sekhar Burada, JM Rubi, Peter Reimann, and Peter Hänggi. Entropic transport: kinetics, scaling, and control mechanisms. *Physical Review Letters*, 96(13):130603, 2006.
- ⁵⁹N Laachi, M Kenward, E Yariv, and KD Dorfman. Force-driven transport through periodic entropy barriers. *Europhysics Letters*, 80(5):50009, 2007.
- ⁶⁰Marco-Vinicio Vazquez, Alexander M Berezhkovskii, and Leonardo Dagdug. Diffusion in linear porous media with periodic entropy barriers: A tube formed by contacting spheres. *The Journal of Chemical Physics*, 129(4):046101, 2008.
- ⁶¹Giuseppe Forte, Fabio Cecconi, and Angelo Vulpiani. Transport and fluctuation-dissipation relations in asymptotic and preasymptotic diffusion across channels with variable section. *Physical Review E*, 90(6):062110, 2014.
- ⁶²D Reguera and JM Rubi. Engineering tube shapes to control confined transport. *The European Physical Journal Special Topics*, 223(14):3079–3093, 2014.
- ⁶³J Miguel Rubi. Entropic diffusion in confined soft-matter and biological systems. *Europhysics Letters*, 127(1):10001, 2019.

- ⁶⁴Ivan Pompa-García and Leonardo Dagdug. Two-dimensional diffusion biased by a transverse gravitational force in an asymmetric channel: Reduction to an effective one-dimensional description. *Physical Review E*, 104(4):044118, 2021.
- ⁶⁵Poornachandra Sekhar Burada, Gerhard Schmid, David Reguera, JM Rubi, and Peter Hänggi. Biased diffusion in confined media: test of the Fick-Jacobs approximation and validity criteria. *Physical Review E*, 75(5):051111, 2007.
- ⁶⁶Yu A Makhnovskii, V Yu Zitserman, and AM Berezhkovskii. Diffusion in quasi-one-dimensional structures with a periodic sharp narrowing of the cross section, 2009.
- ⁶⁷Alexander M Berezhkovskii and Leonardo Dagdug. Analytical treatment of biased diffusion in tubes with periodic dead ends. *The Journal of Chemical Physics*, 134(12):124109, 2011.
- ⁶⁸Yu A Makhnovskii, AM Berezhkovskii, LV Bogachev, and V Yu Zitserman. Driven diffusion in a periodically compartmentalized tube: Homogeneity versus intermittency of particle motion. *The Journal of Physical Chemistry B*, 115(14):3992–4002, 2011.
- ⁶⁹Alexander M Berezhkovskii and Leonardo Dagdug. Biased diffusion in periodic potentials: Three types of force dependence of effective diffusivity and generalized Lifson-Jackson formula. *The Journal of Chemical Physics*, 151(13):131102, 2019.
- ⁷⁰Narender Khatri and PS Burada. Diffusion of interacting particles in a channel with reflection boundary conditions. *The Journal of Chemical Physics*, 151(9):094103, 2019.
- ⁷¹Hai-Wei Hu, Lin Du, Liang-Hui Qu, Zi-Lu Cao, Zi-Chen Deng, and Ying-Cheng Lai. Current reversal and particle separation in Brownian transport. *Physical Review Research*, 3(3):033162, 2021.
- ⁷²Leonardo Dagdug, Alexander M Berezhkovskii, Vladimir Yu Zitserman, and Sergey M Bezrukov. Effective diffusivity of a Brownian particle in a two-dimensional periodic channel of abruptly alternating width. *Physical Review E*, 103(6):062106, 2021.
- ⁷³Peter Reimann, Christian Van den Broeck, H Linke, Peter Hänggi, JM Rubi, and Agustín Pérez-Madrid. Giant acceleration of free diffusion by use of tilted periodic potentials. *Physical Review Letters*, 87(1):010602, 2001.
- ⁷⁴Peter Reimann, Christian Van den Broeck, H Linke, Peter Hänggi, JM Rubi, and Agustín Pérez-Madrid. Diffusion in tilted periodic potentials: Enhancement, universality, and scaling. *Physical Review E*, 65(3):031104, 2002.
- ⁷⁵Benjamin Lindner, Marcin Kostur, and Lutz Schimansky-Geier. Optimal diffusive transport in a tilted periodic potential. *Fluctuation and Noise Letters*, 2001.
- ⁷⁶Pavel Ivanovich Kuznetsov, RL Stratonovich, and Vasiliĭ Ivanovich Tikhonov. *Non-linear transformations of stochastic processes*. Pergamon, 1965.
- ⁷⁷Shneior Lifson and Julius L Jackson. On the self-diffusion of ions in a polyelectrolyte solution. *The Journal of Chemical Physics*, 36(9):2410–2414, 1962.
- ⁷⁸Robert Zwanzig. *Nonequilibrium statistical mechanics*. Oxford University Press, 2001.
- ⁷⁹Sidney Redner et al. *A guide to first-passage processes*. Cambridge University Press, 2001.
- ⁸⁰Joseph L Doob. *Stochastic processes*, volume 7. Wiley New York, 1953.
- ⁸¹Theodore E Harris. The existence of stationary measures for certain Markov processes. *Matematika*, 4(1):131–143, 1960.
- ⁸²Zlochín Mark and Yoram Baram. The bias-variance dilemma of the Monte Carlo method. In *International Conference on Artificial Neural Networks*, pages 141–147. Springer, 2001.
- ⁸³Igor V Grigoriev, Yurii A Makhnovskii, Alexander M Berezhkovskii, and Vladimir Yu Zitserman. Kinetics of escape through a small hole. *The Journal of Chemical Physics*, 116(22):9574–9577, 2002.

Appendix A: Review of related theoretical frameworks

Below are included reviews of other theoretical frameworks in which the direct and indirect mobilities have been shown to be equivalent. All of these derivations are less general than the derivation included in App. B. The result of App. B is more broadly applicable, as it does not make strong assumptions about separation of timescales in the system, and it holds for many-body molecules and for time-varying force fields. Moreover, App. B yields specific equations for the limiting x position distribution, enabling the convergence rate analysis in App. C.

1. Quasi-1D systems: Fick-Jacobs theory

The Fick-Jacobs equation was first presented by Jacobs⁵⁴ as an effective one-dimensional model for steady-state diffusion in a confined system of varying cross-section. Essentially, the two- or three-dimensional Smoluchowski equation is integrated across the cross-section of the system. The volume available to particles in these transverse coordinates is approximated in the Fick-Jacobs equation by an extra free energy term. A more rigorous formulation was put forth by Zwanzig⁵⁵, who greatly increased the applicability of the equation by formulating a position-dependent effective diffusion coefficient. Further extensions and corrections were proposed in subsequent works^{56,57}.

The Fick-Jacobs equation and similar approaches have been used successfully to explain diffusion in quasi-one-dimensional systems with or without applied forces, including cases with periodic geometries^{58–64}. However, despite the various refinements that have been proposed, it generally fails to perform adequately in certain limits. Because the theory assumes rapid relaxation in the transverse coordinates, it tends to encounter problems in systems with strong applied forces or sudden changes in cross-sectional area^{65–72}. Moreover, the theory has a limited capacity to handle spatial variations in the applied force field, especially in the direction of the transverse coordinates (see however Pompa-García and Dagdug⁶⁴ for an example where Fick-Jacobs was successfully extended in this manner). Most importantly for

the study of biomolecules in periodic MNFDs, the Fick-Jacobs equation is restricted to the diffusion of single Brownian particles, and does not directly deal with the case of many-body molecules.

Since the Fick-Jacobs equation is effectively a one-dimensional equation, it benefits from many results applicable to the one-dimensional Smoluchowski equation. While studying the effective diffusion coefficient of one-dimensional Brownian particles in tilted periodic potentials, Reimann et al.⁷³ and Reimann et al.⁷⁴ proved that the indirect mobility (Eqn. 2) is equal to the direct mobility (Eqn. 1). Lindner et al.⁷⁵ connected that work to a classical result due to Stratonovich⁷⁶. These results have since been used to compute mobilities in periodic quasi-one-dimensional systems^{58,70,71}. However, this proof of equivalence is naturally restricted to the scope of applicability of Fick-Jacobs theory. Moreover, because it is based on one-dimensional approximations of point-particles, arguments like those in Reimann et al.⁷³ cannot account for correlations between crossing times (cf. App. B 2).

2. Large barriers: Poisson point processes

The equivalence of the direct and indirect mobilities is also known to hold in the case that the transport of molecules across each period of the system is obstructed by a large free energy barrier. In such a setting, there is a well-defined separation of timescales between the period-to-period transport process and all other processes occurring in the system. Reaction rate theories, such as Kramers theory, can be brought to bear on the problem (see Hänggi et al.⁵⁰ for a review of such theories).

Systems in this regime can be described as Poisson point processes. The probability that a particle initially trapped in a given period has not yet escaped to the next period decays exponentially as a function of time:

$$P(\text{not absorbed after time } t) \sim \exp(-\lambda t). \quad (\text{A1})$$

Moreover, transfers between distinct periods will certainly be statistically independent events, since by assumption these events occur more slowly than all other relaxation timescales in the system. For such exponentially distributed times, the mean rate λ is related to the mean first passage time τ by

$$\lambda = \frac{1}{\langle \tau \rangle}. \quad (\text{A2})$$

In the context of mobility through periodic geometries, the mean position on long timescales will thus be

$$\langle x(t) \rangle \rightarrow L\lambda t = \frac{Lt}{\langle \tau \rangle}. \quad (\text{A3})$$

Dividing both sides by t yields the equivalence of Eqns. 1 and 2.

These theories have been used to analyse particle transport in titled periodic potentials⁵⁰. However, Kramers theory and related reaction rate perspectives are restricted in applicability by their strong assumption of timescale separation. Whereas Fick-Jacob methods assume rapid relaxation of position coordinates in the directions transverse to bulk motion, reaction rate theories generally assume rapid relaxation of all processes but the dominant transport process. This approximation once again breaks down in situations with strong driving forces and non-equilibrium effects.

Despite their limitations, these theories are still widely used in practice for describing motion in periodic MNFDs. In particular, the assumption of exponentially distributed times is often used to justify physical models based on mean first passage times (see, for instance, Han et al.¹⁹, Cheng et al.²³, and Wang et al.⁴¹, for a few examples of such arguments). The results presented in this work show that the connection between transport rates and mean first passage times can be extended to more general physical circumstances, so long as the stationary distribution upon which the mean first passage times are based is defined appropriately.

3. Other mean first passage time methods

Besides the Fick-Jacobs and reaction rate theoretical frameworks, there have also been a variety of other cases in which mean first passage times were used to understand the mobilities of molecules traversing periodic geometries. In particular, mean first passage time perspectives have been used successfully to study the driven diffusion of Brownian particles in geometries with abruptly changing cross-sections, where the Fick-Jacobs perspective is not applicable^{66–69,72}. We will also mention in passing that these studies have also successfully used mean first passage time frameworks to understand effective diffusion, another important concept in the research and development of periodic MNFDs. The Lifson-Jackson method is one of the earliest methods that studied effective diffusion from this perspective^{69,77}. We will briefly comment on how our own result connects to the concept of effective diffusion in App. B 4, but leave more careful considerations of this aspect for future work.

The Smoluchowski equation describing the evolution of the position probability density function for Brownian molecules is another important theoretical framework for mean first passage time analysis. An adjoint equation to the Smoluchowski equation can be constructed whose solution at any point in the domain equals the mean first passage time from that point to absorbing regions on the domain's boundary⁷⁸. In fact, essentially this method was used by Lifson and Jackson⁷⁷ in their analysis.

Another very similar equation is the time-integrated Smoluchowski equation, whose solution is commonly denoted g_0 ⁷⁹. The source term in the time-integrated Smoluchowski equation corresponds to a certain choice

of initial particle positions in the system, and the solution g_0 has the property that its integral over the domain equals the mean first passage time. A recursive hierarchy of equations can be constructed to obtain the higher moments of the first passage time in the same manner.

The qualitative behaviour of g_0 solutions in a periodic MNFD was studied by Magill et al.⁴⁴. This analysis motivated a certain normalization of first passage times which elucidated a universal scaling behaviour across system geometries. Moreover, Magill et al.⁴⁴ argued that the long-time of molecules traversing that MNFD was entirely determined by the first and second moments of the first passage times across each period, which would be completely captured by the g_0 and g_1 fields.

Elaborating on the g_0 field as a proxy for connecting MNFD geometries with effective mobilities, Nagel et al.⁵² used a method based on neural networks to solve g_0 in a system similar to that studied by Cheng et al.²³. By computing four-dimensional approximations of g_0 as a function of both domain coordinates and model parameters, Nagel et al.⁵² demonstrated the idea of using neural networks to construct differentiable mappings from system design parameters to physical observables of interest (in this case, effective mobility). The use of g_0 in this way is a particular motivation for understanding the indirect mobility; the direct mobility formulation cannot be expressed in such a straightforward manner as the solution to a partial differential equation.

Appendix B: Derivation of the equivalence of direct and indirect mobilities

This section presents a proof that the indirect mobility (Eqn. 2) is equivalent to the more common direct mobility (Eqn. 1), so long as the initial conditions used to compute the indirect mobility are chosen correctly. The approach of the proof is to derive the limiting form of the position distribution $\rho(x)$ at long times in terms of the mean first passage time across a single period $\langle\tau_1\rangle$. From this solution, it is possible to equate the limiting drift velocity $\lim_{t\rightarrow\infty}\langle x(t)\rangle/t$ to $L/\langle\tau_1\rangle$. It then follows readily that the two mobility definitions are equivalent.

The general setup for the proof (App. B 1) is very similar to the arguments presented previously by Reimann et al.⁷³ and Magill et al.⁴⁴. The final steps of the proof (Apps. B 3 and B 4) are very similar to the steps taken by Magill et al.⁴⁴. The argument justifying $x \approx kL$ despite analyte backflow (App. B 4) is essentially the same used by Reimann et al.⁷³. The first part of the derivation (App. B 2), however, differs substantially from those prior derivations in order to account for correlations in the crossing times between periodic subunits. Such correlations were absent in the system studied by Magill et al.⁴⁴ because of geometric bottlenecks between the periodic subunits, and were irrelevant to the study of Reimann et al.⁷³ which considered only Brownian point particles in a one-dimensional system. They are handled

here via the judicious application of the Markov chain central limit theorem to an appropriately constructed Markov chain model of the transport process.

1. The time to first cross k subunits

Recall from Sec. II A that x_i denotes the threshold into the i th period, θ_i denotes the values of the auxiliary coordinates measured at the first time for which $x(t) = x_i$, and $t_{i,i+1}$ denotes the time between first contact with x_i and first contact with x_{i+1} . Now let us denote by τ_k the total first passage time from the original analyte position at $x = x_0$ to the threshold of the k th periodic subunit at $x = x_k$. By definition,

$$\tau_k = t_{0,1} + t_{1,2} + t_{2,3} + \cdots + t_{k-1,k}, \quad (\text{B1})$$

where $t_{i,i+1}$ is the time to reach x_{i+1} for the first time after having reached x_i for the first time. In the rest of this section, the index k will be used to indicate the total number of channels being crossed, whereas the index i with $0 \leq i \leq k-1$ will be used to refer to the intermediate channels crossed along the way to the k th channel.

Since τ_k is the sum of a series of random variables, it is tempting to appeal to the central limit theorem to deduce its limiting distribution. However, the application of the central limit theorem would require that the random variables $\{t_{i,i+1}\}_{i=0}^{k-1}$ be identically distributed and uncorrelated. As will be shown in App. B 2, it is usually possible to initialize the auxiliary coordinates θ_0 such that the $\{t_{i,i+1}\}_{i=0}^{k-1}$ are indeed identically distributed. However, in general it is not possible to eliminate the correlations between the crossing times. Specifically, the correlation of $t_{i,i+1}$ with $t_{i-1,i}$ is mediated by the auxiliary coordinates θ_i measured at first contact with x_i .

Conveniently, the nature of these correlations is still very tractable. The Markovian assumption made in Sec. II A amounts to the statement that the sequence $\{\theta_i\}_{i=0}^{k-1}$ is a Markov chain. Thus, the random process

$$(\theta_0, t_{0,1}) \rightarrow (\theta_1, t_{1,2}) \rightarrow \cdots \rightarrow (\theta_k, t_{k,k+1}) \rightarrow \cdots \quad (\text{B2})$$

is also a Markov chain. Incidentally, since θ_i alone completely specifies the joint distribution of $(\theta_{i+1}, t_{i+1,i+2})$, Eqn. B2 is a special type of Markov chain known as a hidden Markov model; however, this has no bearing on the current analysis. What is important is that the $t_{i,i+1}$ are fixed observables (i.e., real-valued functionals) of the state $(\theta_i, t_{i,i+1})$.

2. The stationary distribution

The distribution of τ_k can be deduced by applying the Markov chain central limit theorem to the Markov chain given by Eqn. B2. This theorem generalizes the central limit theorem, which applies to a sum of independent and identically distributed (i.i.d.) random variables, to

the cumulative sum of real-valued functionals of a stationary Markov chain. In particular, we will consider the functional $g(\theta_i, t_{i,i+1}) = t_{i,i+1}$. Note that many variations and extensions of the Markov chain central limit theorem exist, but we only need to appeal to the version of Doob⁸⁰.

In order to apply the theorem, it is necessary for the Markov chain to be initialized in its stationary distribution π , which satisfies

$$\pi(\theta_{i+1}, t_{i+1,i+2}) = \pi(\theta_i, t_{i,i+1}). \quad (\text{B3})$$

In particular, because the marginal distributions of each $t_{i,i+1}$ are completely determined by θ_i , this reduces to the requirement that

$$\pi(\theta_{i+1}) = \pi(\theta_i). \quad (\text{B4})$$

In general, the existence and uniqueness of a Markov chain's stationary distribution π depends on the details of its transition operator. If the state space is finite, then it is sufficient for the transition to be irreducible and aperiodic. This is the case, for instance, when every state θ_{i+1} has a non-zero probability of occurring after any state θ_i . Physically, this type of behaviour is common: motion driven by Brownian noise, for instance, usually behaves this way.

Unfortunately, ensuring the existence and uniqueness of a stationary distribution π can be challenging in the case of Markov chains with continuous state spaces—irreducibility and aperiodicity of the transition operators are no longer sufficient conditions. Alas, this is probably the more common scenario in biophysics, arising for instance in the case where the auxiliary coordinates θ are the atomic coordinates of a molecule and space is modelled as continuous. Various conditions are known to ensure existence and uniqueness of stationary distributions for continuous state spaces; see for instance Doob⁸⁰ or Harris⁸¹. In particular, if θ_{i+1} is distributed according to a probability density function that is continuous in θ_i , irreducible, and aperiodic, then Eqn. B2 satisfies the conditions of Example 2 on page 215 of Doob⁸⁰. Under these conditions, the stationary distribution exists and is unique, and moreover the Markov chain converges exponentially fast to this stationary distribution from any initial condition. The auxiliary coordinates θ are likely to satisfy this condition (at least to a very good approximation) in most relevant biophysical models.

The exponential convergence of Eqn B2 to its stationary distribution π suggests that Markov Chain Monte Carlo is a practical method for sampling from π . That is, if Eqn. B2 can be initialized in any convenient state θ_0 and the evolution of the system is simulated until its first passage through k_{relax} of periods through the device, the final state $\theta_{k_{\text{relax}}}$ will be approximately sampled from π . The number of relaxation periods k_{relax} should not need to be very large if the convergence of Eqn. B2 to π is indeed exponential for the system under study. The computational cost of this sampling method will be neglected from the cost analysis of computing the indirect

mobility in App. C. However, an empirical examination of its performance in practice will be presented in Sec. III B for the example of nanoparticles in the slit-well system (Sec. II B).

Finally, assuming that the system is initialized according to the stationary distribution, the Markov chain central limit theorem can be applied to deduce the distribution of Eqn B1. In general, the Markov chain central limit theorem states that in the limit of large k , for any real-valued function g of the stationary Markov chain state $(\theta_i, t_{i,i+1})$,

$$\rho\left(\sum_{i=0}^{k-1} g(\theta_i, t_{i,i+1})\right) \rightarrow \mathcal{N}(k\langle g(\theta_0, t_{0,1}) \rangle, k\sigma^2). \quad (\text{B5})$$

This result closely resembles the classical central limit theorem. For instance, $\langle g(\theta_0, t_{0,1}) \rangle$ is the ensemble average of $g(\theta_0, t_{0,1})$ taken with respect to the stationary distribution π . However, the quantity σ in Eqn. B5 is not simply the variance of g ; see below.

For the choice $g(\theta_i, t_{i,i+1}) = t_{i,i+1}$, and since $\tau_k = \sum_{i=0}^{k-1} t_{i,i+1}$ and $\tau_1 = t_{0,1}$, it follows that

$$\rho(\tau_k) \rightarrow \mathcal{N}(k\langle \tau_1 \rangle, k\sigma^2). \quad (\text{B6})$$

where $\langle \tau_1 \rangle$ is the mean first passage time across the first periodic subunit when the analytes are initialized according to $\pi(\theta_0)$. The parameter controlling the variance of τ_k is

$$\sigma^2 = \text{var}_\pi(t_{0,1}) + 2 \sum_{i=1}^{\infty} \text{cov}_\pi(t_{0,1}, t_{i,i+1}), \quad (\text{B7})$$

where var_π and cov_π denote variances and covariances, respectively, computed when the system is initialized according to $\pi(\theta_0)$. Since the $t_{i,i+1}$ are all identically distributed, the relationship can be rewritten in the form

$$\sigma^2 = \text{var}_\pi(t_{0,1}) \left[1 + 2 \sum_{i=1}^{\infty} \text{corr}_\pi(t_{0,1}, t_{i,i+1}) \right], \quad (\text{B8})$$

where $\text{corr}_\pi(t_{0,1}, t_{i,i+1})$ are the correlations between distinct crossing times. The first term is the variance of the first passage time across any single periodic subunit. The terms in the series capture the correlations in the passage times $t_{i,i+1}$ across distinct subunits i , which are mediated by the correlations in the degrees of freedom θ_i . In the special case where these correlations are all zero, we recover i.i.d. behaviour in the $\{t_{i,i+1}\}_{i=0}^{k-1}$ and the result reduces to the standard central limit theorem.

3. The number of subunits k that have been crossed at least once at the time t

Consider the (discrete) random variable $\tilde{k}(t)$, the number of channels that the analyte has crossed at least once

at time t . The probability that $\tilde{k}(t)$, exceeds some threshold k is given by

$$P(\tilde{k}(t) \geq k) = P(\tau_k \leq t) = \int_0^t \rho(\tau_k) d\tau_k. \quad (\text{B9})$$

Under the conditions leading to Eqn. B6, this integral can be approximated for large k (or, equivalently, large t) as

$$P(\tilde{k}(t) \geq k) \approx \frac{1}{2} \left(1 + \text{erf} \left[\frac{t - k\langle\tau_1\rangle}{\sqrt{2k\sigma^2}} \right] \right). \quad (\text{B10})$$

From Eqn. B10, the probability mass function of k at any (large) time t can be obtained as

$$P(\tilde{k}(t) = k) = P(\tilde{k}(t) \geq k) - P(\tilde{k}(t) \geq k+1). \quad (\text{B11})$$

However, a more useful form can be deduced by making a discrete-to-continuous approximation, i.e., by pretending that k is a continuous random variable. When k is large, which is the limit of interest, this is an arbitrarily good approximation. Given this, it is therefore sensible to say that the limiting probability density function for k after a long time t is

$$\begin{aligned} \rho_t(k) &\approx -\frac{\partial P(\tilde{k}(t) \geq k)}{\partial k} \\ &= \frac{t + k\langle\tau_1\rangle}{\sqrt{8\pi k^3 \sigma^2}} \exp\left(-\frac{(t - k\langle\tau_1\rangle)^2}{2k\sigma^2}\right). \end{aligned} \quad (\text{B12})$$

Although $\rho_t(k)$ is a probability density function, the corresponding probability mass function is approximately

$$P(\tilde{k}(t) = k) \approx \int_{k-0.5}^{k+0.5} \rho_t(k) dk \approx \rho_t(k) \quad (\text{B13})$$

At long times, $\rho_t(k)$ changes very little from $k - 0.5$ to $k + 0.5$, and this approximation is again arbitrarily good. In other words, $\rho_t(k)$ can be interpreted fairly as the probability that, at time t , the analyte has reached x_k at least once, but has not yet reached x_{k+1} .

4. The position distribution at long times

Eqn. B12 does not directly describe the position of the analyte at a time t . During the time interval after its first passage to x_k and before its first passage to x_{k+1} , the analyte can potentially move to any position with $x < x_{k+1}$. However, as argued below, the distinction between $\tilde{k}(t)$ and $x(t)/L$ is negligible at long times, so Eqn. B12 is in fact an acceptable proxy for the position distribution. The discussion is in the same spirit as that put forth in Reimann et al.⁷³

Consider the analyte's x -position, $x(t)$, in the time interval of duration $t_{k,k+1}$ occurring between τ_k (when it

first reaches x_k) and τ_{k+1} (when it first reaches x_{k+1}). Write $x(t) = x_{k+1} - \delta_x(t)$, such that $\delta_x(t)$ is the distance from the analyte's current position to x_k . Motion of the analyte in the direction of $-\hat{x}$ carries an energetic cost, as it opposes the direction of the applied force. This is in addition to any entropic cost incurred for moving through the periodic MNFD. Thus, the probability of observing the analyte at $x = x_{k+1} - \delta_x$ at any point during this time interval will decrease rapidly when $\delta_x \gg L$.

More importantly, by the periodicity of the system and the stationarity of the Markov chain, it must be that the dynamics of the random variable δ_x do not depend on k (although correlations between consecutive period crossings are possible). Thus, the typical size of δ_x at any time between τ_k and τ_{k+1} is independent of k . For sufficiently large k , the typical distance that an analyte might move in the $-\hat{x}$ direction after reaching x_k and before reaching x_{k+1} is therefore arbitrarily small compared to the total distance it has traveled since $t = 0$. Similarly, the duration $t_{k,k+1}$ will be a small fraction of the total time τ_k . Thus, although the analyte may briefly move short distances away from x_k before reaching x_{k+1} , these fluctuations do not affect the ensemble dynamics in the limit of long time or, equivalently, large k .

Making the substitution $x \approx kL - \delta_x$ in Eqn. B12 yields

$$\begin{aligned} \rho_t(x + \delta_x) &\approx \\ &\frac{Lt + (x + \delta_x)\langle\tau_1\rangle}{\sqrt{8\pi(x + \delta_x)^3(L\sigma^2)}} \exp\left(-\frac{(Lt - (x + \delta_x)\langle\tau_1\rangle)^2}{2(x + \delta_x)(L\sigma^2)}\right) \end{aligned} \quad (\text{B14})$$

Eqn. B14 gives a mean of

$$\langle x(t) \rangle = L \frac{t}{\langle\tau_1\rangle} + L \frac{1}{2} \frac{\sigma^2}{\langle\tau_1\rangle^2} - \langle\delta_x\rangle \quad (\text{B15})$$

and a variance of

$$\text{var}(x(t)) = L^2 \frac{\sigma^2}{\langle\tau_1\rangle^2} \frac{t}{\langle\tau_1\rangle} + L^2 \frac{5}{4} \frac{\sigma^4}{\langle\tau_1\rangle^4} + \text{var}(\delta_x). \quad (\text{B16})$$

As argued above, the statistics of δ_x are roughly independent of time on the timescale of period-to-period transport. In particular, we must have that $\langle\delta_x/t\rangle$ and $\text{var}(\delta_x/t)$ converge to zero for large t .

At long times, we also find that Eqn. B14 converges to a normal distribution (App. B4 a). In this limit, the constant terms in Eqns. B15 and B16 are negligible, and

$$\rho_t(x) \rightarrow \mathcal{N}\left(L \frac{t}{\langle\tau_1\rangle}, L^2 \frac{\sigma^2}{\langle\tau_1\rangle^2} \frac{t}{\langle\tau_1\rangle}\right). \quad (\text{B17})$$

In this form, it is clear that, on long time scales, distance is naturally counted in units of L , and time in units of $\langle\tau_1\rangle$ (see Lindner et al.⁷⁵ for related modelling).

Finally, it follows from Eqns. B15 or B17 that

$$\lim_{t \rightarrow \infty} \frac{\langle x(t) \rangle}{t} = \frac{L}{\langle\tau_1\rangle}. \quad (\text{B18})$$

Dividing both sides by Φ , we recover the desired result: Eqn. 1 for the direct mobility is equivalent to Eqn. 2 for the indirect mobility.

An additional result is that the quantity

$$D_{\text{eff}} = \frac{1}{2} \frac{\sigma^2}{\langle \tau_1 \rangle^2} \frac{L^2}{\langle \tau_1 \rangle} \quad (\text{B19})$$

behaves as an effective diffusion coefficient for the analyte. The ratio $\sigma/\langle \tau_1 \rangle$ is almost the coefficient of variation of τ_1 . It differs in the fact that σ contains corrections due to the correlations between consecutive crossing times (Eqn. B8). Thus, we see that the model nicely reflects how correlations directly impact the dispersion of analyte as they travel through the system.

a. The limiting position distribution at long times is Gaussian

This section contains the derivation that the probability density function in Eqn. B12 converges to the probability density function of a normal distribution. Consider the shifted and scaled variable

$$q = \frac{k - \frac{t}{\mu}}{\frac{\sigma}{\mu} \sqrt{\frac{t}{\mu}}}, \quad (\text{B20})$$

which at large times will have a mean approaching zero and a variance approaching one. This has probability density function

$$\rho(q; t) = \frac{1 + \frac{1}{2} \frac{\sigma}{\sqrt{\mu t}} q}{\sqrt{2\pi(1 + \frac{\sigma}{\sqrt{\mu t}} q)^3}} \exp\left(-\frac{q^2}{2(1 + \frac{\sigma}{\sqrt{\mu t}} q)}\right). \quad (\text{B21})$$

When considering values of q that are small compared to $\frac{\sqrt{\mu t}}{\sigma}$, the distribution will be very close to its limiting form of

$$\rho(q; t) \rightarrow \frac{1}{\sqrt{2\pi}} \exp\left(-\frac{q^2}{2}\right) = \mathcal{N}(0, 1). \quad (\text{B22})$$

At any fixed t , no matter how large, the distribution of q will differ from this limiting form for sufficiently large q , i.e., in the distant tails of the distribution. However, given that the distribution is normalized, at very large t the total probability assigned to these distant tails will be vanishingly small. The same derivation applies to the distribution in terms of $x + \delta_x$, rather than k (Eqn. B14).

Appendix C: Convergence analysis

In this section, we will analyse and compare the numerical properties of the direct and indirect mobility formulations. Specifically, we will examine the computational cost of estimating each kind of mobility using molecular

dynamics simulations of the analyte moving through the periodic geometry. Certain simplifying assumptions will be needed in order to advance the analysis. Most importantly, the analysis of the direct mobility will neglect the dynamics of δ_x (App. B4), and the analysis of the indirect mobility will be based on the assumption that first passage times are exponentially distributed at long times. The predicted scaling behaviours will be compared to numerical results on the example described in Sec. II B.

The underlying simulation implementation is assumed to be identical between the two cases, except for boundary conditions and termination conditions. In particular, we won't consider the convergence of numerical error with respect to the discretization scheme. The error introduced by discretizing the equations of motion depend on the discretization scheme. Better schemes can be combined with either mobility formulation, and this consideration is essentially orthogonal to the comparison being made here. Of course, it is possible that the discretization error propagates differently in the simulations that would be used to calculate direct and indirect mobilities. Thus, in practice, some residual numerical error will always exist between the two.

1. Convergence of direct mobility

The direct mobility is typically estimated as

$$\hat{\mu}_{\text{direct}} = \frac{1}{\Phi t_{\text{direct}}} \left(\frac{1}{N_{\text{direct}}} \sum_{j=1}^{N_{\text{direct}}} x^{(j)} \right), \quad (\text{C1})$$

where, N_{direct} molecules are simulated (independently) for a long period of time t_{direct} and the final states are used to estimate the direct mobility. In practice, t_{direct} is commonly chosen approximately as the time after which at least a certain number of periods $k_{\text{direct}} \approx t_{\text{direct}}/\langle \tau_1 \rangle$ will have been traversed on average.

Eqns. B15 and B16 from App. B, allow us to predict the limiting behaviour of the relative error between $\hat{\mu}_{\text{direct}}$ and the true direct mobility μ_{direct} . For large t_{direct} , each particle's position is identically and independently normally distributed with mean and variance given by Eqns. B15 and B16. The relative error is thus also normally distributed, with

$$\text{mean} \left(\frac{\hat{\mu}_{\text{direct}} - \mu_{\text{direct}}}{\mu_{\text{direct}}} \right) \approx \left(\frac{1}{2} \frac{\sigma^2}{\langle \tau_1 \rangle^2} - \frac{\langle \delta_x \rangle}{L} \right) \frac{1}{k_{\text{direct}}}, \quad (\text{C2})$$

$$\text{stderr} \left(\frac{\hat{\mu}_{\text{direct}} - \mu_{\text{direct}}}{\mu_{\text{direct}}} \right) \approx \frac{\sigma}{\langle \tau_1 \rangle} \frac{1}{\sqrt{N_{\text{direct}} k_{\text{direct}}}}, \quad (\text{C3})$$

where we have ignored terms of order $\mathcal{O}(1/(k_{\text{direct}} \sqrt{N_{\text{direct}}}))$ in the standard error. The mean relative error depends on the behaviour of $\langle \delta_x \rangle$, which is outside the scope of the present study. In

the following discussion, we will consider the simple case of $\langle \delta_x \rangle \approx 0$, which we expect to be reasonable for highly driven systems. As we will see in the numerical demonstrations (Sec. III C), $\langle \delta_x \rangle$ plays an important role in weakly driven systems.

The mean relative error (Eqn. C2) indicates a bias due to the finite simulation time with which the direct mobility is being estimated. It cannot be reduced except by increasing k_{direct} , and decays at a rate of $\mathcal{O}(1/k_{\text{direct}})$. Conversely, the standard relative error (Eqn. C3) captures the intrinsic noise in the mobility estimator. This decays as $\mathcal{O}(1/\sqrt{k_{\text{direct}}})$, which is slower than the decay of the mean relative error. Thus, for sufficiently long run-times, the direct mobility estimator will be statistically indistinguishable from an unbiased estimator.

The limiting behaviour of the direct mobility estimator is jointly affected by k_{direct} and N_{direct} . A reasonable choice (see, e.g., Mark and Baram⁸²) for a single scalar error is the square root of the expected square of the relative error between $\hat{\mu}_{\text{direct}}$ and the true direct mobility μ_{direct} :

$$\epsilon_{\text{direct}}^2 := \mathbb{E} \left[\left(\frac{\hat{\mu}_{\text{direct}} - \mu_{\text{direct}}}{\mu_{\text{direct}}} \right)^2 \right] \approx \text{bias}^2 + \text{stderr}^2, \quad (\text{C4})$$

where bias is the mean relative error given by Eqn. C2 and stderr is the standard relative error given by Eqn. C3.

2. Convergence of indirect mobility

Estimating the indirect mobility is tantamount to estimating the mean first passage time of particles crossing a single period starting from the initial conditions $x = x_0$ and $\theta_0 \sim \pi(\theta)$. We will assume that the stationary distribution $\pi(\theta_0)$ is known and/or can be sampled efficiently. A careful cost analysis of this sampling process is relegated to future work.

First, let us consider estimating the indirect mobility using

$$\hat{\mu}_{\text{indirect}} = \frac{L}{\Phi} \left(\frac{1}{N_{\text{indirect}}} \sum_{j=1}^{N_{\text{indirect}}} \tau_1^{(j)} \right)^{-1}, \quad (\text{C5})$$

which is based on sampling N_{indirect} instances $\tau_1^{(j)}$ of the first passage time. It may be difficult in general to ascertain whether this is an unbiased estimator of the indirect mobility. To make progress on this and other questions, we propose that it is reasonable to assume that the first passage time across one period is roughly exponentially distributed at long times. Thus, throughout the rest of this discussion, we will assume

$$\rho(\tau_1) \approx \frac{1}{\tau^*} \exp\left(-\frac{\tau_1}{\tau^*}\right) \quad (\text{C6})$$

at large τ_1 , where τ^* is some constant. Heuristically, this will be the case for systems in which the first passage process converges to a steady-state behaviour after initial transient behaviour decays. It is consistent with the spirit of Kramers theory (App. A 2), since we are restricting our attention to the residual first passage process on long timescales (compared to all other timescales of relaxation in the system). This can be justified more rigorously for many typical systems by considering the behaviour of the eigenfunctions of the Smoluchowski equation in the presence of an absorber (e.g., as done by Grigoriev et al.⁸³ when studying the narrow escape problem); in this case, τ^* will be the first eigenvalue of the PDE. In particular, although the equivalence of indirect and direct mobilities was proven in App. B for any system, including those where transient dynamics are non-negligible, our error analysis in this section will apply to convergence rates in the limit of long times, after transient phenomena have abated.

In the case of an exponential distribution, it is known that the simple estimate used in Eqn. C5 is indeed a maximum likelihood estimator, but is nonetheless biased. Specifically, the limiting relative error in the mobility estimator for exponentially distributed first passage times is simply

$$\text{mean} \left(\frac{\hat{\mu}_{\text{indirect}} - \mu_{\text{indirect}}}{\mu_{\text{indirect}}} \right) \approx \frac{1}{N_{\text{indirect}}}. \quad (\text{C7})$$

In any case, since the N_{indirect} samples of $\tau_1^{(j)}$ are independent, we can estimate the standard error of the indirect mobility estimator in Eqn. C5. The error in the estimate of $\langle \tau_1 \rangle$ will go as

$$\text{stderr} \left(\langle \tau_1 \rangle - \frac{1}{N_{\text{indirect}}} \sum_{j=1}^{N_{\text{indirect}}} \tau_1^{(j)} \right) \approx \frac{\sigma_0}{\sqrt{N_{\text{indirect}}}}, \quad (\text{C8})$$

where we have introduced the notation $\sigma_0 := \text{stddev}(\tau_1)$ to indicate the standard deviation of τ_1 . Propagating the uncertainty therefore yields that

$$\text{stderr} \left(\frac{\hat{\mu}_{\text{indirect}} - \mu_{\text{indirect}}}{\mu_{\text{indirect}}} \right) = \frac{\sigma_0}{\langle \tau_1 \rangle} \frac{1}{\sqrt{N_{\text{indirect}}}}. \quad (\text{C9})$$

Note that, as expected, the standard error of the indirect mobility estimator scales as $\mathcal{O}(1/\sqrt{N_{\text{indirect}}})$. This bias is negligible relative to the standard error for sufficiently large N_{indirect} . The total error will therefore also converge as

$$\epsilon_{\text{indirect}} \approx \frac{\sigma_0}{\langle \tau_1 \rangle} \frac{1}{\sqrt{N_{\text{indirect}}}}. \quad (\text{C10})$$

In our experience, the coefficient of variation is of order one, and the bias is thus at most a 1% correction to Eqn. C10.

3. Comparing convergence: Entirely serial computation

First, let us analyse the runtimes of the direct and indirect mobility estimators in the case where all computations are performed in serial. Then the total runtime for the direct mobility estimator will be proportional to

$$T_{\text{direct}} = t_{\text{direct}} N_{\text{direct}} \approx \langle \tau_1 \rangle k_{\text{direct}} N_{\text{direct}}, \quad (\text{C11})$$

up to a constant factor based on the implementation of the simulations (i.e., the real time elapsed per unit of simulation time simulated). We will assume for the rest of this section that this proportionality factor is some constant, and omit it from the discussion.

The error of the direct mobility estimator can be decreased by increasing either of k_{direct} or N_{direct} . The runtime is linear in each of these, and the standard relative error depends equally on both quantities. However, the bias depends only on k_{direct} , and so the best choice in this circumstance is to fix $N_{\text{direct}} = 1$. Thus, using Eqns. C11 and C4, the runtime necessary to reach a small relative error ϵ scales as

$$\frac{T_{\text{direct}}}{\langle \tau_1 \rangle} \approx \frac{1}{2} \frac{\sigma^2}{\langle \tau_1 \rangle^2} \frac{1}{\epsilon^2} \left(1 + \sqrt{1 + \epsilon^2}\right) \rightarrow \frac{\sigma^2}{\langle \tau_1 \rangle^2} \frac{1}{\epsilon^2}. \quad (\text{C12})$$

The term $1 + \sqrt{1 + \epsilon^2}$ is very nearly equal to two for reasonable values of ϵ (say, below 10%).

Similarly, the total runtime for the indirect mobility estimator will be approximately proportional to

$$T_{\text{indirect}} \approx \langle \tau_1 \rangle N_{\text{indirect}} \quad (\text{C13})$$

for large N_{indirect} . Using Eqns. C13 and C10, we find that its runtime will therefore grow as

$$\frac{T_{\text{indirect}}}{\langle \tau_1 \rangle} \approx \frac{\sigma_0^2}{\langle \tau_1 \rangle^2} \frac{1}{\epsilon^2}. \quad (\text{C14})$$

Therefore, in the case of purely serial computing, the two formulations are nearly identical. Both have runtimes of order $\mathcal{O}(1/\epsilon^2)$. The convergence of the direct mobility estimator is influenced by crossing time correlations (via σ), where the indirect mobility estimator's is not. In highly correlated systems, the direct mobility estimator will exhibit more error than the indirect mobility estimator. However, correctly sampling the stationary distribution required for the indirect mobility estimator may become more difficult in such systems. In any case, it appears that neither algorithm is clearly advantageous when computations are done in serial.

4. Comparing convergence: Entirely parallel computation

Now, suppose instead that all sampling of trajectories will be computed entirely in parallel. Although the total number of floating point operations will be the same as in the case of serial computation, in the case of parallel

computation it is often of more interest to consider the total elapsed time from the start of the algorithm to the termination of the last parallel thread of the computation.

For the direct mobility estimator, every parallel thread has the same fixed runtime (up to fluctuations in the computing speed). Thus, the total time elapsed will simply be proportional to the duration of each trajectory:

$$T_{\text{direct}} = t_{\text{direct}} \approx \langle \tau_1 \rangle k_{\text{direct}}. \quad (\text{C15})$$

The standard relative error will be vanishingly small, so the error will be dominated by the bias. Thus, the runtime will now converge as

$$\frac{T_{\text{direct}}}{\langle \tau_1 \rangle} \approx \frac{1}{2} \frac{\sigma^2}{\langle \tau_1 \rangle^2} \frac{1}{\epsilon}. \quad (\text{C16})$$

This is substantially better than in the case of purely serial computation (Eqn. C12), and is now of order $\mathcal{O}(1/\epsilon)$.

Assessing the total time elapsed for the indirect mobility estimator is more complicated. Because the indirect mobility is independently sampling the first passage time, the runtime of each sample is itself a stochastic quantity. In the case of serial computation, the central limit theorem ensures that the estimate in Eqn. C13 will be fairly accurate for large N_{indirect} . However, if trajectories are computed entirely in parallel, then the elapsed time from start to finish will be dictated by the sample of the maximum first passage time, rather than the mean first passage time:

$$T_{\text{indirect}} \approx \max_{N_{\text{indirect}}} (\tau_1^{(j)}). \quad (\text{C17})$$

Naturally, the sample maximum will depend on the simulated ensemble size N_{indirect} .

The typical maximum first passage time can be estimated if we again suppose that, at long times, the first passage time is exponentially distributed. Setting the cumulative distribution of Eqn. C6 equal to $1 - (1/N_{\text{indirect}})$ yields the following estimate for the time at which the last particle will escape:

$$\frac{1}{N_{\text{indirect}}} \approx \exp\left(-\frac{T_{\text{indirect}}}{\tau^*}\right) \quad (\text{C18})$$

$$\Rightarrow T_{\text{indirect}} \approx \tau^* \ln(N_{\text{indirect}}). \quad (\text{C19})$$

A more rigorous justification for this estimate of the runtime can be obtained by considering the distribution of the maximum of an ensemble of i.i.d. exponentially distributed variables. By making appeal to Poisson processes and harmonic numbers, one can recover again the logarithmic scaling of T_{indirect} with N_{indirect} . Incidentally, when the coefficient of variation of the first passage time is greater than one, the argument to the logarithm in Eqn. C19 should in fact be the fraction of the population belonging to long tails of the distribution. However, because the dependence is logarithmic, the effect of this correction is small.

Note that this perspective implicitly assumes that all of the parallel computation hardware remains reserved for the respective calculations until all trajectories are completed. This is in fact true for the direct mobility estimator, since all parallel computations will have the same runtime. Conversely, most of the samples generated towards the indirect mobility estimator will have runtimes much smaller than the maximum first passage time. Thus, T_{indirect} is certainly an overestimate of the computational cost in settings where parallel computing resources can be repurposed dynamically as soon as these samples terminate.

In any case, using Eqn. C19 for the runtime with Eqn. C10 for the error, we see that the indirect mobility estimator converges exponentially as

$$\epsilon \approx \frac{\sigma_0}{\langle \tau_1 \rangle} \exp \left(-\frac{1}{2} \frac{T_{\text{indirect}}}{\tau^*} \right). \quad (\text{C20})$$

Equivalently,

$$T_{\text{indirect}} \approx 2\tau^* \ln \left(\frac{\sigma_0}{\langle \tau_1 \rangle} \frac{1}{\epsilon} \right). \quad (\text{C21})$$

Thus, as ϵ becomes small and so $1/\epsilon$ becomes larger, the required runtime grows only logarithmically. This will be exponentially faster than the convergence of the direct mobility estimator for small ϵ .

The results of this analysis have some caveats. The prefactor in Eqn. C16 is likely overestimated in general, because $\langle \delta_x \rangle$ has been ignored. Meanwhile, it is possible that $\tau^* \gg \langle \tau_1 \rangle$. These corrections will tend to improve the relative performance of the direct mobility estimator against the indirect mobility estimator, as is indeed observed in the numerical demonstrations of Sec. III C. Regardless, because the indirect mobility converges exponentially (Eqn. C20) whereas the direct mobility converges as $T_{\text{direct}} \sim \mathcal{O}(1/\epsilon)$ (Eqn. C16), these corrections are only important in comparing behaviour at moderately large ϵ . When unlimited parallel computation is available, the indirect mobility formulation will always be much more efficient at sufficiently small ϵ .

5. Comparing convergence: Limited parallel computation

In practice, of course, unlimited parallelization is not feasible. Suppose that N_{para} samples can comfortably be simulated in parallel. For ensemble sizes larger than this, calculations must be broken into batches of N_{para} .

In this case, the direct mobility estimator's runtime will scale as

$$T_{\text{direct}} \approx \langle \tau_1 \rangle k_{\text{direct}} \left\lceil \frac{N_{\text{direct}}}{N_{\text{para}}} \right\rceil, \quad (\text{C22})$$

where $\lceil \cdot \rceil$ denotes the ceiling function. Because of the ceiling function, the runtime does not increase at all with the number of parallel trajectories until N_{direct} reaches

an integral multiple of N_{para} . Thus, the best choice of N_{direct} is certainly at least N_{para} . However, going from $N_{\text{direct}} = N_{\text{para}}$ to $N_{\text{direct}} = 2N_{\text{para}}$ increases the runtime by a factor of two while leaving the bias (Eqn. C2) unchanged. Increasing k_{direct} by a factor of two would have the same impact on runtime and standard relative error, but would also decrease the bias. Mirroring the reasoning from App. C3, we thus find that the optimal choice is precisely $N_{\text{direct}} = N_{\text{para}}$.

Using Eqn. C4 with $N_{\text{direct}} = N_{\text{para}}$ and Eqn. C22 yields the runtime necessary to attain a target accuracy ϵ :

$$\frac{T_{\text{direct}}}{\langle \tau_1 \rangle} = \frac{1}{2} \frac{\sigma^2}{\langle \tau_1 \rangle^2} \frac{1}{\epsilon^2} \frac{1}{N_{\text{para}}} \left(1 + \sqrt{1 + (\epsilon N_{\text{para}})^2} \right). \quad (\text{C23})$$

This is similar to the result for serial computation (Eqn. C12), but differs in two places. First, the prefactor of $1/N_{\text{para}}$ corresponds to the acceleration of convergence by a factor of N_{para} in the small- ϵ limit. Here, the error is dominated by noise (Eqn. C3) and $\epsilon \sim \mathcal{O}(1/\sqrt{T_{\text{direct}}})$. Increasing the number of independent samples is essentially as beneficial as increasing k_{direct} by the same amount.

On the other hand, the term $(\epsilon N_{\text{para}})^2$ inside the square root of Eqn. C23 corresponds to an acceleration at larger values of ϵ . Since the computational cost does not increase with N_{indirect} until $N_{\text{direct}} = N_{\text{para}}$, the convergence for $N_{\text{direct}} < N_{\text{para}}$ is essentially the same as in the case of unlimited parallel computation. Specifically, when $\epsilon N_{\text{para}} \gg 1$

$$1 + \sqrt{1 + (\epsilon N_{\text{para}})^2} \approx \epsilon N_{\text{para}}, \quad (\text{C24})$$

which implies that

$$\frac{T_{\text{direct}}}{\langle \tau_1 \rangle} \approx \frac{1}{2} \frac{\sigma^2}{\langle \tau_1 \rangle^2} \frac{1}{\epsilon}. \quad (\text{C25})$$

In other words, when $N_{\text{direct}} < N_{\text{para}}$, the error is dominated by bias (Eqn. C2), and the convergence is of order $\epsilon \sim \mathcal{O}(1/T_{\text{direct}})$.

Consider now the indirect mobility estimator. Its runtime in the case of limited parallel computation depends on the manner in which it is implemented. We will consider two approaches. To enable these more complicated analyses, we will assume first passage times are distributed exponentially as per Eqn. C19 with $\tau^* \approx \langle \tau_1 \rangle$. Accounting for deviations from a single exponential distribution makes these algorithms more difficult to analyse. Conservative approximations can be obtained by increasing the runtime estimates by $\tau^*/\langle \tau_1 \rangle$.

First, consider an algorithm for computing the indirect mobility estimator in which N_{indirect} trajectories are initiated at once and evolved in time at the same rate. The first N_{para} trajectories are incremented by one timestep, then the next N_{para} are incremented once, and so on until all trajectories have been incremented once. Early in the

simulation, it will take $\lceil N_{\text{indirect}}/N_{\text{para}} \rceil$ passes to increment all trajectories by one timestep. As the simulation advances and some events terminate, fewer passes will be required to increment time. In this case, the runtime will scale as

$$\frac{T_{\text{indirect}}}{\langle \tau_1 \rangle} \approx \ln(N_{\text{last}}) + \quad (\text{C26})$$

$$\sum_{k=1}^{M-1} \ln \left(\frac{N_{\text{indirect}} - (k-1)N_{\text{para}}}{N_{\text{indirect}} - kN_{\text{para}}} \right) (M - (k-1)) \quad (\text{C27})$$

$$= (4M - 2) \ln(M) - \ln(M!) + \ln(N_{\text{last}}), \quad (\text{C28})$$

$$\approx (3M - 2) \ln(M) + M + \frac{1}{2} \ln(2\pi M) + \ln(N_{\text{last}}), \quad (\text{C29})$$

where $M = \lceil N_{\text{indirect}}/N_{\text{para}} \rceil$ and $N_{\text{last}} = N_{\text{indirect}} - (M-1)N_{\text{para}}$. The factor of $\ln(N_{\text{last}})$ accounts for the maximum first passage time in that final batch. The approximation in Eqn. C29 is based on Stirling's approximation, and reveals that this runtime is $\mathcal{O}(M \ln(M))$.

Consider next an alternative implementation of the indirect mobility estimator with limited parallel computation. In this case, the simulation begins by initializing only N_{para} trajectories. Whenever a trajectory terminates, a new trajectory is initiated on the same thread, until a total of N_{indirect} have been initiated. In this case, the runtime will scale as

$$\frac{T_{\text{indirect}}}{\langle \tau_1 \rangle} \approx \frac{N_{\text{indirect}} - N_{\text{min}}}{N_{\text{para}}} + \ln(N_{\text{min}}), \quad (\text{C30})$$

where $N_{\text{min}} = \min(N_{\text{indirect}}, N_{\text{para}})$. The first term estimates the time until a total of N_{indirect} trajectories have been initiated, and the second term estimates the time required for the simulations to terminate thereafter. This algorithm's runtime is $\mathcal{O}(N_{\text{indirect}})$.

Surprisingly, this second implementation is consistently faster than the first one. There are two reasons for this. Firstly, it ensures that no parallel computing threads are idle until the very last batch of simulations, where some idling is inevitable. More important, however, is that the second algorithm allows for a natural balancing of fast and slow events across different threads. Threads on which events terminate quickly will more quickly be re-initialized with new events. Conversely, in the first algorithm all events are simulated independently, so that the maximum first passage time over all N_{indirect} events factors into the overall runtime. Worse, the speed at which these long trajectories are simulated is impaired by a factor of $\mathcal{O}(M)$ for most of the runtime. We will proceed with the analysis of the second, faster algorithm, but we include the analysis of the first version here as a warning to the reader.

Equation C30 can be written more explicitly as

$$\frac{T_{\text{indirect}}}{\langle \tau_1 \rangle} \approx \begin{cases} \ln(N_{\text{indirect}}) & N_{\text{indirect}} \leq N_{\text{para}} \\ \frac{N_{\text{indirect}}}{N_{\text{para}}} + \ln(N_{\text{para}}) - 1, & N_{\text{indirect}} > N_{\text{para}} \end{cases}. \quad (\text{C31})$$

Reversing Eqn. C10, we find that the number of samples necessary for the indirect mobility estimator to achieve a target relative error ϵ is

$$N_{\text{indirect}} = \frac{\sigma_0^2}{\langle \tau_1 \rangle^2} \frac{1}{\epsilon^2}. \quad (\text{C32})$$

Substituting Eqn. C32 into Eqn. C31 yields the runtime required to achieve the target accuracy. When $N_{\text{indirect}} \leq N_{\text{para}}$,

$$\frac{T_{\text{indirect}}}{\langle \tau_1 \rangle} \approx 2 \ln \left(\frac{\sigma_0}{\langle \tau_1 \rangle} \frac{1}{\epsilon} \right). \quad (\text{C33})$$

This is equivalent to Eqn. C20. In this regime, the error behaves as if there were unlimited parallelism, and so decreases exponentially with runtime.

Conversely, when $N_{\text{indirect}} > N_{\text{para}}$,

$$\frac{T_{\text{indirect}}}{\langle \tau_1 \rangle} \approx \frac{\sigma_0^2}{\langle \tau_1 \rangle^2} \frac{1}{\epsilon^2} \frac{1}{N_{\text{para}}} + \ln(N_{\text{para}}) - 1. \quad (\text{C34})$$

This is very similar to the result for entirely serial computation (Eqn. C14), but accelerated by a factor of N_{para} . The $\ln(N_{\text{para}})$ term arises here because the large parallel batches are more vulnerable to rare long-duration events. However, this term is negligible in the limit of $N_{\text{indirect}} \gg N_{\text{para}}$ and the error scales as $\mathcal{O}(1/\sqrt{T_{\text{indirect}}})$, as expected. In fact, on the limit of small target error and/or small N_{para} , the two mobility formulations are once again essentially equivalent up to a factor of $\psi = \sigma/\sigma_0$.

Figure 7 in the main body of the paper summarizes the theoretically predicted ratio of runtimes for the direct and indirect mobilities; ψ was factored out from the direct mobility runtime. Figure 7(a) shows the predicted ratio as a function of the final relative error ϵ . The colours blue, orange, green correspond to $N_{\text{para}} = 10^5, 10^4, 10^3$, respectively. A typical modern consumer-grade GPU can effectively deliver tens of thousands of parallel threads, justifying $N_{\text{para}} = 10^4$ for single-body molecule simulations. Parallelizing across ten such GPUs is reasonably economical in many cases, motivating the case $N_{\text{para}} = 10^5$. Conversely, many-body molecules will reduce the number of independent simulation that can be conducted in parallel, which motivated the choice of $N_{\text{para}} = 10^3$. The different line styles correspond to coefficients of variation equal to 3 (solid), 1 (dashed), and 0.5 (dotted), which are values encountered in the example system from Sec. IIB²³. The lines are truncated at the values of ϵ such that either k_{direct} or N_{indirect} would be required to equal 10 or less; the various modelling assumptions certainly do not apply in that regime.

Altogether, the theoretical analysis predicts that the indirect mobility estimator will converge up to 2-70 times faster than the direct mobility estimator under these circumstances. The ϵ of maximum relative advantage is in the range of 0.1-1%. This is often a perfectly acceptable error threshold for assisting with the research and design of periodic MNFDs, as modelling errors are often larger than this. As noted, at very small target relative errors the estimators are essentially equivalent (i.e., the ratio converges to 1).

Figures 7(b,c) highlight the expected transitions of the direct and indirect mobility estimators, respectively, from parallel-like scaling to serial-like scaling. For the direct mobility estimator, this occurs at $\epsilon \approx 1/N_{\text{para}}$,

where for the indirect mobility estimator it occurs near $\epsilon \approx (\sigma_0/\langle\tau_1\rangle)/\sqrt{N_{\text{para}}}$. The ϵ value of maximum relative advantage for the indirect mobility estimator is expected to occur somewhere between these two points.

As noted at the end of App. C 3, the net effect of the correlation factor ψ does not clearly favour either algorithm. Although it directly amplifies the predicted runtime of the direct mobility estimator, it likely also increases the cost of sampling the stationary distribution. In any case, it does not appear that large correlation functions are likely to occur in most applications, so this effect is likely to be modest. If there are applications of interest where ψ is found to be large, this aspect of the algorithm should be investigated in more detail.

Chapter 7

Conclusions

The work outlined in this thesis has explored the opportunity to use the neural network method (NNM) to facilitate the use of complicated partial differential equation (PDE) models in computational biophysics. These PDEs are often high-dimensional, and traditionally such models are prohibitively expensive to solve numerically. As such, models based on stochastic differential equations (SDEs) are dominant in contemporary computational biophysics. However, PDEs have some appealing advantages over SDEs, most notably the possibility of using model order reduction techniques to study parameterized models more efficiently. The NNM appears to be remarkably well-suited for solving high-dimensional PDEs, and moreover can readily perform nonlinear model order reduction. It is thus very tempting to imagine the NNM as a powerful new tool in the computational biophysicist's toolkit.

These expectations must be tempered by the limitations of our current understanding of the NNM. It is unclear why or how it is so effective at solving high-dimensional or highly parameterized PDEs. Moreover, there are no theoretical guarantees available today ensuring that it will perform reliably on a

given PDE problem. In practice, the studies in Chapter 4 and Apps. C and D encountered significant challenges when attempting to apply the NNM naively to PDE models of typical biophysical systems. These challenges were eventually understood to arise from the irregularities created by the non-convex domain geometry. This directly contradicts the prior wisdom that the NNM is particularly effective at handling complex geometries due to its mesh-free character. It also serves as an important warning for future research into using the NNM in biophysics, where such geometries are common.

These technical challenges were overcome and reliable performance of the NNM was recovered. Moreover, the derivation in App. B was produced to show that the NNM loss functional acts as an *a posteriori* error estimator to bolster practical confidence in the method. Despite this, the convergence rate of the NNM remains a confusing matter. NNM approximation error decreases rather slowly with increases in the number of degrees of freedom and also with increases in the training time. Moreover, the error appears to be limited to some finite minimum that is much larger than machine precision.

The discussion in Chapter 5 suggests a possible resolution to this paradox. It appears that the representations learned by each hidden layer of the NNM trial function converge very consistently to some fixed set of features that is independent of width (for widths above some threshold). Thus, the effective number of degrees of freedom of the NNM is much lower than the actual number of degrees of freedom required to specify all the weights and biases of the architecture. Moreover, this effective capacity converges to a finite value at large widths. This is consistent with observations in Chapter 4 and Apps. C and D.

These results may imply that the standard NNM formulation is limited to approximations that are only moderately accurate but, nonetheless, that it can reliably attain this moderate level of accuracy.

Chapter 4 and App. C explored the idea of using the NNM to solve for external electric fields that act as force fields in subsequent molecular dynamics simulations. The main motivation put forth was to use the NNM approximations as low-memory, GPU-compatible representations of high-dimensional functions capturing nonlinear force fields that depend dynamically on the molecular conformation. Once training issues arising from the irregular solution were accounted for, the NNM was shown to be able to reliably produce force fields of acceptable accuracy for this application. Unfortunately, the slow convergence of the NNM with respect to the actual number of degrees of freedom (and therefore actual memory consumption) undermines the intended use for this application. However, the discussion in Chapter 5 suggests that this memory utilization could potentially be dramatically improved by reducing the width of hidden layers to the size that actually contributes to solution accuracy.

Motivated by the studies of the time-integrated Smoluchowski in Chapter 3, the work in App. D identified another promising application of the NNM in computational biophysics. Rather than using the NNM to approximate the force fields used in molecular dynamics simulations, the NNM can be used to solve the corresponding time-integrated Smoluchowski equation directly. Moreover, the nonlinear model order reduction capabilities of the NNM are used to solve this problem directly as a function of problem parameters. The NNM is shown to reliably approximate the resulting four-dimensional function with relative ease. In

particular, the appropriate integral of this solution provides a smooth approximation of the end-to-end mapping from problem input parameters directly to a key observable for periodic MNFD design: the mean first passage time and an associated mobility-like quantity.

Finally, Chapter 6 extends the analysis of Chapter 3 to demonstrate that the electrical mobility can in fact be expressed exactly in terms of the reciprocal of an appropriately computed mean first passage time. This provides a firm theoretical foundation for applying the technique in App. D to study electrical mobilities in any MNFD that operates by driving molecules through a periodic geometry. As an added bonus, the derivations in that analysis reveal that computing the electrical mobility via the mean first passage time can actually be substantially more computationally efficient even for measurements based on molecular dynamics simulations, under certain reasonably common conditions.

In summary, this thesis has presented works that investigate exciting opportunities for exploiting the NNM to solve traditionally intractable PDE models in biophysics. Two applications were identified in which such a capability could be particularly beneficial. The first opportunity is to produce memory-efficient representations of nonlinear force fields coupled to molecular conformations for subsequent use in molecular dynamics simulations. The theory-to-practice gap of the NNM leads to unexpectedly large memory consumption, undermining this application; however, the discussion in Chapter 5 suggests possible ways of overcoming this. The second opportunity is the direct solution of parameterized many-body time-integrated Smoluchowski equations for studies of effective electrical mobility through periodic MNFDs. App. D

demonstrates a successful proof-of-concept of this approach, and the work in Chapter 6 clarifies how to pose the physical models correctly to match traditional electrical mobility definitions. The results in this thesis suggest that there is ample opportunity for future development, both theoretical and applied, towards capitalizing on the unique potential of the NNM as a tool in computational biophysics.

Bibliography

- [1] Ivo Babuska and Barna Szabo. “On the rates of convergence of the finite element method”. In: *International Journal for Numerical Methods in Engineering* 18.3 (1982), pp. 323–341.
- [2] Christian Beck, Weinan E, and Arnulf Jentzen. “Machine learning approximation algorithms for high-dimensional fully nonlinear partial differential equations and second-order backward stochastic differential equations”. In: *Journal of Nonlinear Science* 29.4 (2019), pp. 1563–1619.
- [3] Peter Benner et al. *Model reduction and approximation: Theory and algorithms*. SIAM, 2017.
- [4] Jens Berg and Kaj Nyström. “A unified deep artificial neural network approach to partial differential equations in complex geometries”. In: *Neurocomputing* 317 (2018), pp. 28–41.
- [5] Kyle Briggs et al. “DNA translocations through nanopores under nanoscale preconfinement”. In: *Nano Letters* 18.2 (2018), pp. 660–668.
- [6] Kuang-Ling Cheng et al. “Electrophoretic size separation of particles in a periodically constricted microchannel”. In: *The Journal of Chemical Physics* 128.10 (2008), p. 101101.

-
- [7] George Cybenko. “Approximation by superpositions of a sigmoidal function”. In: *Mathematics of control, signals and systems* 2.4 (1989), pp. 303–314.
 - [8] Ronald A DeVore, Ralph Howard, and Charles Micchelli. “Optimal nonlinear approximation”. In: *Manuscripta mathematica* 63.4 (1989), pp. 469–478.
 - [9] M. W. M. G. Dissanayake and N. Phan-Thien. “Neural-network-based approximations for solving partial differential equations”. In: *Communications in Numerical Methods in Engineering* 10.3 (1994), pp. 195–201.
 - [10] Weinan E, Jiequn Han, and Arnulf Jentzen. “Deep learning-based numerical methods for high-dimensional parabolic partial differential equations and backward stochastic differential equations”. In: *Communications in Mathematics and Statistics* 5.4 (2017), pp. 349–380.
 - [11] Weinan E and Bing Yu. “The deep Ritz method: a deep learning-based numerical algorithm for solving variational problems”. In: *Communications in Mathematics and Statistics* 6.1 (2018), pp. 1–12.
 - [12] Alexandre Ern and Jean-Luc Guermond. *Theory and practice of finite elements*. Vol. 159. Springer, 2004.
 - [13] Xavier Glorot and Yoshua Bengio. “Understanding the difficulty of training deep feedforward neural networks”. In: *Proceedings of the thirteenth international conference on artificial intelligence and statistics*. 2010, pp. 249–256.

-
- [14] Philipp Grohs and Felix Voigtlaender. “Proof of the theory-to-practice gap in deep learning via sampling complexity bounds for neural network approximation spaces”. In: *arXiv preprint arXiv:2104.02746* (2021).
 - [15] Philipp Grohs et al. “A proof that artificial neural networks overcome the curse of dimensionality in the numerical approximation of Black-Scholes partial differential equations”. In: *arXiv preprint arXiv:1809.02362* (2018).
 - [16] Jiequn Han, Arnulf Jentzen, and Weinan E. “Solving high-dimensional partial differential equations using deep learning”. In: *Proceedings of the National Academy of Sciences* 115.34 (2018), pp. 8505–8510.
 - [17] Kaiming He et al. “Deep residual learning for image recognition”. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2016, pp. 770–778.
 - [18] Kurt Hornik. “Approximation capabilities of multilayer feedforward networks”. In: *Neural networks* 4.2 (1991), pp. 251–257.
 - [19] Martin Hutzenthaler et al. “A proof that rectified deep neural networks overcome the curse of dimensionality in the numerical approximation of semilinear heat equations”. In: *SAM Research Report 2019* (2019).
 - [20] Arnulf Jentzen, Diyora Salimova, and Timo Welti. “A proof that deep artificial neural networks overcome the curse of dimensionality in the numerical approximation of Kolmogorov partial differential equations with constant diffusion and nonlinear drift coefficients”. In: *arXiv preprint arXiv:1809.07321* (2018).

-
- [21] Diederik P. Kingma and Jimmy Ba. “Adam: A method for stochastic optimization”. In: *arXiv preprint arXiv:1412.6980* (2014).
 - [22] Peter E Kloeden and Eckhard Platen. “Stochastic differential equations”. In: *Numerical Solution of Stochastic Differential Equations*. Springer, 1992, pp. 103–160.
 - [23] Isaac E. Lagaris, Aristidis Likas, and Dimitrios I. Fotiadis. “Artificial neural networks for solving ordinary and partial differential equations”. In: *IEEE Transactions on Neural Networks* 9.5 (1998), pp. 987–1000.
 - [24] Michelle H Lam et al. “Entropic trapping of DNA with a nanofiltered nanopore”. In: *ACS Applied Nano Materials* 2.8 (2019), pp. 4773–4781.
 - [25] Yann LeCun et al. “Efficient backprop”. In: *Neural networks: Tricks of the trade*. Springer, 1998, pp. 9–50.
 - [26] Martin Magill, Andrew M Nagel, and Hendrick W de Haan. “Neural network solutions to differential equations in nonconvex domains: Solving the electric field in the slit-well microfluidic device”. In: *Physical Review Research* 2.3 (2020), p. 033110.
 - [27] Martin Magill, F. Z. Qureshi, and H. W. de Haan. “Compact neural network solutions to Laplace’s equation in a nanofluidic device”. In: *Workshop on Compact Deep Neural Network Representations with Industrial Applications* (2018).
 - [28] Martin Magill, Ed Waller, and Hendrick W de Haan. “A sequential nanopore-channel device for polymer separation”. In: *The Journal of chemical physics* 149.17 (2018), p. 174903.

-
- [29] Martin Magill et al. “Translocation time through a nanopore with an internal cavity is minimal for polymers of intermediate length”. In: *Physical Review Letters* 117.24 (2016), p. 247802.
- [30] B. Ph. van Milligen, V. Tribaldos, and J. A. Jiménez. “Neural Network Differential Equation and Plasma Equilibrium Solver”. In: *Physical Review Letters* 75.20 (1995), p. 3594.
- [31] Marzieh Alireza Mirhoseini and Matthew J Zahr. “Model reduction of convection-dominated partial differential equations via optimization-based implicit feature tracking”. In: *arXiv preprint arXiv:2109.14694* (2021).
- [32] Nagel, Andrew M. and Magill, Martin and de Haan, Hendrick W. “Studying First Passage Problems using Neural Networks: A Case Study in the Slit-Well Microfluidic Device”. In: *Physical Review E* 106.2 (2022), p. 025311.
- [33] Yurii Nesterov. *Introductory lectures on convex optimization: A basic course*. Vol. 87. Springer Science & Business Media, 2003.
- [34] Vladimir V Palyulin, Tapio Ala-Nissila, and Ralf Metzler. “Polymer translocation: the first two decades and the recent diversification”. In: *Soft Matter* 10.45 (2014), pp. 9016–9037.
- [35] E. A. J. F. Peters and Th. M. A. O. M. Barenbrug. “Efficient Brownian dynamics simulation of particles near walls. I. Reflecting and absorbing walls”. In: *Physical Review E* 66.5 (2002), p. 056701.
- [36] Sidney Redner et al. *A guide to first-passage processes*. Cambridge University Press, 2001.

-
- [37] Takahiro Sakaue. “Dynamics of Polymer Translocation: A Short Review with an Introduction of Weakly-Driven Regime”. In: *Polymers* 8.12 (2016).
- [38] Justin Sirignano and Konstantinos Spiliopoulos. “DGM: A deep learning algorithm for solving partial differential equations”. In: *Journal of Computational Physics* 375 (2018), pp. 1339 –1364. ISSN: 0021-9991.
- [39] Suvrit Sra, Sebastian Nowozin, and Stephen J Wright. *Optimization for machine learning*. Mit Press, 2012.
- [40] M Von Smoluchowski. “Contribution to the theory of electro-osmosis and related phenomena”. In: *Bull. Int. Acad. Sci. Cracovie* 3 (1903), pp. 184–199.
- [41] Uwe Winter and Tihamér Geyer. “Coarse grained simulations of a small peptide: Effects of finite damping and hydrodynamic interactions”. In: *The Journal of Chemical Physics* 131.10 (2009), p. 104102.
- [42] Neha Yadav, Anupam Yadav, and Manoj Kumar. *An introduction to neural network methods for differential equations*. Springer, 2015.
- [43] Jason Yosinski et al. “How transferable are features in deep neural networks?” In: *Advances in Neural Information Processing Systems* 27 (2014).
- [44] Olek C Zienkiewicz, Robert L Taylor, and Jian Z Zhu. *The Finite Element Method: Its Basis and Fundamentals*. Elsevier, 2005.

Appendix A

Numerical methods

A.1 Particle simulations

A.1.1 Basic algorithms

Langevin equations of motion (e.g., Eqn. 2.10) are stochastic differential equations (SDEs): the derivatives of position with respect to time are described in terms of a random function $R(t)$. This has implications for the manner in which the equations can be solved numerically.

The simplest numerical discretization for SDEs is known as the Euler-Maruyama method [22]. It is a straightforward generalization of the simple Euler method for discretizing ordinary differential equations (ODEs). With this method, the discretized version of the overdamped Langevin equation (Eqn. 2.10) is

$$\gamma \frac{x_{i+1} - x_i}{\Delta t} = qE(x_i) + \sqrt{\frac{2\gamma k_B T}{\Delta t}} R_i, \quad (\text{A.1})$$

where Δt is the discrete timestep, x_i is the position after i timesteps, and R_i is a

random variable drawn from a standard normal distribution independently at every timestep i .

Solving for x_{i+1} , we get

$$x_{i+1} = x_i + qE(x_i)\Delta t + \sqrt{2D\Delta t}R_i. \quad (\text{A.2})$$

The first two terms on the RHS resemble the standard Euler discretization of a deterministic equation of overdamped motion under the influence of the force qE . In particular, the term $qE(x_i)\Delta t$ is linear in Δt , which is equivalent to assuming a constant velocity of $qE(x_i)$ during each timestep. In contrast, the random term $\sqrt{2D\Delta t}R_i$ only grows in proportion to the square root of Δt . This is essentially a manifestation of the central limit theorem: the total random position update $\sqrt{2D\Delta t}R_i$ is effectively the summation of many independent identically distributed random position updates occurring throughout the timestep Δt .

This discussion of the Euler-Maruyama method is sufficient to illustrate the qualitative differences that arise when numerically treating SDEs versus ODEs. In practice, the Euler-Maruyama method is sufficient for many simple biophysical systems, and is used in parts of Chapters 3, 4, and 6, as well as App. D.

When solving more challenging biophysical SDE models, the Euler-Maruyama discretization of the overdamped Langevin equation is not always adequate. Unfortunately, extending ODE solvers to corresponding SDE solvers is increasingly difficult for higher-order solvers. When simulating the dynamics of generic polymer models, motion occurs over a wide range of timescales, and approximating the dynamics effectively with the Euler-Maruyama of the overdamped model requires excessively small choices of Δt . A common approach

is to discretize the full (i.e., not overdamped) Langevin equation with a stochastic variant of the velocity Verlet method [41]. This approach tends to exhibit better numerical stability and accuracy without substantially increasing the numerical cost. This methodology is applied in my works on polymer translocation through nanopores, including the microscopic model in Chapter 3 and the related works in Briggs et al. [5] and Lam et al. [24] of which I am a co-author.

A.1.2 Rates of convergence

A.1.2.1 Timestep

Typically, the approximate solutions produced by a numerical SDE solver are expected to converge to the true solution of the SDE as the size of the discrete timestep goes to zero. The convergence of numerical SDE solvers is a more subtle concept than the convergence of comparable ODE solvers. Two main versions of convergence are commonly discussed: strong convergence and weak convergence. Strong convergence quantifies the rate at which each trajectory of a numerical SDE solver converges to the correct trajectory. Weak convergence essentially quantifies the rate at which the statistical properties of the ensemble of trajectories converge to the true statistics of the SDE solution. In biophysics, physical observables of interest are almost exclusively ensemble statistics, and so weak convergence rates are of primary interest.

The Euler-Maruyama method has a weak convergence rate of 1 [22]. For instance, the error in the ensemble mean of position after some given time T decreases as Δt to the power of 1. This is comparable to the deterministic Euler method for ODEs, which also has an error convergence rate of 1. However, it is

worth noting that the strong convergence rate of the Euler-Maruyama method is only 0.5: the error in any given trajectory generated by the solver only decays in proportion to the square root of Δt .

A.1.2.2 Geometry

An additional subtlety arises when discretizing SDE models for biophysical systems in confinement: the implementation of boundary conditions. Naively, purely reflective boundary conditions can be implemented by explicitly repositioning particle trajectories that exit the domain at reflective boundaries. Purely absorbing boundary conditions can be implemented by considering absorption to have occurred for trajectories that would exit across such a boundary. Implementing these conditions is more complicated in domains with intricate geometries. Moreover, as discussed by Peters and Barenbrug [35], SDE trajectories near walls may in actuality interact with boundary conditions even when discretized trajectories begin and end within the domain. Neglecting such phenomena can reduce the weak convergence rate of the Euler-Maruyama method for the overdamped Langevin equation to 0.5 near perfectly reflecting or absorbing boundary conditions. The improved integration schemes proposed in that work are not utilized in this thesis, so it should be assumed that most of the simulations herein have limiting convergence rates proportional to $\sqrt{\Delta t}$.

The implementation of boundary conditions is the primary pathway via which the complexity of the problem geometry affects the computational cost of particle simulations. Domain decomposition methods are commonly used, so that boundary interactions are only evaluated for particles that are reasonably near

the edge of the domain. Thanks to such methodologies, particle simulations can very efficiently handle very complicated geometries. Indeed, the ease with which particle simulations can be implemented in domains with complicated geometries is historically one of the major motivations for using SDE models over PDE models.

A.1.2.3 Ensemble size

As noted above, biophysical research is primarily concerned with statistical properties of the SDE models. Thus, it is also important to consider the rate at which estimates of these statistics converge with respect to the ensemble size N . Ultimately, this convergence is governed by statistical analysis of the observable being studied; see Chapter 6 for examples of such an analysis. In the simplest case of measuring the mean of some stochastic quantity, the error convergence will tend to be dictated by the central limit theorem. That is, the standard error of the estimator will generally decay in proportion to \sqrt{N} . Each of the N trajectories in the ensemble is entirely independent; this makes the calculation of these statistical estimators embarrassingly parallel, which is another enormous advantage of particle simulations.

A.1.2.4 Many-body systems

Finally, it is important to contemplate the performance of particle systems in many-body systems as a function of the number of interacting particles n . Essentially, the computational cost to estimate an ensemble statistic to a given level of accuracy always scales in proportion to \sqrt{N} , where N is the number of

trajectories in the statistical ensemble. The prefactor for this relationship depends primarily on the computational cost of advancing the SDE solver by one timestep. This growth of this cost with n is typically dominated by the evaluation of the inter-particle interactions.

If long-ranged two-body interactions occur between all particles in the system, then the cost of evaluating these forces grows as n^2 . Examples of such force fields include certain models for hydrodynamic interactions as well as unscreened Coulomb interactions. In practice, n^2 scaling is considered very expensive; luckily, biophysical models can typically be massively simplified to essentially eliminate the need to compute long-ranged interactions between all particles. For instance, Coulomb interactions are typically screened by the action of intervening free charges (i.e., dissolved ions in the solvent); accurate field models can be implemented by explicitly considering only the long-ranged interactions between particles within some small cut-off radius, and approximate interactions farther than this. Similarly, neighbour list algorithms are used to accelerate the computation of short-ranged interactions by anticipating which pairs have essentially no probability of interacting in a given timestep. In many cases, the growth of computational costs can be limited to the order of $n \log(n)$.

In any case, it is generally the case that the computational cost of particle simulations grows at most polynomially with n . This is in stark contrast to the cost of traditional solvers to PDE models, for which the computational cost grows exponentially with n (Sec. A.2). As a result, the numerical solvability of PDE models is typically restricted to $n \approx 1$, whereas particle simulations can be extended (with some effort) to systems in which n is in the millions. For instance,

the many-body model studied in Chapter 3 ranges up to $n \approx 100$ and uses a very simple molecular model, and yet would already be virtually impossible to solve using methods like FEM. This can be seen as a fundamental reason for the current widespread success of particle simulations and SDE models in biophysics. It is also one of the main reasons that the NNM, which may have the ability to solve PDE models for large n (Sec. A.3), is such an exciting technique.

A.2 The finite element method

A.2.1 Basic algorithms

The finite element method (FEM) is one of the most common numerical methods for solving partial differential equations in domains with complex geometries, such as those arising in MNFDs. In this approach, the shape of the domain is decomposed into a mesh of simple geometric primitives (such as triangles or tetrahedra). On each of these mesh elements is defined a simple, usually piecewise polynomial function. Together, these simple functions form a basis (in the sense of a vector space) spanning the space of all trial functions that can be used to approximate the true solution. That is, the unknown solution u of the PDE problem is approximated by trial functions of the form

$$\tilde{u} = \sum_i c_i \phi_i, \tag{A.3}$$

where each ϕ_i is a simple piecewise polynomial function over one of the mesh elements, and c_i are the coordinates of \tilde{u} in the basis formed by the ϕ_i .

The basic tasks of FEM are to construct a good mesh over the domain, define a good basis of trial functions using this mesh, and find coordinates in that basis that give a good approximation to the (unknown) true solution to the PDE. The meaning of “good” for each of these choices often depends on the PDE to be solved and the resources available. However, some basic perspectives are common to the majority of FEM implementations.

A.2.1.1 Mesh generation

The quality of a mesh can be gauged in many ways depending on the desired application. For instance, it is often desirable for FEM meshes to be constructed out of simple geometric primitives with roughly isotropic shapes, as elements that are very elongated in one direction tend to produce ill-conditioned algorithms.

An even more important measure of quality in an FEM mesh is that all important geometric features of the PDE solution be adequately resolved. Of course, this is challenging to ensure since the true solution is unknown *a priori*. Approximate solutions can provide some guess as to the structure of the true solution, but these approximate solutions can only be obtained by first constructing meshes of unknown quality. In any case, the computational cost of FEM grows with the quantity and complexity of the geometric primitives in the mesh, so it is often desired to keep these at a minimum.

Automatically constructing a good mesh for a complicated geometry is generally considered a difficult task. There exist robust algorithms for automatically generating triangular or tetrahedral meshes with some quality guarantees in arbitrary two- and three-dimensional geometries. However, these meshes may be sub-optimal; in many cases, human mesh designers can produce higher-quality and lower-cost meshes more efficiently than automated geometry meshers.

In industrial practice today, it is often the case that mesh development is an expensive task that requires substantial human labour. Anecdotally, the cost of the human time spent manually perfecting geometric meshes is frequently far more significant than the computational cost of the subsequent FEM calculation.

Such a claim is difficult to verify in a precise way, and doing so is certainly outside the scope of this thesis. However, it is a fundamentally important consideration when comparing FEM to deep learning techniques like the neural network method (NNM) of solving PDEs (Sec. A.3). As identified in Chapter 4 and App. D, the NNM struggles to converge to better accuracies than 0.1% or so, and tends to bear a higher computational cost than FEM at the same level of precision. However, the NNM is totally mesh-free: it can handle arbitrary geometries with minimal human oversight. Moreover, the modelling errors in most applications substantially exceed 0.1% in the first place. Now, the performance of the NNM is expected to continue improving with algorithmic refinements and advances in hardware, and it is possible that the NNM will become competitive with FEM as a result of such progress. However, a more fundamental question may be: if errors on the order of 0.1% are acceptable in practice, are the savings in human labour on mesh generation afforded by the NNM more beneficial than modest increases in computational cost relative to FEM? Such a question cannot be assessed by numerical analysis and computational experiments alone, and require us to maintain a more holistic view of these algorithms as tools to accomplish specific goals.

A.2.1.2 The method of weighted residuals

Let us now assume that the domain of the PDE problem has been decomposed into a good mesh and that a basis for the trial functions has been defined on that mesh. The next goal is to identify coefficients c_i in Eqn. A.3 that make \tilde{u} a good approximation to the true solution u . This is more challenging than simple

interpolation because the true solution u is not known *a priori*. The coefficients must be selected using only the implicit description of u that is afforded by the data of the PDE problem (i.e., the PDE and its boundary conditions).

The method of weighted residuals is a very general framework for approaching such problems. Consider a PDE of the form

$$Lu(x) = f(x), \tag{A.4}$$

where L is a linear differential operator and $f(x)$ is a source term over the domain Ω . Define

$$R[\tilde{u}](x) \equiv L\tilde{u}(x) - f(x) \tag{A.5}$$

which will be called the residual of the PDE; this is a measure of the amount by which \tilde{u} fails to satisfy the PDE at the point x . By definition, the true solution has a residual of zero everywhere in the domain Ω . Conversely, if the PDE is well-posed, there is only one function that satisfies the boundary conditions and that also has a residual of zero everywhere in the domain.

To simplify the following discussion, we will assume that the trial functions are selected so as to exactly satisfy the boundary conditions. This can often be accomplished exactly or to a very good approximation for practical FEM algorithms, as explained briefly later in this section. In a more general setting, we may wish to define residuals for the boundary conditions as well as the PDE itself. In fact, this is done in the standard formulation of the NNM (Sec. [A.3](#)).

Generally, the solution to a non-trivial PDE problem cannot be expressed

exactly in a given FEM trial function basis. In other words, there is no set of coordinates c_i for which $R[\tilde{u}](x) = 0$ at all $x \in \Omega$. Instead, the method of weighted residuals sets a more tractable criterion: for some suitable set of weighting functions $w_j(x)$, find the coordinates c_i such that

$$\langle R[\tilde{u}], w_j \rangle \equiv \int_{\Omega} R[\tilde{u}](x) w_j(x) dx = 0 \quad (\text{A.6})$$

for all w_j . Equation A.6 is the weighted integral of the residual R over the problem domain, and it is from this that the method of weighted residuals derives its name.

Equivalently, Eqn. A.6 is the inner product (in L_2) of R with the weighting functions w_j . For this reason, the weighting functions w_j are also called test functions. For each j , Eqn. A.6 tests the residual R in a certain dimension of function space, and we will select c_i so as to pass all these tests. If Eqn. A.6 is satisfied for all j , then the residual of the trial function \tilde{u} is orthogonal to the space spanned by the test functions.

Because we are concerned with \tilde{u} of the form in Eqn. A.3, the residual can be expressed in the same basis as \tilde{u} . The RHS of Eqn. A.6 can be reduced as

$$\langle R[\tilde{u}], w_j \rangle = \langle L\tilde{u} - f, w_j \rangle \quad (\text{A.7})$$

$$= \left\langle L \left(\sum_i c_i \phi_i \right) - f, w_j \right\rangle \quad (\text{A.8})$$

$$= \left(\sum_i c_i \langle L\phi_i, w_j \rangle \right) - \langle f, w_j \rangle. \quad (\text{A.9})$$

The condition from Eqn. A.6 that this be equal to zero becomes

$$\sum_i c_i \langle L\phi_i, w_j \rangle = \langle f, w_j \rangle. \quad (\text{A.10})$$

In practice, the LHS of this equation is usually simplified further using integration by parts to reduce the order of the highest derivatives by one.

Equation A.10 is in fact a discretization of the weak formulation of the PDE, which describes the solution u as the function that satisfies

$$\langle Lu, w \rangle = \langle f, w \rangle \quad (\text{A.11})$$

for all possible test functions w in the appropriate Sobolev space. In particular, the weak formulation of a PDE is usually posed in terms of an infinite-dimensional space of test functions w . Thus, we see that the method of weighted residuals is equivalent to discretizing the weak formulation of the PDE in two ways: the space of trial function is approximated using the basis of ϕ_i , and the space of test functions is approximated by the basis w_j .

For a given choice of ϕ_i and w_j , Eqn. A.10 is a linear system of equations of the form

$$Ac = b, \quad (\text{A.12})$$

whose solution c gives the coordinates of the trial function \tilde{u} that has no weighted residuals for this choice of test functions. The entries of A and b are obtained by evaluating the corresponding integrals in Eqn. A.10; in practice these integrals

are obtained using numerical quadrature methods. Different trial and test functions are appropriate for different applications, but generally these are chosen so that the system in Eqn. A.10 can be solved efficiently and accurately.

One of the classical choices is called the Galerkin method, in which the sets of test functions and trial functions are chosen to be the same: $\{\phi_i\}_i = \{w_j\}_j$. Moreover, the FEM typically chooses as trial functions piecewise polynomial functions with compact support (i.e., that are only non-zero over one or a few geometric primitives of the domain mesh). As such, most of the inner products in the LHS of Eqn. A.10 are zero. The resulting system of equations is sparse, and such systems can be solved far more efficiently than general dense systems of equations.

The above discussion has mostly overlooked the consideration of boundary conditions. Typically, FEM literature distinguishes between essential boundary conditions and natural boundary conditions. Natural boundary conditions emerge naturally from the system of equations in Eqn. A.10; they usually correspond to some neutral behaviour of the solution derivatives at the boundaries. Essential boundary conditions are not implied by Eqn. A.10, and must be implemented manually. These can be introduced into the linear system by replacing the appropriate rows of Eqn. A.10 with the equation of the boundary condition instead of the equation based on setting the weighted PDE residual to zero. In this way, it is typically straightforward to guarantee that FEM solutions of linear PDEs will satisfy all boundary conditions to a very high level of precision.

The above is an overview of FEM in its simplest incarnation; many more sophisticated variants exist. For time-dependent problems, FEM is typically

applied to solve the spatial problem once per timestep; discretization of time is handled separately using ODE techniques. Some PDE problems do not respond well to the standard Galerkin method (i.e., choosing $\{\phi_i\}_i = \{w_j\}_j$) and require more careful selection of the test functions. Nonlinear PDEs often produce nonlinear systems of equations, which must be solved via techniques like Newton's method. Some PDEs can be formulated in variational forms based on minimizing an energy functional; these are often solved with FEM methods that minimize that energy functional rather than weighted residuals. These many variants are beyond the present scope.

A.2.1.3 Mixed formulation

In this thesis, the FEM is frequently used to solve for the electric fields in MNFDs. Specifically, with the electrostatic potential modelled by Laplace's equation $\nabla^2 u = 0$, the goal is to numerically approximate the electric field $E = -\nabla u$. The basic FEM formulation described in Sec. A.2.1.2 is known to be vulnerable to pathological behaviour for such an application. Essentially, the methodology of Sec. A.2.1.2 is capable of producing approximate solution \tilde{u} such that

$$\tilde{u} \approx u \tag{A.13}$$

but

$$\nabla \tilde{u} \not\approx \nabla u. \tag{A.14}$$

Note that here $\nabla \tilde{u} = \sum_i c_i \nabla \phi_i(x)$ is the exact gradient of the trial function \tilde{u} . In practice, using the simple FEM formulation of Sec. A.2.1.2 on the electric field models in Chapters 3 and 4 and Apps. C and D was found to yield electric field approximations with extremely large electric field values near the re-entrant boundaries of the domain.

A very successful remedy to this problem is to reformulate the equations in a so-called mixed form [44]. We will illustrate this here in the special case of Laplace's equation. Rather than applying the FEM directly to approximating the electrostatic potential u , the mixed formulation proposes to numerically approximate both u and E separately. Let us approximate u by the (scalar) trial function \tilde{u} as before, but now also define a new (vector) trial function \tilde{E} with which to approximate E directly. Laplace's equation can be written in term of E as

$$\nabla \cdot E = 0 \tag{A.15}$$

throughout the interior of the problem domain. Neumann boundary conditions can be written in terms of $E \cdot \hat{n}$ on the corresponding boundaries. Dirichlet boundary conditions, however, cannot be written conveniently in terms of E alone, and must be written in terms of u . Thus, we see that it is not possible to solve for E only, and rather we must solve for u simultaneously. The two fields are connected by the condition

$$-\nabla u = E \tag{A.16}$$

throughout the domain.

In essence, we can now apply a method of weighted residuals in which \tilde{E} minimizes the weighted residual of Eqn. A.15 and \tilde{E} and \tilde{u} simultaneously minimize the weighted residual of Eqn. A.16. Neumann conditions are enforced on \tilde{E} and Dirichlet conditions on \tilde{u} . Each of the trial functions \tilde{E} and \tilde{u} is expressed in a separate basis of piecewise polynomial functions with compact support. As with the basic formulation of Sec. A.2.1.2, projecting the equations onto test functions yields a sparse system of linear equations that can be solved for coordinates describing \tilde{E} and \tilde{u} .

A.2.1.4 Model order reduction

Often, practical PDE models are expressed in terms of some problem parameters p . That is, the PDE is of the form

$$L(p)u(x;p) = f(x;p), \quad x \in \Omega(p), \quad (\text{A.17})$$

indicating that the differential operator L , the source term f , and even the domain Ω may all depend on the problem parameters p . The boundary conditions may similarly be functions of p . As a result, the solution u naturally also depends on p .

Often, it is of great practical interest to understand how the solutions to these parameterized PDE models depend on p . Perhaps some natural phenomenon only occurs for certain values of p . Maybe the performance of some engineered system is optimal for some choice of p . Sometimes a certain value of p corresponds to an equilibrium state of the system, and we want to know how the system responds to

perturbations in p . Similarly, we may wish to understand how errors or statistical uncertainties in p propagate into errors or uncertainties in the PDE solution.

In many cases, it is very expensive to solve the parameterized PDE model repeatedly at all values of p required. Recall that each solution of the PDE may require the generation of a mesh, the selection of appropriate trial and test bases, the evaluation of inner products determining the system of equations for the FEM model, and finally the solution of the resulting linear or nonlinear system of equations. Depending on the details of the model, some of these steps may only need to be completed once. For instance, if the geometry is independent of p then a single properly constructed mesh may be sufficient for many choices of p . Nonetheless, the cost is often still quite high.

The cost of repeatedly solving parameterized PDE models is especially prohibitive in applications where a large number of p values must be considered in a limited amount of time. For instance, in the setting of real-time simulations, parameterized PDE models must be solved quickly enough to respond to changing data measurements from a real physical system. In computational biophysics, an important example is that of sophisticated force field models in molecular dynamics simulations. These force fields can depend on the state of the entire MD system at every moment in time, and the time spent computing the forces easily becomes the bottleneck to the overall computational cost of the simulation. Precomputing the PDE solutions across a large number of values of p ahead of time is not a trivial matter either. Naively generating candidate values of p by combining many values of each parameter of p leads to a combinatorially large number of cases as the number of parameters in p increases.

Essentially, model order reduction techniques attempt to facilitate the solution of parameterized PDE models by interpolating between a small number of high-resolution solutions precomputed over reference values of p [3]. That is, the solution at some new value of p would be approximated by a trial function of the form

$$\tilde{u}(x; p) = \sum_k d_k \tilde{u}(x; p_k). \quad (\text{A.18})$$

Here, each $\tilde{u}(x; p_k)$ is an FEM approximation to the PDE problem with problem parameters set to p_k , and the d_k are coefficients for linearly interpolating to new p values. There are many techniques for choosing the d_k , or even the reference values of p_k , which together define different approaches to model order reduction. However, most classical model order reduction techniques essentially amount to some form of linear interpolation as described here.

The ability to perform model order reduction is nominally a substantial advantage of PDE-based modelling over SDE-based modelling. There is no straightforward analog for interpolating between ensemble a SDE trajectories precomputed at a handful of p values in order to efficiently compute the solutions at a new value of p . Specifically, parameterized molecular dynamics models must be simulated entirely independently at every relevant choice of p .

In theory, linear model order reduction can sometimes be quite effective. The effectiveness of these methods is often discussed in terms of the Kolmogorov width, which captures the worst-case error that can be achieved using the best possible basis of a given size n . Mirhoseini and Zahr [31] includes a survey of various articles discussing the convergence of Kolmogorov width with n . This

error can decay exponentially for elliptic and parabolic equations, and thus model order reduction can be effective for such problems. Conversely, it can converge as slowly as $n^{-1/2}$ for convection-dominated problems, and model order reduction with a linear basis is intrinsically ineffective for such systems.

Moreover, the Kolmogorov width describes the best achievable error. It does not account for the difficulty in actually determining a near-optimal linear approximation. In practice, a very large number of solutions may be required to make model order reduction work well even if the Kolmogorov width decays rapidly in theory.

A more fundamental barrier to the use of linear model order reduction techniques based on FEM is that it cannot easily handle parameterized geometries. If the domain geometry $\Omega = \Omega(p)$ varies with the problem parameters, then a new FEM mesh must be computed at every value of p . Moreover, it is unclear how to unambiguously interpolate between reference solutions $\tilde{u}(x; p_k)$ when these functions are not defined on the same domains for x . Some model order reduction techniques have been proposed for handling parameterized geometries (see discussion in App. D), but these formulations are typically highly problem-specific. In contrast, the study in App. D shows that the NNM (which is both mesh-free and nonlinear) can readily handle parameterized PDE models in which p affects the domain geometry.

A.2.2 Rates of convergence

Quantifying the convergence of FEM approximations can be complicated. The sources of error typically include:

- The approximation of the geometry by a finite mesh;
- The approximation of the solution by a piecewise polynomial function;
- The approximation of the integrals in Eqn. [A.10](#) using numerical quadrature;
- The numerical method used to solve the system of equations (linear or nonlinear) defining the coordinates of the trial function.

All of these errors must be considered in the context of computations made using finite precision. The magnitudes of these errors are particularly affected by:

- The dimensionality of the domain;
- The complexity of the shape of the domain;
- The specific functional forms chosen for the trial and test function bases;
- The regularity of the true solution to the PDE.

Conversely, we might be interested to know the rate at which these errors decrease with respect to:

- The quantity, shapes, or sizes of geometric primitives in the mesh;
- The polynomial degrees in the trial or test function bases;
- The total number of degrees of freedom in the trial function \tilde{u} ;
- The total computation time required to obtain the FEM solution.

Additionally, decreasing the error can also incur costs in terms of human factors, for example due to increased difficulty of implementation or additional human labour required for mesh generation.

The rate at which some of these errors decrease with respect to these variables can only be deduced for certain problems and in certain limits. In practice, error convergence is often assessed empirically by trial and error. Nonetheless, it is very helpful to understand the available theoretical results on convergence. The discussion in this section is meant as a simple review of the convergence results that are most relevant to the work in this thesis. In particular, the presentation below is applicable to time-independent linear coercive elliptic PDEs, which include all the Laplace, Poisson, and Smoluchowski equations arising in this thesis.

First, consider the case of a standard (not mixed) FEM approximation to a scalar PDE problem. Let us define the error function

$$e \equiv u - \tilde{u}, \tag{A.19}$$

where u is the (scalar) true solution and \tilde{u} is an FEM approximation of the form in Eqn. A.3 obtained by the method of weighted residuals. Recall that \tilde{u} is not necessarily the best approximation of u in the span of the trial function basis. However, a famous result called Céa's lemma reveals that

$$\|e\| \leq C_1 \inf_w \|u - w\|, \tag{A.20}$$

where the infimum is taken over all w that can be represented in the trial

function basis used to obtain \tilde{u} . In other words, $\inf_w \|u - w\|$ is the best error that could possibly be achieved using the chosen basis. The coefficient C_1 depends on the details of the PDE problem, but not on the choice of mesh or FEM basis. Thus, although \tilde{u} may not be the best representation of u in the chosen basis, its error is no worse than the best possible error multiplied by a problem-dependent factor C_1 . Céa's lemma has certain technical requirements¹, but these can be omitted from the discussion here for simplicity.

Directly analysing the best achievable error $\inf_w \|u - w\|$ is difficult in general. However, there is a special value of w for which powerful convergence results are available. Specifically, define w^* as the projection of the true solution u into the space spanned by the trial function basis ϕ_i , given by

$$w^*(x) = \sum_i c_i^* \phi_i(x), \quad (\text{A.21})$$

$$c_i^* = \frac{\langle u, \phi_i \rangle}{\|\phi_i\|}. \quad (\text{A.22})$$

The function w^* is also called the interpolation of u in this FEM basis. It is certainly true that

$$\inf_w \|u - w\| \leq \|u - w^*\| \quad (\text{A.23})$$

¹Specifically, the result requires that the bilinear form arising in the weak formulation of the PDE be continuous and coercive in the same norm arising in Eqn. A.20. In that case, the coefficient C_1 can be expressed as the ratio of its constants of continuity of coercivity. The PDEs solved in this thesis satisfy these conditions in the $H^0 = L_2$ and H^1 norms (at least), which is sufficient for our current purposes.

and thus that

$$\|e\| \leq C_1 \|u - w^*\|. \quad (\text{A.24})$$

We can now appeal to results on the rate of convergence of the interpolation of a function onto an FEM basis. This relationship depend on the regularity r of the function being interpolated. Classically, the regularity of a function (also called its smoothness) is measured by the number of times it is continuously differentiable; the space C^k denotes functions that are k times differentiable. More appropriate definitions of regularity for the solutions of PDEs are defined in terms of the number of times a function is weakly differentiable (see Sobolev spaces). In fact, it is even possible to define non-integer notions of regularity. For present purposes, however, the reader unfamiliar with fractional regularity can simply understand that r is a number that indicates how many times a function is (weakly) differentiable.

It can be shown that the interpolation error of a function u with regularity r being interpolated onto an FEM basis of piecewise polynomials of degree p on a mesh whose cells have maximum diameter² h will scale as

$$\|u - w^*\| \leq C_2 h^{\min(r, p+1)}. \quad (\text{A.25})$$

Here the norm $\|\cdot\|$ is the standard L^2 norm; similar results apply for Sobolev norms of varying order. The constant C_2 depends on the PDE domain, the specifics of the mesh, the order of the polynomial interpolants, and the regularity

²The diameter of a cell is the maximum distance between any two points in the cell.

of the true solution u . However, C_2 does not depend directly on h , and the exponent of h is typically considered the rate of convergence of the interpolation.

Combining Eqn. A.25 with Eqn. A.24 yields finally that

$$\|e\| \leq Ch^{\min(r,p+1)}, \quad (\text{A.26})$$

where $C = C_1C_2$. Although this results applies only for a specific FEM variant, the convergence of FEM error is typically a power law of roughly this form. See Ern and Guermond [12] for analogous results for mixed FEM formulations³.

Equation A.26 suggests two ways to improve the accuracy of the FEM solution. The first option is to decrease h by using a higher-resolution mesh, which is known as h -refinement. The second option is to increase p by using a trial function basis of larger polynomial order. Babuska and Szabo [1] carefully compares the convergence of these two methods of refinement. In practice, practitioners of FEM typically combine the two types of refinement as appropriate for the problem at hand.

It is also helpful to express the convergence rate in terms of the total number of degrees of freedom in the FEM approximation, N . Typically,

$$h \sim N^{\frac{1}{d}} \quad (\text{A.27})$$

³Essentially, the errors on both sides Eqn. A.24 are replaced with the sum of the errors for the scalar and vector components of the approximation, and then interpolation errors are used as upper bounds for each of their best approximation errors.

where d is the dimensionality of the PDE domain. Thus, if we use sufficiently high-order trial functions such that $p > r$, we see that

$$\|e\| \sim \mathcal{O}\left(N^{-\frac{r}{d}}\right) \quad (\text{A.28})$$

for small $\|e\|$. From this, we can see that smoother functions (with larger r) can be more efficiently approximated by FEM.

Conversely, the cost of attaining a given error grows rapidly with the problem dimensionality d . In fact, for large d

$$N \sim \mathcal{O}\left(\|e\|^{-\frac{d}{r}}\right). \quad (\text{A.29})$$

Thus, for functions of a fixed regularity r , the number of degrees of freedom required to approximate a target function to a given accuracy $\|e\|$ grows *exponentially* with the problem dimensionality d . This is the curse of dimensionality for FEM, and essentially precludes the use of FEM approximation for high-dimensional PDEs.

In fact, Eqn. [A.28](#) is a specific instance of a far more general result from approximation theory. DeVore, Howard, and Micchelli [\[8\]](#) proved that *any* approximation scheme seeking to approximate functions of regularity r in a domain of dimensionality d using N degrees of freedom will exhibit errors ϵ that are at least as large as

$$\epsilon \sim \mathcal{O}\left(N^{-\frac{r}{d}}\right). \quad (\text{A.30})$$

This result only requires that the approximation scheme depend continuously on the approximated function; that is, if the target function is perturbed slightly, the values assigned by the approximation scheme to the degrees of freedom only change slightly. In particular, the method of DeVore, Howard, and Micchelli [8] even applies to nonlinear approximation scheme (including deep learning methods like the NNM).

The DeVore convergence rate essentially reflects the intrinsic complexity of the class of functions of regularity r . If the approximated functions can exhibit wild fluctuations in their derivatives at any points in the problem domain, then it is very hard to anticipate their behaviour based on partial information. Conversely, highly regular functions are easily predicted based on small amounts of information⁴. The problem of representing irregular functions is exacerbated in higher dimensions; intuitively, this is because the function has more volume in which to behave unpredictably.

Actually, despite the far-reaching applicability of DeVore's result, it is possible to achieve (much) better error convergence even using the humble FEM approach outlined above. As discussed by Babuska and Szabo [1], the error of FEM can actually converge exponentially faster with the number of degrees of freedom. The key is to combine h -refinements of the mesh with p refinements of the trial function basis in an adaptive manner tailored to the irregularities of the function being approximated. Essentially, a small number of high- p elements should be used in regions of the domain where the target function is well-behaved. In

⁴The reader may find it helpful to think of the identity theorem for analytic functions in complex analysis, noting that analytic functions are even more regular than infinitely differentiable functions.

regions where the function is irregular, the mesh should be thoroughly refined and a large number of piecewise linear elements should be used. This type of approximation scheme is referred to as hp-FEM.

The exponential convergence of hp-FEM paradoxically overcomes the general limits placed by DeVore. How is this possible? The key to this result is that the hp-FEM approximation is adaptive. DeVore's result applies to approximation schemes that are to be applied to the entire set of functions of regularity r . An adaptive hp-FEM solution tailored to one specific function of regularity r would surely achieve terrible approximation rates for most other functions of regularity r . By incorporating prior knowledge of the function's irregularity, or by deducing this knowledge iteratively through repeated FEM approximations, hp-FEM is able to allocate its degrees of freedom far more efficiently.

This result sets a very important precedent for the NNM, introduced in the next section, and for deep learning methods in general. Deep learning methods are frequently referred to as being intrinsically adaptive. In a sense, this is true: deep neural networks are very flexible approximation functions that are applied very generically across applications involving images, natural language, PDEs, and many other types of data. The algorithms seem to perform well across all these settings—they appear to adapt to the task at hand. Given this adaptive nature, one might hope that the NNM would be capable of attaining the same exponential convergence rates as adaptive FEM techniques like hp-FEM. Indeed, a variety of theoretical results have been produced demonstrating that deep neural networks can, in theory, approximate a dizzying array of important function classes all with exponential convergence rates. Unfortunately, the studies

in Chapter 4 and App. C demonstrate that the NNM is far from attaining these theoretical bounds in practice.

A.3 The neural network method

A.3.1 Background

The neural network method (NNM) of solving differential equations was explored as early as the 1990s. Dissanayake and Phan-Thien [9] presented possibly the first demonstration of this technique for engineering applications. This was followed by a seemingly independent rediscovery of very similar algorithms by Milligen, Tribaldos, and Jiménez [30], and again by Lagaris, Likas, and Fotiadis [23]. In the following years, a series of refinements and applications for the NNM were produced; much of this work was covered in a textbook by Yadav, Yadav, and Kumar [42].

These early NNM research efforts were mostly limited to the use of shallow neural network architectures (see Sec. 1.3). In the early 2010s, the so-called deep learning revolution began to unfold: GPU-accelerated deep neural network algorithms became widely successful, breaking performance records on tasks in machine vision, natural language processing, and biochemistry. Free deep learning software frameworks like PyTorch and TensorFlow became readily accessible.

Around 2017, the success of deep learning began to stimulate renewed interest in the NNM. Inspired by the success of deep learning at overcoming the curse of dimensionality in machine learning applications, several studies demonstrated that deep variants of the NNM could directly solve high-dimensional PDEs [10, 11, 38, 16, 2]. Similarly, Sirignano and Spiliopoulos [38] demonstrated that the NNM could easily be extended to directly solve parameterized PDEs, effectively providing the functionality of FEM and model order reduction in a single

algorithm.

In the years since this rekindling of NNM research, a great diversity of methods have been developed and applied broadly to problems across many disciplines. Unfortunately, there is as of yet still no mature theoretical understanding of the NNM. In particular, it is not clear exactly why deep neural networks perform so much better than shallow neural networks, nor is it known when and how deep variants of the NNM are truly useful in practice. Much of the interest in the NNM is based on impressive empirical demonstrations like those listed above. However, the NNM research community lacks careful benchmarks, and as a result it is difficult to systematically compare the performance of the many variations of the NNM. Work is ongoing to improve on both of these fronts. This section will provide the reader with an overview of the NNM topics relevant to this thesis, including the basic NNM formulation, some relevant directions of innovation, and some of the major theoretical results available today.

A.3.2 Basic algorithms

At its core, the NNM is simply the idea of using a neural network as the trial function to solve a PDE. That is, the true solution $u(x)$ of the PDE will be approximated by a trial function

$$\tilde{u}(x) = N(x; \theta). \tag{A.31}$$

This can be contrasted directly with the form of the trial function used for FEM outlined in Eqn. A.3. Here N is any neural network function that takes x as an

input and outputs a value $\tilde{u}(x)$ that is of the same type (e.g., vector, scalar, etc.) as the true solution u . The parameter θ denotes the learnable parameters of N ; these are the counterparts to the coefficients c_i in the FEM trial function (Eqn. A.3). However, whereas the FEM trial function depends linearly on the parameters c_i , the neural network trial function N will typically depend nonlinearly on most of its parameters θ . The many variations of the NNM basically correspond to different choices of the trial function N in Eqn. A.31 and different algorithms for generating θ values corresponding to a given target function u .

This nonlinearity is ultimately the source of most of the advantages and disadvantages of the NNM relative to classical numerical techniques like FEM. On the one hand, it means that neural networks can potentially be much more expressive trial functions than linear combinations of simple basis functions. Indeed, the ansatz of depth, which necessarily implies nonlinearity, appears to be an essential requirement for the manner in which the NNM overcomes the curse of dimensionality. On the other hand, finding neural network parameters θ that correspond to a good approximation of a target function u is far more difficult than, e.g., finding good coordinates c_i to approximate u with a basis of compactly supported piecewise polynomial functions, as done in FEM. Not only is it more difficult to find such θ values in practice, but the common algorithms used to this end are far more difficult to analyse theoretically.

A.3.2.1 Neural network architectures

Equation A.31 specifies that the trial function N should be a neural network, but what does this mean? Many different classes of parameterized functions have been called neural networks: fully-connected neural networks, convolutional neural networks, recurrent neural networks, transformers networks, etc. These classes seem to vary greatly in their mathematical and computational properties, and are only loosely connected by themes such as depth and occasionally some heuristic motivations from neuroscience. A given family of neural network functions is typically referred to as a type of neural network architecture. In this section, we will review the basics of neural network architecture.

The earliest versions of the NNM used neural networks of the form

$$N_{\text{shallow}}(x; \theta) = c \cdot \sigma(Ax + b) + d. \quad (\text{A.32})$$

If $x \in \mathbb{R}^n$, then A is a matrix in $\mathbb{R}^{m \times n}$ for some choice of m , and b is a vector in \mathbb{R}^m . The function σ is some choice of nonlinear function called the activation function. It is usually a scalar function that is applied elementwise to the vector $Ax + b$, so that $\sigma(Ax + b)$ is another vector in \mathbb{R}^m . Finally, c is another vector in \mathbb{R}^m and d is a scalar, so that $N_{\text{shallow}}(x; \theta)$ is a scalar function of x . This form of trial function is only appropriate for approximating scalar functions, but could easily be modified to approximate a vector function⁵. The elements used in multiplications (here A and c) are generally called weights, whereas those used in addition (here b and d) are called biases.

⁵To approximate a vector function of x , one could for instance replace c with a matrix of the appropriate shape and d with a corresponding vector.

Neural network architectures of the form in Eqn. A.32 will be referred to in this thesis as shallow fully-connected neural networks⁶. What is most salient about this function class is that they are shallow: only one nonlinear operation is present between input and output. Specifically, we will say that this architecture has one hidden layer of width m . The vector $Ax + b$ is the preactivation vector of the hidden layer, and the vector $\sigma(Ax + b)$ is the activation vector of the hidden layer. The output $N_{\text{shallow}}(x; \theta)$ is sometimes called the output layer, but this name is more natural when $N_{\text{shallow}}(x; \theta)$ is not a scalar. Similarly, the input x is sometimes called the input layer. In summary, the input layer is mapped linearly to the hidden layer; an elementwise nonlinearity is applied; and finally the hidden layer is mapped linearly to the output layer.

What are the learnable parameters θ of N_{shallow} in Eqn. A.32? The answer to this question depends on the algorithm used to fit N_{shallow} to target functions. In practice, it is most common for practitioners to fix m and σ *a priori*. These are called hyperparameters of the architecture⁷. All the remaining degrees of freedom are learnable parameters: that is, θ includes the weights (elements of A and c) and the biases (b and d).

The NNM implementations in this thesis use trial functions from the more general function class of deep fully-connected neural networks. These can be

⁶They may be referred to elsewhere as artificial neural networks, multilayer perceptrons, or by other names.

⁷Algorithms do exist for learning hyperparameters. These are variously referred to as automatic machine learning, neural architecture search, adaptive activation functions.

defined recursively as:

$$\begin{aligned}
 f_1 &= \sigma_1 (A_1 x + b_1), \\
 f_2 &= \sigma_2 (A_2 f_1 + b_2), \\
 f_3 &= \sigma_3 (A_3 f_2 + b_3), \\
 &\dots \\
 f_L &= \sigma_L (A_L f_{L-1} + b_L), \\
 N_{\text{deep}}(x; \theta) &= c \cdot f_L + d.
 \end{aligned} \tag{A.33}$$

This form is precisely the composition of L shallow neural networks. The input x is linearly transformed to the first hidden layer's preactivation vector $A_1 x + b_1$, which is in \mathbb{R}^{m_1} for some width m_1 . The first activation function is applied elementwise to produce the first postactivation vector. The i th hidden layer is obtained from the $(i - 1)$ th layer in a similar fashion, with each hidden layer having its own width m_i , weights A_i , biases b_i , and activation function σ_i . The deep network in Eqn. A.33 has a total of L hidden layers. Finally, the output layer maps the activation vector of the last hidden layer f_L to the output layer $N_{\text{deep}}(x; \theta)$ via a linear transformation.

The deep fully-connected neural network is one of the simplest classes of deep neural networks. In practice, it is most common to further simplify the architecture by assuming that all hidden layers are of equal width⁸ (i.e., $m_i = m$ for all i) and use the same activation function (i.e., $\sigma_i = \sigma$ for all i). As in the shallow case, m and σ are usually considered hyperparameters, so that the

⁸The results in Chapter 5, however, suggest that this may not always be the optimal choice. Rather, that analysis suggests that increasing the m_i with i may be advantageous.

learnable parameters θ are the elements of all the A_i and b_i as well as c and d .

The choice of activation function σ has important implications for the performance of neural network architectures. One of the most popular choices is the rectified linear unit (ReLU) function, $\sigma(y) = \max(0, y)$. Unfortunately, ReLU is not compatible with many NNM training algorithms⁹. As such, smooth activation functions are more common. The hyperbolic tangent function $\sigma(y) = \tanh(y)$ is very common, and will be used for all the NNM implementations in this thesis.

Modern deep neural networks are usually far more sophisticated than the humble fully-connected architectures. For instance, residual networks add identity mappings to the hidden layers, so that they are of the form

$$f_j = f_{j-1} + \sigma(A_j f_{j-1} + b_j). \quad (\text{A.34})$$

The identity term can dramatically improve the performance of the networks in practice, at least in part because they work synergistically with gradient-based optimization techniques for finding θ [17]. Residual networks are very similar to highway networks, which have layers of the form

$$T_j = g(E_j f_{j-1} + h_j), \quad (\text{A.35})$$

$$f_j = (1 - T_j)f_{j-1} + T_j \sigma(A_j f_{j-1} + b_j). \quad (\text{A.36})$$

⁹Specifically, the output of a ReLU-based networks is a piecewise linear function of its input. Loss functionals that are based on the second- or higher-order derivatives of the output with respect to the input will not function properly. In particular, differential operators of order two or higher will be zero almost everywhere for such networks, so loss functionals based on the strong form of the PDE for second- or higher-order PDEs will not be properly defined for such networks. Loss functionals based on weaker formulations of the PDE may still work.

Here E_j is a matrix and h_j is a vector (both of the appropriate sizes) and g is another activation function. The g function must have an output range in $[0, 1]$, so that T_j acts as a gating mechanism between the two terms in the layer. Setting $T_j = 0.5$ recovers an architecture similar to residual networks. Residual networks and related architectures are extremely popular architectures in deep learning today; furthermore, both residual networks and highway networks have inspired successful NNM algorithms (e.g., Refs. [11, 38]).

A.3.2.2 Training neural networks

Having selected the neural network architecture in the trial function N (Eqn. A.31), the next step in the NNM is to determine parameters θ for which $N(x; \theta)$ is a good approximation to the solution u of the PDE being solved. Recall that, when solving linear PDEs via FEM, determining coefficients c_i that provide a good approximation of u is as simple as inverting a system of linear equations. Alas, because the NNM trial function generally depends nonlinearly on θ , no such simple algorithm exists. Instead, the selection of θ is accomplished using techniques from the broader deep learning community.

This process is typically called training the neural network, a name reflecting the origins of these techniques in artificial intelligence research. The standard recipe for neural network training is conducted in two phases. First, an initial guess for θ is generated, usually by randomly drawing from some simple distribution. Second, an iterative gradient-based optimization scheme is used to update the current guess of θ by minimizing some functional, called the loss.

The most common initialization distributions for θ are Gaussian

distributions¹⁰. This introduces additional hyperparameters, namely the choice of the mean and variance for the Gaussian distribution from which each parameter is drawn. All NNM implementations in this thesis use the simple initialization of Glorot and Bengio [13].

The optimization of θ from its initial value is typically conducted using algorithms based on gradient descent. In basic gradient descent, θ is updated iteratively according to

$$\theta_i = \theta_{i-1} - \eta \nabla_{\theta} \mathcal{L}[N(x; \theta_{i-1})]. \quad (\text{A.37})$$

Here, $\mathcal{L}[N(x; \theta)]$ is a functional that maps N to a scalar value called the loss. The gradients ∇_{θ} with respect to θ can be computed efficiently on GPU hardware using the backpropagation algorithm [25]. The constant η here is called the learning rate; this is yet another hyperparameter. For sufficiently small values of η , gradient descent tends to produce an updated value θ_i that produces a smaller loss value than did θ_{i-1} .

In practice, the pure gradient descent algorithm is rarely used. The loss functional typically cannot be computed exactly as most loss functionals involve complicated integrals over the domain of x . These integrals can be estimated by Monte Carlo integration, leading to what is known as stochastic gradient descent. The number of samples used for these Monte Carlo integration estimators is called the batch size of the algorithm. Besides this, various heuristic

¹⁰Note that initializing all weights and biases to identical values leads to pathological training. Standard gradient-based optimization schemes are unable to distinguish between the effects of different parameters in a given layer, so that the parameters are updated identically in every iteration and remain equal in value forever.

modifications of gradient descent are commonly used to improve its convergence properties. Another important class of optimizers sometimes used with the NNM are quasi-Newton methods, such as the BFGS method; these are beyond the scope of the current thesis. The NNM implementations in this thesis are all based on the Adam optimizer, which is a variant of stochastic gradient descent that incorporates adaptive heuristics to dynamically adjust the effective learning rate during training [21].

A.3.2.3 Loss functionals

The loss functional for the NNM must be constructed such that small values of the loss functional correspond to θ values for which N is a good approximation of the unknown solution u of the PDE. The most common choice of NNM loss functional is of the form

$$\mathcal{L}[N(x; \theta)] = \int_{\Omega} (LN(x; \theta) - f(x))^2 dx + \beta \int_{\partial\Omega} (BN(x; \theta) - g(x))^2 ds. \quad (\text{A.38})$$

Here L is the differential operator with source f for a target PDE of the form $Lu = f$ on a domain Ω , and B is an operator with source g describing the boundary conditions $Bu = g$ on the boundary $\partial\Omega$. The first loss term quantifies how badly the trial function fails to satisfy the PDE, and the second loss term quantifies how badly the trial function fails to satisfy the boundary conditions. The hyperparameter β sets the relative importance of satisfying the PDE versus the boundary conditions. In this thesis, β is simply fixed to a constant value near 1 that was determined by trial and error to produce good results.

When optimizing θ using stochastic gradient descent and similar methods, each term in Eqn. A.38 can be approximated by a separate Monte Carlo approximation over its respective domain. In this way, the NNM is mesh-free: all that it requires is samples drawn uniformly from the domain and its boundary, and generating such samples does not require a mesh. Indeed, it is thanks to this mesh-freedom that the NNM can operate effectively in high-dimensional domains.

A.3.2.4 Regarding generalization

For most applications of deep learning, the concept of generalization is of fundamental importance to the training process. In standard deep learning applications, the deep neural network is trained to fit partial measurements from some target function. The measurements are typically noisy, and the data available for training may not be a perfect representation of the data that will be encountered when the model is deployed for its intended application. Thus, various methods exist for gauging how well the network will perform on previously unseen data. In particular, standard training algorithms run the risk of overfitting to the training data, producing neural networks that have essentially memorized meaningless patterns.

In the case of the NNM, generalization and overfitting are usually not important concerns. Usually, the NNM is applied to PDE models in which the problem data is assumed to be known exactly. The training data corresponds simply to samples of x taken from the problem domain, which can be generated at essentially no cost. Moreover, PDE problems do not usually involve testing the approximated solution on points outside the domain on which the problem was

originally posed. Although there are applications of the NNM in which these statements are false and for which generalization is an important consideration, this is not the case for the work in this thesis.

A.3.2.5 Model order reduction with the NNM

One of the most exciting features of the NNM is the ease with which it can be extended to act as a model order reduction technique. As in Sec. A.2.1.4, consider a parameterized PDE of the form

$$L(p)u(x;p) = f(x;p), \quad x \in \Omega(p). \quad (\text{A.39})$$

As first popularized by Sirignano and Spiliopoulos [38], the NNM can be implemented directly with a trial function of the form

$$\tilde{u}(x,p) = N(x,p;\theta) \quad (\text{A.40})$$

to produce an approximation that is a function of both the PDE domain coordinates x and the problem parameters p . Training can be conducted essentially as usual, for instance by generalizing the loss functional in Eqn. A.38 to

$$\mathcal{L}[N(x,p;\theta)] = \int_{\mathcal{P}} \left[\int_{\Omega} (LN(x;\theta) - f(x))^2 dx + \beta \int_{\partial\Omega} (BN(x;\theta) - g(x))^2 ds \right] dp, \quad (\text{A.41})$$

where $p \in \mathcal{P}$ defines some set of problem parameters for which the parameterized PDE is to be solved. This capability is of great interest for applications in

computational biophysics in general and the design of periodic MNFDs in particular; App. D contains a study exploring this very use case, and is discussed in Chapter 6.

A.3.3 Rates of convergence

The questions and concerns regarding error for the NNM are essentially the same listed for FEM at the beginning of Sec. A.2.2. However, the trial function in the NNM is a nonlinear function of its parameter θ , and the training of a deep neural network is a nonlinear nonconvex optimization problem. Each of these points makes the NNM quite challenging to analyse theoretically.

Because of these challenges, the NNM still lacks a comprehensive theoretical foundation comparable to that available for FEM and other well-established numerical methods. The practical development of NNM algorithms is still highly dependent on empirical demonstrations. However, often proof-of-concept demonstrations are conducted on toy models that are not representative of the challenging features arising in real PDE applications. Further, few studies carefully and systematically analyse the relationship between NNM error, runtime, and hyperparameter selection.

Despite the difficulty in theoretical analysing the NNM, there are some important theoretical results available today, and the theoretical NNM literature has been growing steadily in recent years. Relevant literature can be broadly categorized into two main topics: expressivity and trainability. Expressivity results attempt to understand how well deep neural networks can approximate different functions under ideal conditions. This is analogous to the concepts of

best achievable error and interpolation error discussed for FEM in Sec. A.2.2. As reviewed below, several studies have suggested that deep neural networks have much better expressivity than FEM approximations, at least in theory.

On the other hand, trainability results assess whether practical NNM algorithms actually achieve approximation performance comparable to the best rates predicted by expressivity analyses. Recall that powerful results like Céa's lemma ensure that the performance of practical FEM implementations is fairly close to the best possible FEM performance achievable in theory. In contrast, empirical studies have consistently identified that the convergence of the NNM in practice is far worse than expressivity results predict in theory. This paradox has come to be known as the theory-to-practice gap.

It is important to recall that, although the theory-to-practice gap draws into question the practical relevance of many available expressivity results, empirical studies have successfully demonstrated the use of the NNM to solve some challenging problems. In particular, empirical demonstrations of the NNM solving high-dimensional PDEs and parameterized PDEs were made without any particular theoretical motivation. Thus, the theory-to-practice gap is a shortcoming of current NNM theory, not of the NNM itself.

Ultimately, what remains unclear is under precisely what circumstances the NNM is able to produce approximations of acceptable accuracy in a manner that is faster, easier, or otherwise more useful than the best available alternatives. This thesis includes studies intended to help clarify the answer to this question. The behaviour of NNM errors is investigated from various directions in Chapter 4 and Apps. C, B, and D. Moreover, Chapter 5 presents a study of the internal

representations learned in the hidden layers of deep neural networks trained via the NNM,.

A.3.3.1 Expressivity

The discussion of deep neural network expressivity begins most naturally with the so-called universal approximation theorems. The first of these was proven in Cybenko [7], followed shortly thereafter by a more general version in Hornik [18]. These proofs demonstrated that shallow neural networks could uniformly approximate any continuous function on a compact domain, given sufficient capacity. Many extensions of these theories have been published over the years, extending the results to different activation functions and architectures.

There are some major problems with these theories. They provide no indication as to how one should obtain the weights that are implied to exist. They do not clearly demonstrate that the neural networks are in any way better than alternative classes of universal approximants, such as polynomials or piecewise polynomials. They do not indicate how big the neural networks should be to achieve a given level of accuracy. Moreover, they do not imply any need or advantage to deep neural networks; in fact, rather than explaining the modern success of deep learning, the universal approximation theorems historically discouraged research in this direction for many years. Nonetheless, this class of results is still used today to justify and motivate much work with neural networks, when the implications of these theorems are far less relevant than they may first appear.

Results from the field of approximation theory overcome some of the

limitations of universal approximation theorems. As discussed in Sec. A.2.2, a very powerful theoretical result in this area is due to DeVore, Howard, and Micchelli [8], called the method of continuous nonlinear widths. The theorem applies very broadly to any approximation system in which the coefficients of the approximation depend continuously on the function being approximated. For such methods, DeVore concluded very broadly that

$$\epsilon \sim \mathcal{O}(N^{\frac{r}{d}}), \quad (\text{A.42})$$

where ϵ is the worst-case approximation error, N is the number of degrees of freedom, r is the regularity of the target function, and d is the dimensionality of the problem domain.

It is not clear that the approximation of a function by deep neural networks depends continuously on the target function. In fact, the standard training protocol (Sec. A.3.2.2) involves the use of a random initial guess for the degrees of freedom followed by a stochastic optimization algorithm. Indeed, the rate set by Eqn. A.42 suffers from the curse of dimensionality unless $r \rightarrow \infty$. This would contradict the successful demonstrations of the NNM on high-dimensional PDEs. Conversely, the empirical results in Chapter 5 actually suggest that, asymptotically, the NNM may generate hidden layer representations that appear to depend continuously on the target function. Future work should attempt to clarify whether deep learning does produce functions that depend continuously (in some sense) on the target function, and whether this has implications related to the theoretical rate in Eqn. A.42.

Various authors have proposed proofs that claim to directly confirm that deep

neural networks can represent high-dimensional functions without an exponential growth in computational cost [15, 20, 19]. Many of these proofs proceed by manually constructing deep neural networks that efficiently approximate some well-understood function class for which exponentially good approximation theorems are already known. This does not reflect the reality of how deep neural networks are trained in practice. As such, the predicted convergence rates are not observed empirically, as discussed below.

A.3.3.2 Trainability

Unfortunately, the exponentially fast convergence rates predicted by theoretical analyses of deep neural network expressivity have not been observed empirically in real NNM algorithms. As mentioned earlier in this section, this is known as the theory-to-practice gap. Recent work by Grohs and Voigtlaender [14] has actually provided some robust theoretical proof that this gap must exist, at least in a certain setting. A proper discussion of that work is too technical for the current context, but some tangible intuition towards the inevitability of this gap can be obtained by reviewing the convergence of gradient-based optimization schemes.

As reviewed in Sec. A.3.2.2, deep learning training is generally conducted using some variant of gradient descent. Theoretical results for the convergence of gradient descent are difficult to obtain for general non-convex and nonlinear optimization problems like deep neural network training. However, very clear results are available in the case of convex optimization [39].

Specifically, in the convex case, the error ϵ of gradient descent is known to converge as

$$\epsilon \sim \mathcal{O}\left(\frac{1}{k}\right), \quad (\text{A.43})$$

where k is the number of iterations. In fact, it is known that the best possible rate that can be achieved by any first-order gradient based optimization scheme is, in the convex scenario,

$$\epsilon \sim \mathcal{O}\left(\frac{1}{k^2}\right). \quad (\text{A.44})$$

Nesterov [33] proposed an accelerated gradient descent technique that achieves this rate.

Gradient descent and related methods can converge faster than these worst-case rates suggest for particularly well-conditioned optimization objectives (e.g., strongly convex functions). On the other hand, convergence can potentially become much worse when stochastic variants of gradient descent are utilized. The basic stochastic gradient descent is expected to converge as

$$\epsilon \sim \mathcal{O}\left(\frac{1}{k^{1/2}}\right), \quad (\text{A.45})$$

under typical conditions [39].

Of course, these results all apply for convex optimization, and deep neural network training is not a convex optimization problem. In fact, it is known empirically to be a difficult optimization problem. Thus, it seems reasonable to

expect, heuristically, that convergence in deep learning is unlikely to be faster than the rates listed above. This is, in practice, a problematic bottleneck to achieving very low levels of error with the NNM. Only modest levels of error may be attained in a reasonable amount of time.

It is worth noting that the slow convergence and/or moderate levels of achievable error of the NNM do not necessarily compromise its utility in many cases.

- For molecular dynamics described by high-dimensional Smoluchowski PDEs, there are few alternative solution methods. Particle simulations, the most common approach to study these systems, are also slow and limited to modest levels of accuracy in practice.
- For high-dimensional PDE models of force fields, it may be sufficient to precompute the fields once before using the fields repeatedly in subsequent molecular dynamics simulations. The up-front cost of precomputing a large NNM solution may be competitive with the overall computational cost of computing the forces repeatedly during simulations.
- For PDEs in complicated geometries, the mesh-free nature of the NNM enables it to operate without any human supervision. Conversely, the real-world cost of FEM is often bottlenecked by the human labour required to properly mesh the domain. So long as the NNM can be trusted to reliably converge eventually, it may be cost-effective in some cases to replace large amounts of human labour with very large amounts of computing time. What is more, the computational cost of FEM and other

mesh-based methods grows rapidly with the intricacy of the problem geometry. For this application, the greatest barrier to the use of the NNM is a lack of guarantees that it will converge reliably to even a modest level of accuracy. See Chapter 4 and Apps. B and D for related discussions.

- For highly parameterized problems, existing model order reduction methods have various limitations that have hindered their widespread use. In particular, handling parameterized geometries with such methods is difficult. App. D directly illustrates the potential advantage of using the NNM to study such systems. Again, modest levels of error are often acceptable for such applications. Conversely, the computational cost is not necessarily prohibitive since the NNM is providing a novel functionality that cannot be reproduced with existing tools.
- For nonlinear PDEs, even the FEM requires the solution of a nonlinear system of equations. Nonlinear FEM is also difficult and potentially computationally expensive. The relative disadvantage of the NNM may diminish for hard nonlinear problems.

Appendix B

Error bounds using Green functions

Proofs of error bounds

Martin Magill,¹ Andrew Nagel,¹ and Hendrick W. de Haan^{1,*}

¹*Faculty of Science, University of Ontario Institute of Technology,
2000 Simcoe St N, Oshawa, Ontario, Canada L1H7K4*

(Dated: April 28, 2022)

This brief technical note contains a proof that the error of an NNM solution to a certain class of PDEs is bounded above by the same loss functional used during training, up to a problem-dependent constant factor. The idea of the proof is to express the true and approximate solutions in terms of a common Green function, and then select the Green function in a way that allows the loss functional to emerge as an upper bound. The method appears to be directly extensible to a broader class of second-order linear PDEs by using the integral form of Lagrange's identity in place of Green's third identity, including at least some time-integrated Smoluchowski equations.

In this discussion, we use \mathbf{x} and \mathbf{x}' to denote points in the PDE domain Ω ; we use $\langle a, b \rangle_\Omega$ to denote the integral of the product ab over Ω ; and we use $\langle a, b \rangle_{\partial\Omega}$ to denote the integral of ab over $\partial\Omega$. Suppose the PDE being solved is of the form

$$\nabla_{\mathbf{x}}^2 u(\mathbf{x}) = f(\mathbf{x}) \quad \mathbf{x} \in \Omega, \quad (1)$$

$$u(\mathbf{x}) = g(\mathbf{x}) \quad \mathbf{x} \in \partial\Omega_D, \quad (2)$$

$$u_{\hat{n}}(\mathbf{x}) = h(\mathbf{x}) \quad \mathbf{x} \in \partial\Omega_N, \quad (3)$$

Here $\nabla_{\mathbf{x}}^2$ is the Laplacian operator acting over \mathbf{x} , f is a source term, g specifies Dirichlet boundary conditions on the subset of the boundary $\partial\Omega_D$, and h specifies Neumann boundary conditions on the subset of the boundary $\partial\Omega_N$.

If $G(\mathbf{x}, \mathbf{x}')$ is any Green function satisfying $\nabla_{\mathbf{x}}^2 G(\mathbf{x}, \mathbf{x}') = \delta(\mathbf{x} - \mathbf{x}')$, then, using Green's third identity, any arbitrary trial function $\tilde{u}(\mathbf{x})$ can be written as

$$\begin{aligned} \tilde{u}(\mathbf{x}') &= \langle G(\mathbf{x}, \mathbf{x}'), \nabla^2 \tilde{u}(\mathbf{x}) \rangle_\Omega \\ &+ \langle \tilde{u}(\mathbf{x}), G_{\hat{n}}(\mathbf{x}, \mathbf{x}') \rangle_{\partial\Omega} \\ &- \langle G(\mathbf{x}, \mathbf{x}'), \tilde{u}_{\hat{n}}(\mathbf{x}) \rangle_{\partial\Omega} \end{aligned} \quad (4)$$

The above is true for any Green function. Now choose specifically the Green function that satisfies:

$$\nabla_{\mathbf{x}}^2 G(\mathbf{x}, \mathbf{x}') = \delta(\mathbf{x} - \mathbf{x}') \quad \mathbf{x} \in \Omega, \quad (5)$$

$$G(\mathbf{x}, \mathbf{x}') = 0 \quad \mathbf{x} \in \partial\Omega_D, \quad (6)$$

$$G_{\hat{n}}(\mathbf{x}, \mathbf{x}') = 0 \quad \mathbf{x} \in \partial\Omega_N. \quad (7)$$

With this choice, our representation of \tilde{u} reduces to

$$\begin{aligned} \tilde{u}(\mathbf{x}') &= \langle G(\mathbf{x}, \mathbf{x}'), \nabla^2 \tilde{u}(\mathbf{x}) \rangle_\Omega \\ &+ \langle \tilde{u}(\mathbf{x}), G_{\hat{n}}(\mathbf{x}, \mathbf{x}') \rangle_{\partial\Omega_D} \\ &- \langle G(\mathbf{x}, \mathbf{x}'), \tilde{u}_{\hat{n}}(\mathbf{x}) \rangle_{\partial\Omega_N} \end{aligned} \quad (8)$$

The same decomposition can be written for the true solution, $u(\mathbf{x})$. Since the integration operations are linear, this yields the following expression for the difference between the true and approximate solutions at any point \mathbf{x}' :

$$\begin{aligned} u(\mathbf{x}') - \tilde{u}(\mathbf{x}') &= \langle G(\mathbf{x}, \mathbf{x}'), \nabla^2 u(\mathbf{x}) - \nabla^2 \tilde{u}(\mathbf{x}) \rangle_\Omega \\ &+ \langle u(\mathbf{x}) - \tilde{u}(\mathbf{x}), G_{\hat{n}}(\mathbf{x}, \mathbf{x}') \rangle_{\partial\Omega_D} \\ &- \langle G(\mathbf{x}, \mathbf{x}'), u_{\hat{n}}(\mathbf{x}) - \tilde{u}_{\hat{n}}(\mathbf{x}) \rangle_{\partial\Omega_N} \end{aligned} \quad (9)$$

or, applying the boundary conditions of the PDE,

$$\begin{aligned} u(\mathbf{x}') - \tilde{u}(\mathbf{x}') &= \langle G(\mathbf{x}, \mathbf{x}'), f(\mathbf{x}) - \nabla^2 \tilde{u}(\mathbf{x}) \rangle_\Omega \\ &+ \langle g(\mathbf{x}) - \tilde{u}(\mathbf{x}), G_{\hat{n}}(\mathbf{x}, \mathbf{x}') \rangle_{\partial\Omega_D} \\ &- \langle G(\mathbf{x}, \mathbf{x}'), h(\mathbf{x}) - \tilde{u}_{\hat{n}}(\mathbf{x}) \rangle_{\partial\Omega_N} \end{aligned} \quad (10)$$

Thus, the squared error in the approximate solution at point \mathbf{x}' is given by

$$\begin{aligned} |u(\mathbf{x}') - \tilde{u}(\mathbf{x}')|^2 &= |\langle G(\mathbf{x}, \mathbf{x}'), f(\mathbf{x}) - \nabla^2 \tilde{u}(\mathbf{x}) \rangle_\Omega \\ &+ \langle g(\mathbf{x}) - \tilde{u}(\mathbf{x}), G_{\hat{n}}(\mathbf{x}, \mathbf{x}') \rangle_{\partial\Omega_D} \\ &- \langle G(\mathbf{x}, \mathbf{x}'), h(\mathbf{x}) - \tilde{u}_{\hat{n}}(\mathbf{x}) \rangle_{\partial\Omega_N}|^2 \end{aligned} \quad (11)$$

By the triangle inequality, this can be reduced to

$$\begin{aligned} |u(\mathbf{x}') - \tilde{u}(\mathbf{x}')|^2 &\leq |\langle G(\mathbf{x}, \mathbf{x}'), f(\mathbf{x}) - \nabla^2 \tilde{u}(\mathbf{x}) \rangle_\Omega|^2 \\ &+ |\langle g(\mathbf{x}) - \tilde{u}(\mathbf{x}), G_{\hat{n}}(\mathbf{x}, \mathbf{x}') \rangle_{\partial\Omega_D}|^2 \\ &+ |\langle G(\mathbf{x}, \mathbf{x}'), h(\mathbf{x}) - \tilde{u}_{\hat{n}}(\mathbf{x}) \rangle_{\partial\Omega_N}|^2 \end{aligned} \quad (12)$$

Next, via the Cauchy-Schwarz inequality, this implies that

$$\begin{aligned} |u(\mathbf{x}') - \tilde{u}(\mathbf{x}')|^2 &\leq \|G(\mathbf{x}, \mathbf{x}')\|_\Omega^2 \|f(\mathbf{x}) - \nabla^2 \tilde{u}(\mathbf{x})\|_\Omega^2 \\ &+ \|G_{\hat{n}}(\mathbf{x}, \mathbf{x}')\|_{\partial\Omega_D}^2 \|g(\mathbf{x}) - \tilde{u}(\mathbf{x})\|_{\partial\Omega_D}^2 \\ &+ \|G(\mathbf{x}, \mathbf{x}')\|_{\partial\Omega_N}^2 \|h(\mathbf{x}) - \tilde{u}_{\hat{n}}(\mathbf{x})\|_{\partial\Omega_N}^2, \end{aligned} \quad (13)$$

where $\|\cdot\|$ denotes the L^2 norm over the respective domains. Define the problem-specific constants

$$A(\mathbf{x}') = \|G(\mathbf{x}, \mathbf{x}')\|_\Omega^2, \quad (14)$$

$$B(\mathbf{x}') = \|G_{\hat{n}}(\mathbf{x}, \mathbf{x}')\|_{\partial\Omega_D}^2, \quad (15)$$

$$C(\mathbf{x}') = \|G(\mathbf{x}, \mathbf{x}')\|_{\partial\Omega_N}^2, \quad (16)$$

$$K(\mathbf{x}') = \max(A(\mathbf{x}'), B(\mathbf{x}'), C(\mathbf{x}')). \quad (17)$$

Thus, the absolute error at \mathbf{x}' can be bounded above as

* Hendrick.deHaan@uoit.ca

follows:

$$|u(\mathbf{x}') - \tilde{u}(\mathbf{x}')|^2 \leq A(\mathbf{x}') \|f(\mathbf{x}) - \nabla^2 \tilde{u}(\mathbf{x})\|_\Omega^2 + B(\mathbf{x}') \|g(\mathbf{x}) - \tilde{u}(\mathbf{x})\|_{\partial\Omega_D}^2 \quad (18)$$

$$+ C(\mathbf{x}') \|h(\mathbf{x}) - \tilde{u}_{\hat{n}}(\mathbf{x})\|_{\partial\Omega_N}^2 \leq K(\mathbf{x}') (\|f(\mathbf{x}) - \nabla^2 \tilde{u}(\mathbf{x})\|_\Omega^2 + \|g(\mathbf{x}) - \tilde{u}(\mathbf{x})\|_{\partial\Omega_D}^2 + \|h(\mathbf{x}) - \tilde{u}_{\hat{n}}(\mathbf{x})\|_{\partial\Omega_N}^2) \quad (19)$$

Recall that the loss functional used for training of the NNM on this PDE would be

$$\begin{aligned} \mathcal{L}[\tilde{u}] &= \int_\Omega (\nabla^2 \tilde{u}(\mathbf{x}) - f(\mathbf{x}))^2 d\mathbf{x} \\ &+ \int_{\partial\Omega_D} (\tilde{u}(\mathbf{x}) - g(\mathbf{x}))^2 ds \\ &+ \int_{\partial\Omega_N} (\tilde{u}_{\hat{n}}(\mathbf{x}) - h(\mathbf{x}))^2 ds. \end{aligned} \quad (20)$$

However, this is precisely equal to

$$\begin{aligned} \mathcal{L}[\tilde{u}] &= \|f(\mathbf{x}) - \nabla^2 \tilde{u}(\mathbf{x})\|_\Omega^2 \\ &+ \|g(\mathbf{x}) - \tilde{u}(\mathbf{x})\|_{\partial\Omega_D}^2 \\ &+ \|h(\mathbf{x}) - \tilde{u}_{\hat{n}}(\mathbf{x})\|_{\partial\Omega_N}^2, \end{aligned} \quad (21)$$

which means that, at each point \mathbf{x}' , the squared error is bounded above according to

$$|u(\mathbf{x}') - \tilde{u}(\mathbf{x}')|^2 \leq K(\mathbf{x}') \mathcal{L}[\tilde{u}]. \quad (22)$$

Furthermore, this implies that the MSE is bounded above as

$$\text{mean}_{\mathbf{x}'} (|u(\mathbf{x}') - \tilde{u}(\mathbf{x}')|^2) \leq \bar{K} \mathcal{L}[\tilde{u}], \quad (23)$$

where

$$\bar{K} = \text{mean}_{\mathbf{x}'} (K(\mathbf{x}')) \quad (24)$$

is a problem-specific constant (i.e., it does not depend on \tilde{u}).

The derivation above can be extended to provide bounds for the mean and max absolute errors of the potential, as well. Taking the square root of both sides of Eqn. 22 yields

$$|u(\mathbf{x}') - \tilde{u}(\mathbf{x}')| \leq \sqrt{K(\mathbf{x}')} \sqrt{\mathcal{L}[\tilde{u}]}, \quad (25)$$

from which is it clear that

$$\text{mean}_{\mathbf{x}'} (|u(\mathbf{x}') - \tilde{u}(\mathbf{x}')|) \leq \text{mean}_{\mathbf{x}'} (\sqrt{K(\mathbf{x}')} \sqrt{\mathcal{L}[\tilde{u}]}) \quad (26)$$

and

$$\max_{\mathbf{x}'} (|u(\mathbf{x}') - \tilde{u}(\mathbf{x}')|) \leq \max_{\mathbf{x}'} (\sqrt{K(\mathbf{x}')} \sqrt{\mathcal{L}[\tilde{u}]}) \quad (27)$$

Thus, the mean and max absolute errors of the potential scale in proportion to the square root of the loss functional.

Similarly, in addition to the above errors in the potential, the derivation can be extended to apply to the errors in the electric field. Taking the gradient of Eqn. 10 with respect to \mathbf{x}' yields

$$\begin{aligned} \nabla_{\mathbf{x}'} u(\mathbf{x}') - \nabla_{\mathbf{x}'} \tilde{u}(\mathbf{x}') &= \langle \nabla_{\mathbf{x}'} G(\mathbf{x}, \mathbf{x}'), f(\mathbf{x}) - \nabla^2 \tilde{u}(\mathbf{x}) \rangle_\Omega \\ &+ \langle g(\mathbf{x}) - \tilde{u}(\mathbf{x}), \nabla_{\mathbf{x}'} G_{\hat{n}}(\mathbf{x}, \mathbf{x}') \rangle_{\partial\Omega_D} \\ &- \langle \nabla_{\mathbf{x}'} G(\mathbf{x}, \mathbf{x}'), h(\mathbf{x}) - \tilde{u}_{\hat{n}}(\mathbf{x}) \rangle_{\partial\Omega_N}. \end{aligned} \quad (28)$$

This is a vector equation: the left and right hand sides describe vector fields. However, the procedure used above to derive Eqn. 22 from Eqn. 10 can be applied identically to each component of Eqn. 28. Thus, it follows that the MSE, MAE, and max AE of each component of the electric field are bounded by the loss functional in the same fashion as the corresponding electric potential errors. These component-wise bounds can then be combined into bounds for the Euclidean errors on the electric field.

Appendix C

Compact neural network solutions
to Laplace's equation in a
nanofluidic device

Compact Neural Network Solutions to Laplace’s Equation in a Nanofluidic Device

Martin Magill

U. of Ontario Inst. of Tech.
martin.magill1@uoit.net

Faisal Z. Qureshi

U. of Ontario Inst. of Tech.
faisal.qureshi@uoit.ca

Hendrick W. de Haan

U. of Ontario Inst. of Tech.
hendrick.dehaan@uoit.ca

Abstract

We explore the use of neural networks to solve the Laplace equation in a two-dimensional geometry. Specifically, we study a PDE problem that models the electric potential inside the slit-well nanofluidic device. Such devices are typically used to separate polymer mixtures by molecular size. Processes like these are commonly studied using GPU-accelerated coarse-grained particle simulations, for which GPU memory is a bottleneck. We compare the memory required to represent the field using neural networks to that needed to store solutions obtained using the finite element method. We find that even simple fully-connected neural networks can achieve accuracy to memory consumption ratios comparable to the good finite element solutions. These preliminary results demonstrate an industrial application that would benefit greatly from compact neural network representation techniques.

1 Introduction

Micro- and nanofluidic devices (MNFDs) consist of small geometries filled with electrolytic solution, such as water with dissolved NaCl [1–3]. One major application of these devices is the separation of polymer mixtures by size; specifically, the process of sorting DNA molecules by size is of widespread importance. MNFDs are being investigated as next-generation separation technologies for advantages such as miniaturization, automation, as well as improved speed and efficiency. Separation in MNFDs is often accomplished by introducing the polymers into the confined solution, then applying an electric field to drive them through the geometry. Over time, as a result, polymers become spatially segregated by size.

As the dynamics of polymers driven through confinement can be quite rich, MNFD design is research-intensive. Experimental investigations can be expensive, and have some intrinsic limitations (such as the optical resolution limit of light). Molecular dynamics (MD) simulations are often used in tandem with experiments, as they are cheaper and provide information that is inaccessible in experiment [4]. These simulations must balance physical realism against computational cost. Efficiency is particularly important because, as polymer dynamics in MNFDs are stochastic, simulations must be repeated many times to accurately measure statistical information. Simulations of MD in complete atomistic detail are quite computationally expensive. Coarse-grained (CG) models can be simulated far more efficiently, but must preserve sufficient detail to capture the essential phenomenology of the system under consideration. In coarse-grained Langevin dynamics (CGLD) models, a polymer is modelled as a chain of beads connected by springs, whereas in coarse-grained Brownian dynamics (CGBD) models an entire polymer is represented by a single effective particle. CGLD is often appropriate for nanofluidic devices, and CGBD for microfluidic devices.

These CG models can be simulated very efficiently, especially because they can readily exploit GPU acceleration. HOOMD-blue, a free open-source particle general purpose particle simulation toolkit, can accelerate simulations by over an order of magnitude using a single GPU, and achieves strong

scaling across up to thousands of GPUs [5, 6]. Furthermore, whereas published benchmarks of HOOMD-blue are conducted on simulations of millions of interacting particles, CG simulations for MNFDs often contain a few hundred particles or less. Since each simulation consumes so little memory, many independent instances can be simulated in parallel on the same GPU. This means nearly-perfect scaling is attainable in theory, limited only by available GPU memory.

One of the challenges of using GPUs to accelerate particle simulations of polymers driven through MNFDs is incorporating accurate electric field solutions. The standard methods for solving electric fields in complicated geometries (e.g. finite volume or finite element methods) are mesh-based. Users must decompose the problem domain into a mesh, which is often a time-consuming (and, therefore, expensive) process. The resulting field solution is represented in memory as a table containing several numbers per mesh point. Because field must be evaluated repeatedly during simulations, and since GPU-to-CPU communication is slow, it is usually necessary for the field to be stored and evaluated in the GPU memory during simulations. Alas, the memory consumed by the mesh-based field solution directly reduces the CG simulation efficiency by displacing potential threads. Since mesh-based solution accuracy is proportional to mesh resolution, this memory cost cannot easily be reduced.

In this work, we use fully-connected tanh neural networks (NNs) to solve the electric potential in a MNFD geometry. We argue that NN solutions are particularly well-suited for GPU-accelerated CG particle simulations. First, they remove the need for custom mesh design. Second, as we showed in Magill et al. [7], NN representations of PDE solutions are overparametrized, suggesting the compact representation techniques could be used to significantly reduce their memory consumption. We solve the electric potential in the so-called slit-well device (Fig 1(a)) [3, 8–10]. We obtain a reference solution to the potential using the finite element method (FEM) with a high-resolution mesh. We compare the relative accuracy of NN solutions to FEM solutions as a function of memory consumption. The best NNs perform on par with the best FEM solutions, even without compact representation techniques.

2 Methodology

Dissanayake and Phan-Thien [11] showed that neural networks (NNs) can learn to approximate solutions to partial differential equation (PDE) problems using only the information available in the problem statements themselves. Many authors have since explored variations on this method (see [12–17] and others). Berg and Nyström [15] demonstrated that deeper NNs achieved better accuracy for a given memory cost. Sirignano and Spiliopoulos [16] and Han et al. [17] demonstrated that deep NNs could actually solve PDEs in hundreds of dimensions, which is a revolutionary feat. Whereas the computational cost (in memory and time) of mesh-based solvers grows exponentially in the dimensionality of the PDE, Grohs et al. [18] recently released a proof that the cost of solving Black-Scholes PDEs with deep NNs only grows at most polynomially in the dimensionality.

The electric potential in the slit-well device, u , can be modelled by the following PDE problem:

$$\nabla^2 u(x, y) = 0, \quad (x, y) \in \Omega, \quad (1)$$

$$u(x, y) = 1, \quad (x, y) \in (\partial\Omega)_1, \quad (2)$$

$$u(x, y) = -1, \quad (x, y) \in (\partial\Omega)_2, \quad (3)$$

$$u_{\hat{n}}(x, y) = 0, \quad (x, y) \in (\partial\Omega)_i, i \in \{3, 4, \dots, 10\}, \quad (4)$$

where Ω is the problem domain, whose boundaries $(\partial\Omega)_i, i \in \{1, 2, \dots, 10\}$ are illustrated in Fig 1(a). Fully-connected tanh NNs $\tilde{u}(x, y; \vec{p})$ were trained to minimize the loss function

$$\mathcal{L}[u] = \int_{\Omega} \|\nabla^2 u\|^2 dA + \lambda \int_{\partial\Omega} \|B[u]\|^2 dS, \quad (5)$$

where $B[u]$ encodes the boundary conditions (BCs) given by Eqns 2-4. As n minimizes $\mathcal{L}[n]$, it also approximates u . Training is accomplished by batch stochastic gradient descent. The training data are coordinates \vec{x}_i randomly drawn from Ω ; this is equivalent to numerically approximating $\mathcal{L}[\tilde{u}]$ by the Monte Carlo method. Because the boundaries have measure zero in Ω , we treat any training points within a small distance s from the boundaries as being on the nearest boundary. Given this approximation, our loss function effectively has a weighting factor of $\lambda = 10$. To differentiate training difficulties from representational limitations, we also trained NNs by supervised learning to mimic the reference FEM solution. We will refer to training on Eqns 1-4 as the indirect method, and training against the reference solution as the supervised method.

3 Results

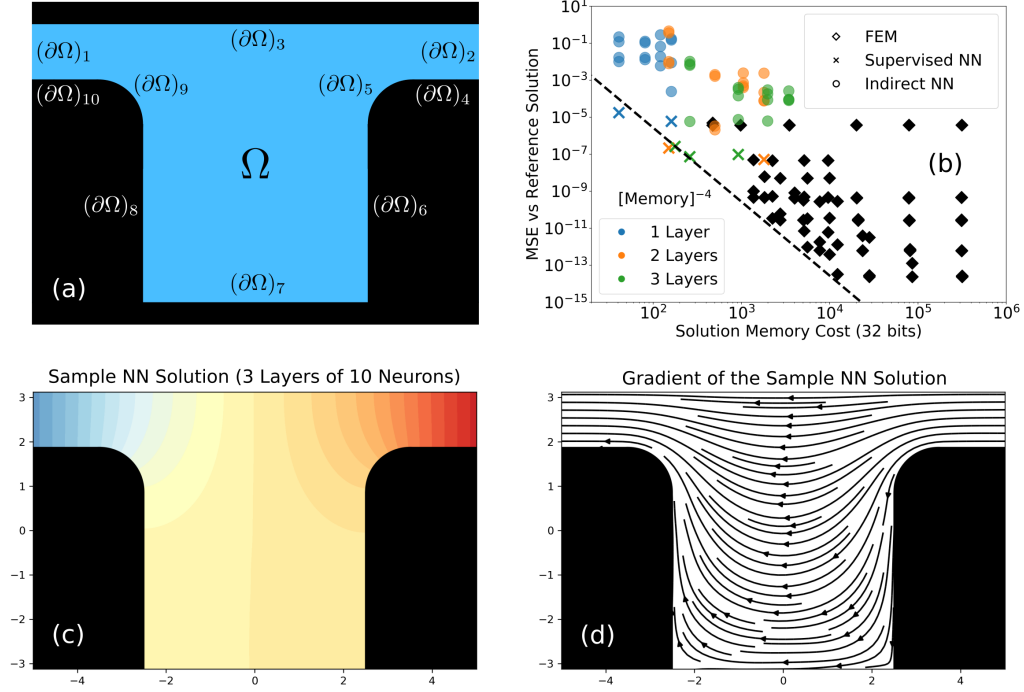


Figure 1: (a) Illustration of the PDE domain. The black boundaries are perfect insulators, modelled by homogeneous Neumann conditions. The two remaining boundaries carry uniform non-homogeneous Dirichlet conditions representing applied voltages. (b) The mean squared error of various numerical solutions relative to the high-resolution reference FEM solution, against the memory cost in increments of 32 bits. NNs trained using Eqns 1-4 are shown as circles, and NNs trained to mimic the reference solution are shown as crosses. Marker color indicates the depth of the networks. The black diamonds show FEM solutions. The dotted reference line scales as the -4 th power of memory consumption. Panels (c) and (d) show a solution learned by an NN directly from Eqns 1-4.

Fig 1(b) compares the accuracy and memory costs (calculated as per App. A) of various NN and FEM solutions. As expected, the FEM solution converge to the reference FEM solution with increasing mesh density. Although the NN solutions exhibited worse accuracy, the best NN accuracies are still acceptable for most CG simulations. Fig 1(c) and (d) illustrate the NN solution corresponding to the bottom-left-most green circle marker in Fig 1(b), which is a good approximation to the true solution.

The dotted line in Fig 1(b) indicates the rough scaling of memory and accuracy observed amidst the best FEM solutions. Although the NNs trained from Eqns 1-4 do not attain this level of performance, some of the NNs trained in a supervised fashion do.

These results demonstrate the feasibility of using NNs to represent solutions to PDE problems for use in GPU-accelerated particle simulations. At the very least, the NN method has the appeal of removing the need for expensive mesh design. Future work will explore higher-dimensional problems, where the relative compactness of NN solutions over mesh-based solutions is expected to become far more pronounced. Indeed, whereas the memory cost of FEM solutions grows exponentially with increasing dimension, Grohs et al. [18] showed it grows polynomially for NNs. Three-dimensional MNFDs are common; time-varying fields in such devices produce four-dimensional fields. Furthermore, in many MNFDs, distortions of the electric field by molecular motion are important. Accounting for this coupling increases the dimensionality of the problem linearly in the number of moving particles. In the higher-dimensional models, the use of compact representation techniques will be crucial if the NN solutions are to be incorporated into GPU-accelerated simulations. In App. B, we have included estimates of the memory costs of FEM solutions for increasing problem dimension.

References

- [1] Rafael Mulero, Anmiv S Prabhu, Kevin J Freedman, and Min Jun Kim. Nanopore-based devices for bioanalytical applications. *JALA: Journal of the Association for Laboratory Automation*, 15(3):243–252, 2010.
- [2] Stephen L Levy and Harold G Craighead. DNA manipulation, sorting, and mapping in nanofluidic systems. *Chemical Society Reviews*, 39(3):1133–1152, 2010.
- [3] Kevin D Dorfman. DNA electrophoresis in microfabricated devices. *Reviews of Modern Physics*, 82(4):2903, 2010.
- [4] Gary W Slater, Christian Holm, Mykyta V Chubynsky, Hendrick W de Haan, Antoine Dubé, Kai Grass, Owen A Hickey, Christine Kingsbury, David Sean, Tyler N Shendruk, and Lixin Zhan. Modeling the separation of macromolecules: A review of current computer simulation methods. *Electrophoresis*, 30(5):792–818, 2009.
- [5] Joshua A. Anderson, Chris D. Lorenz, and A. Travasset. General purpose molecular dynamics simulations fully implemented on graphics processing units. *Journal of Computational Physics*, 227(10):5342 – 5359, 2008. ISSN 0021-9991. doi: <https://doi.org/10.1016/j.jcp.2008.01.047>. URL <http://www.sciencedirect.com/science/article/pii/S0021999108000818>.
- [6] Jens Glaser, Trung Dac Nguyen, Joshua A. Anderson, Pak Lui, Filippo Spiga, Jaime A. Millan, David C. Morse, and Sharon C. Glotzer. Strong scaling of general-purpose molecular dynamics simulations on gpus. *Computer Physics Communications*, 192:97 – 107, 2015. ISSN 0010-4655.
- [7] Martin Magill, Faisal Qureshi, and Hendrick W de Haan. Neural Networks Trained to Solve Differential Equations Learn General Representations. *arXiv preprint arXiv:1807.00042*, 2018.
- [8] Jongyoon Han and Harold G Craighead. Separation of long DNA molecules in a microfabricated entropic trap array. *Science*, 288(5468):1026–1029, 2000.
- [9] Kuang-Ling Cheng, Yu-Jane Sheng, Shaoyi Jiang, and Heng-Kwong Tsao. Electrophoretic size separation of particles in a periodically constricted microchannel, 2008.
- [10] Hanyang Wang, Gary Slater, and Hendrick Haan. Electrophoretic ratcheting of spherical particles in a simple microfluidic device: making particles move against the direction of the net electric field. In *APS March Meeting Abstracts*, 2017.
- [11] M. W. M. G. Dissanayake and N. Phan-Thien. Neural-network-based approximations for solving partial differential equations. *Communications in Numerical Methods in Engineering*, 10(3):195–201, 1994. doi: 10.1002/cnm.1640100303. URL <https://onlinelibrary.wiley.com/doi/abs/10.1002/cnm.1640100303>.
- [12] B. Ph. van Milligen, V. Tribaldos, and J. A. Jiménez. Neural Network Differential Equation and Plasma Equilibrium Solver. *Phys. Rev. Lett.*, 75:3594–3597, Nov 1995. doi: 10.1103/PhysRevLett.75.3594. URL <https://link.aps.org/doi/10.1103/PhysRevLett.75.3594>.
- [13] Isaac E Lagaris, Aristidis Likas, and Dimitrios I Fotiadis. Artificial neural networks for solving ordinary and partial differential equations. *IEEE Transactions on Neural Networks*, 9(5):987–1000, 1998.
- [14] Neha Yadav, Anupam Yadav, and Manoj Kumar. *An introduction to neural network methods for differential equations*. Springer, 2015.
- [15] Jens Berg and Kaj Nyström. A unified deep artificial neural network approach to partial differential equations in complex geometries. *Neurocomputing*, 317:28 – 41, 2018. ISSN 0925-2312. doi: <https://doi.org/10.1016/j.neucom.2018.06.056>. URL <http://www.sciencedirect.com/science/article/pii/S092523121830794X>.
- [16] Justin Sirignano and Konstantinos Spiliopoulos. Dgm: A deep learning algorithm for solving partial differential equations. *Journal of Computational Physics*, 375:1339 – 1364, 2018. ISSN 0021-9991. doi: <https://doi.org/10.1016/j.jcp.2018.08.029>.
- [17] Jiequn Han, Arnulf Jentzen, and Weinan E. Solving high-dimensional partial differential equations using deep learning. *Proceedings of the National Academy of Sciences*, 115(34):8505–8510, 2018. ISSN 0027-8424. doi: 10.1073/pnas.1718942115. URL <http://www.pnas.org/content/115/34/8505>.
- [18] Philipp Grohs, Fabian Hornung, Arnulf Jentzen, and Philippe von Wurstemberger. A proof that artificial neural networks overcome the curse of dimensionality in the numerical approximation of black-scholes partial differential equations. *arXiv preprint arXiv:1809.02362*, 2018.

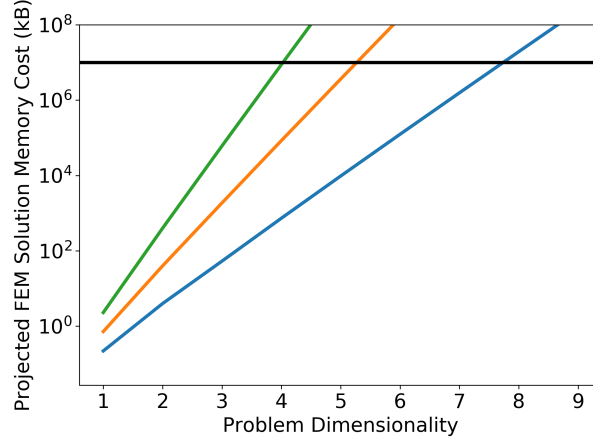


Figure 2: Estimated memory cost of FEM solutions with increasing problem dimensionality assuming constant mesh density. The horizontal black line indicates 10GB, which is the order of magnitude of memory available on most modern GPUs. The three coloured lines consume 4kB, 40kB, and 400kB of memory, respectively, when the dimensionality is 2.

A Calculating memory costs

The number of parameters required to store a fully-connected NN is

$$3w + w(w + 1)(d - 1) + (w + 1), \quad (6)$$

where w is the network's width and d is its depth. Each parameter was stored as single precision floating point numbers, so Eqn. 6 is the memory cost of an NN solution in increments of 32 bits.

The FEM solutions were stored in memory with 3 integers and 3 double precision floating point numbers at every mesh point. Allocating 16 bits per integer and 64 bits per double, the memory cost of a FEM solution in increments of 32 bits is thus

$$2(3N) + 0.5(3N) = 7.5N, \quad (7)$$

where N is the number of mesh points.

The values of w or N for the points in Fig. 1 are implied by Eqn. 6 and 7. The NNs had widths from 8 to 40, and the FEM meshes had roughly 50 to 40,000 mesh points.

B Estimating FEM memory costs in higher dimensions

For every increment that the problem dimensionality is increased, the FEM solution representations will require one extra integer and one extra double at every mesh point. In general, the memory cost of a FEM solution in increments of 32 bits is thus

$$2.5(D + 1)N, \quad (8)$$

where N is the number of mesh points and D is the dimensionality of the problem.

Figure 2 estimates how the memory cost of FEM solutions would scale with problem dimensionality according to Eqn. 8 if the average mesh density is kept constant. The green line corresponds roughly to the densest FEM solutions in Fig. 1; this density would saturate a standard GPU's memory in a four-dimensional domain. The blue line corresponds roughly to the sparsest FEM solutions in Fig. 1; even this density saturates a standard GPU's memory in only eight dimensions.

For reference, modelling the particle-field coupling for a rather small polymer consisting of only 10 monomers in the two-dimensional slit-well device would produce a 22-dimensional PDE. Sirignano and Spiliopoulos [16] demonstrated that NNs could solve such problems with relative ease. Nonetheless, compact representation techniques will almost certainly be necessary to ensure that the NN solutions can be incorporated efficiently into GPU-accelerated simulations. Specifically, the results of Magill et al. [7] suggest that the NN solutions to PDEs are likely to be vastly overparametrized (as is generally the case in other NN applications as well).

Appendix D

Studying first passage problems
using neural networks: A case study
in the slit-well microfluidic device

Studying first passage problems using neural networks: A case study in the slit-well microfluidic device

Andrew M. Nagel , Martin Magill, and Hendrick W. de Haan*

Faculty of Science, University of Ontario Institute of Technology, 2000 Simcoe St N, Oshawa, Ontario, Canada L1H7K4



(Received 27 April 2022; accepted 28 July 2022; published 11 August 2022)

This study presents deep neural network solutions to a time-integrated Smoluchowski equation modeling the mean first passage time of nanoparticles traversing the slit-well microfluidic device. This physical scenario is representative of a broader class of parametrized first passage problems in which key output metrics are dictated by a complicated interplay of problem parameters and system geometry. Specifically, whereas these types of problems are commonly studied using particle simulations of stochastic differential equation models, here the corresponding partial differential equation model is solved using a method based on deep neural networks. The results illustrate that the neural network method is synergistic with the time-integrated Smoluchowski model: together, these are used to construct continuous mappings from key physical inputs (applied voltage and particle diameter) to key output metrics (mean first passage time and effective mobility). In particular, this capability is a unique advantage of the time-integrated Smoluchowski model over the corresponding stochastic differential equation models. Furthermore, the neural network method is demonstrated to easily and reliably handle geometry-modifying parameters, which is generally difficult to accomplish using other methods.

DOI: [10.1103/PhysRevE.106.025311](https://doi.org/10.1103/PhysRevE.106.025311)

I. INTRODUCTION

Micro- and nanofluidic devices (MNFDs) are tools that can be used to manipulate or detect molecules with high precision [1–5]. For instance, the slit-well MNFD was proposed by Han and Craighead [6] as a tool for sorting otherwise free-draining polymers, such as DNA, according to chain length. The same device has also been shown to induce separation of free-draining nanoparticles by size [7,8]. The slit-well is operated by electrophoretically forcing analytes across a periodic array of deeper regions (wells) and shallower regions (slits) between two fixed planes (see Fig. 1). Its sorting effect has been comprehensively studied through theoretical, numerical, and experimental investigations, which have identified a variety of distinct mechanisms that are relevant in different operational regimes. At a high level, the sorting effect depends nonlinearly on the size of the analytes and the magnitude of the applied electric field, as well as the shape and size of the device's wells and slits [2,7–11]. In particular, depending on the choice of these parameters, the mobility of analytes can be made either increasing or decreasing with respect to molecule size.

The practical relevance of biotechnologies such as MNFDs has been stressed in the last year; for instance, Berkenbrock *et al.* [12] surveyed the potential of microfluidics as a means of rapidly testing large numbers of people for COVID-19 infections. Shepherd *et al.* [13] studied a parallelized MNFD that generated scalable lipid nanoparticle formulations needed for applications in RNA therapeutics and vaccines. Nonetheless, the design and optimization of MNFDs is often challenging

because it entails simultaneously considering the influence of many design parameters (e.g., operating voltage, solvent composition, device geometry, etc.) on multiple nonlinearly interdependent phenomena.

In many cases, important biological phenomena can fruitfully be modeled as first passage processes [14]. Moreover, in the study of MNFDs, key transport phenomena are often captured by only the first few moments of an appropriate first passage time distribution. For example, the translocation of a polymer through a nanopore is aptly described as a first passage process, and the mean translocation time is a widely studied metric [15–17]. Magill *et al.* [18] showed that, for the special class of MNFDs with periodic geometries featuring small bottlenecks, the long-term dynamics of molecules driven through the system depend exclusively on the first and second moments of their first passage times across one subunit of the device. The ability to focus on a handful of first passage time moments can greatly simplify the problem of characterizing and designing MNFDs.

This emphasis on the first few moments of the first passage time is of particular interest in light of a convenient mathematical property of the Smoluchowski equation¹ that describes the motion of analytes through MNFDs. For instance, the dynamics of nanoparticles electrophoretically driven through a MNFD can be modeled by the Smoluchowski equation as

$$\rho_t = \nabla \cdot (D \nabla \rho - \mu \vec{E} \rho), \quad (1)$$

¹Note that the Smoluchowski equation is also variously known as the Kolmogorov forward equation, the Fokker-Planck equation, or the convection-diffusion equation, with certain names more common in certain areas of application.

*Hendrick.deHaan@uoit.ca

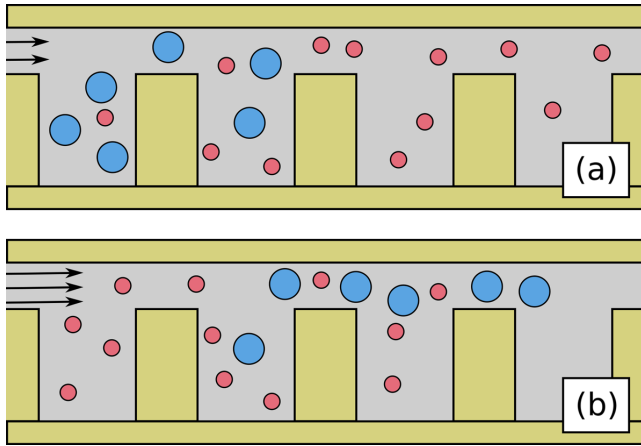


FIG. 1. A schematic of electrophoretic sorting of particles by size in the slit-well device. (a) A weak field causes small particles (red) to traverse the device more quickly on average. (b) A strong field causes large particles (blue) to traverse the device more quickly on average.

where ρ is the position distribution of the particles over space and time, D and μ are the diffusion and free-solution mobility coefficients of the particles, and \vec{E} is the applied electric field. In first-passage problems where the domain geometry and applied fields are time-invariant, Eq. (1) can be integrated over time to obtain the time-integrated Smoluchowski equation [19]

$$-\rho_0 = \nabla \cdot (D\nabla g_0 - \mu\vec{E}g_0), \quad (2)$$

where ρ_0 is the initial condition for ρ . The new field g_0 is defined as

$$g_0(x, y) := \int_0^\infty \rho(x, y, t) dt. \quad (3)$$

The integral of g_0 in any region is the average residence time of particles in that region between initialization and absorption. In particular, it therefore has the property that

$$\int_\Omega g_0 dx = \langle \tau \rangle, \quad (4)$$

when Ω is the entire spatial domain, τ is the stochastic first passage time of the particles to the absorbing boundary conditions, and $\langle \tau \rangle$ is the mean first passage time (MFPT). Moreover, this formulation can be extended recursively to all higher-order moments as well. For instance, the field g_1 satisfying

$$-g_0 = \nabla \cdot (D\nabla g_1 - \mu\vec{E}g_1) \quad (5)$$

has the property that it integrates over the spatial domain to yield the second moment of the first passage time. A more comprehensive discussion of these moment equations can be found in standard references such as Redner [19].

Since the first few moments of first passage time distributions are so important to MNFD phenomena, it is natural to wonder whether solving the moment equations directly might be a useful line of investigation. In practice, however, it appears that this is rarely done. Redner [19] shows the power

of the moment equations for theoretical analysis of first passage problems, especially in the purely diffusive regime where direct analogies with electrostatics can be made. In the context of MNFDs, Magill *et al.* [18] showed that measuring g_0 approximately via particle simulations can aid in understanding the effect of design parameters on system dynamics. Similarly, Wang *et al.* [8] analyzed plots of the time-integrated particle position densities in a periodic model of the slit-well device; however, these maps were constructed in a manner subtly different from g_0 and in particular do not have the property of integrating to the MFPT. The authors are unaware of other studies in which the moment equations are solved numerically towards the goal of understanding the effect of MNFD design parameters on first passage time behavior. Moreover, even though the Smoluchowski equation is also an important mathematical model to study first passage time problems outside biophysics [20–23], we have found no examples in which the g_0 equation (nor any of the higher moment equations) were studied numerically in applied contexts.

A major barrier to the goal of solving the moments equations numerically in biophysics is the so-called curse of dimensionality. That is, for most common numerical methods for partial differential equations (PDEs), the computational cost grows exponentially in the dimensionality of the underlying domain. Thus, whereas highly effective techniques like the finite element method (FEM) can be used to solve PDEs in simple biophysical scenarios, like that of noninteracting nanoparticles, they fail when applied to the high-dimensional PDEs describing the dynamics of many-body systems such as polymers. Indeed, particle-based simulation methods do not exhibit the curse of dimensionality, and this can be seen as a major reason for the dominance of particle simulations over PDE-based calculations in biophysics.

In this work, we investigate a numerical method for PDEs that does not suffer from the curse of dimensionality. The technique, which we refer to as the neural network method (NNM), is inspired by the success of deep learning at solving high-dimensional problems in machine learning, such as image processing and natural language processing [24–26]. A growing body of theoretical and numerical evidence suggests that it can robustly solve high-dimensional PDEs [27–41]. In particular, the NNM has already been used to study high-dimensional problems in biophysics [38].

The NNM has also been shown to solve parametrized problems directly across a continuous range of parameter values [28,42]. As the number of parameters increases, the problem of solving a highly parametrized PDE can exhibit yet another curse of dimensionality. Because the neural network method shares information across parameter space, it is also able to overcome this computational challenge [43–45].

Note that parametrized solutions to PDEs typically cannot be obtained using the FEM, particle simulations, or similar methods. Rather, this goal is usually accomplished using reduced order modeling (ROM) techniques [46,47]. ROM methods typically interpolate between a relatively small number of high-fidelity solutions computed at a handful of reference points in parameter space in order to approximate solutions at new points in parameter space. Whereas most classical ROM methods interpolate to new parameter choices via a linear combination of the reference solutions, the NNM

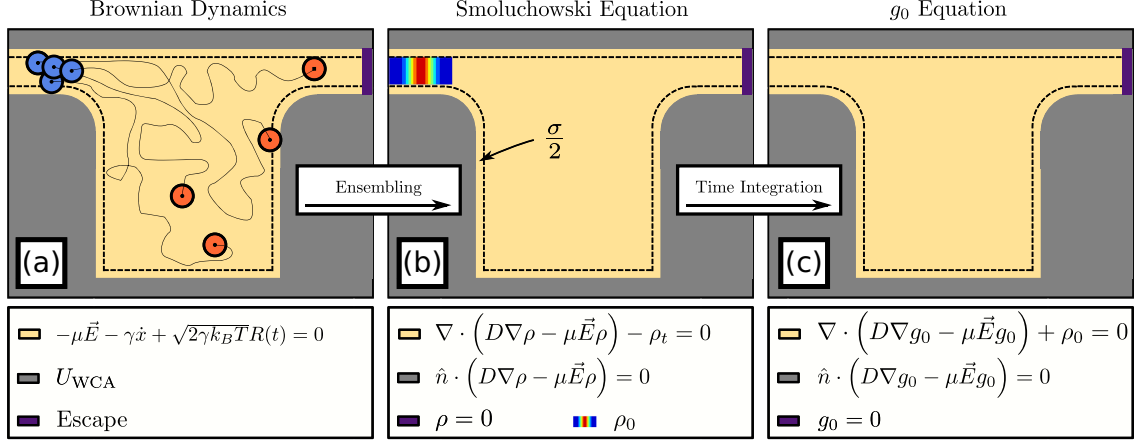


FIG. 2. Schematic of a single periodic subunit of the slit-well device to illustrate passage time models used in this study. (a) Particles are initialized in the left slit (blue particles), confined in the device via a WCA potential on gray walls, undergo Brownian dynamics in the yellow interior (trajectories denoted by black lines) until escaping device at the purple boundary. Dotted line denotes the region of the interior that cannot be occupied by the center of mass of particles. (b) A PDE model of the escape process where the solution ρ satisfies the Smoluchowski equation in the yellow interior. The initial Gaussian band source is located in the left slit, an absorbing boundary in the right slit (purple), and no flux conditions applied on gray walls that move inward to the dotted line to model particle size. (c) A PDE model of the escape process where the solution g_0 satisfies the g_0 equation (first moment) in the yellow interior region. An absorbing boundary is applied at the right slit wall (purple), and no flux conditions are applied on gray boundaries which move inward to the dotted line to model particle size..

is intrinsically nonlinear. Other nonlinear ROMs based on deep neural networks have been proposed in the literature [48–52]. However, these methods require that a database of FEM or other classical solutions be computed prior to training, whereas the NNM simultaneously solves the target PDE and acts as a ROM method over parameter space. In addition, dealing with parameters that modify the domain geometry using classical reduced-order methods can be challenging because these are typically constructed using mesh-based approaches. Although special ROMs can be developed for geometric parameters in some cases [53–55], the mesh-free nature of the NNM is intrinsically advantageous for this application [42,56].

II. PROBLEM DESCRIPTION

The primary goal of this paper is to study the effectiveness of the NNM as a tool for solving the g_0 equation in MNFDs by focusing on a sufficiently complicated representative device, as shown in Fig. 2. The specific problem under consideration is as follows: for an ensemble of noninteracting thermal particles initially located in the left slit of one periodic subunit of the slit-well device, compute the MFPT of these particles to the right slit. Here, the particles are driven by an electric field $\vec{E} = \lambda \vec{E}_0$ for a field strength constant λ and a baseline electric field \vec{E}_0 . The baseline electric field is a solution to Laplace’s equation for a voltage drop of 2 V across the domain. It was obtained using the NNM in the manner described in Magill *et al.* [57], and plots of \vec{E}_0 are included here in Appendix B. The particles represent nanoparticles with diameters σ , diffusion coefficients D , and free-solution mobilities μ . The nanoparticles are assumed to be free-draining, with μ independent of σ , so that separation by size would not occur in free solution. Conversely, the diffusion coefficient is assumed to emerge from Stokes’ law and the fluctuation-dissipation

theorem, so that $D \propto \sigma^{-1}$. For simplicity, these behaviors are implemented as $\mu = 1$ and $D = \sigma^{-1}$. The g_0 equation is thus reduced to

$$F[g_0] \equiv \nabla \cdot \left(\frac{1}{\sigma} \nabla g_0 - \lambda \vec{E}_0 g_0 \right) + \rho_0 = 0, \quad (6)$$

where the particle size σ and field strength λ are the two free parameters, and \vec{E}_0 is the reference electric field.

The problem geometry is shown in Fig. 2 along with depictions of the particle-based, Smoluchowski, and g_0 representations of the problem. The domain is meant to represent a single periodic subunit of the slit-well device illustrated in Fig. 1. In the particle-based model of the problem [Fig. 2(a)], an ensemble of noninteracting nanoparticles are initially located in the left slit, and then these particles proceed to move under a combination of thermal diffusion and electrophoretic drift until reaching the far right purple wall in the right slit. In the Smoluchowski model [Fig. 2(b)], individual particles are eschewed, and the time evolution of the entire distribution of particle positions is modeled instead. Here, the initial position of particles is modeled by the initial condition $\rho_0(x; \sigma)$, located in the left slit. Finally, in the g_0 equation [Fig. 2(c)], the time-dependence of the Smoluchowski is accounted for implicitly by integration over all time. Here, the initial condition $\rho_0(x; \sigma)$ now appears as a source term in the (time-independent) PDE.

In each schematic of Fig. 2, the gray regions represent physical walls. These were modeled as short-range repulsive boundaries (in the particle model; see Appendix A) or no-flux boundary conditions in the continuum models [equations defined in the legends of Figs. 2(b) and 2(c)]. As a result of excluded volume interactions, the particle centers cannot come closer than a distance of roughly $\sigma/2$ from the repulsive boundaries. This exclusion zone is depicted by the dashed black line in Fig. 2. To model this in the continuum models,

the no-flux boundary conditions are applied at the boundary of the exclusion zone (i.e., along the dotted black lines in Fig. 2), rather than at the nominal boundaries (i.e., along the gray walls in Fig. 2).

The nominal dimensions of the domain Ω_0 are the same as those described in Magill *et al.* [57]. In particular, the topmost and bottommost walls are a distance $2L_y = 6.25$ apart, the leftmost and rightmost boundaries are $2L_x = 10$ apart, and the curvature of the re-entrant corners is set to $R = 1$ (see below). The total horizontal lengths of the slits and the well were set equal, to L_x , and the slits were given a height of $h_{\text{slit}} = L_x/4 = 1.25$.

As discussed in Magill *et al.* [57], the standard formulation of the NNM struggles to solve problems exhibiting singularities. For this reason, the re-entrant corners of the slit-well device geometry have been rounded (i.e., represented by circular arcs of finite curvature). Similarly, the NNM was found to perform poorly when the initial distribution of particles was too sharp. Instead, particles were initialized in a Gaussian band in the left slit, given by uniform distribution in y multiplied by a Gaussian distribution in x :

$$\rho_0 = \frac{1}{\sqrt{2\pi}r_s h_{\text{slit}}} \exp\left(-\frac{(x - x_s)^2}{2r_s^2}\right). \quad (7)$$

Here $r_s = 0.25$ is the width of the Gaussian band in the x direction, $h_{\text{slit}} = L_y - y_{\text{slit}} - \sigma$ is the height of the band in the y direction, and $x_s = -L_x + 1$ is the center of the band. Technically, ρ_0 requires a correction factor to be properly normalized over this bounded domain, as the Gaussian distribution in x is normalized over the entire real line, but the discrepancy is numerically insignificant.

The first passage time of the particles is computed when their centers cross the rightmost boundary of the domain for the first time (purple in Fig. 2). In the continuum models, this is represented by an absorbing boundary condition (i.e., a homogeneous Dirichlet condition). Physically, this boundary corresponds to the interface between consecutive periodic subunits of the slit-well, and not to a physical wall. As such, in contrast to the no-flux boundary condition on the gray walls, the placement of the absorbing boundary does not depend on σ .

As a simplifying assumption, particles were prevented from moving through the leftmost boundary of the domain. Mathematically, this was imposed by a no-flux boundary condition. Physically, this corresponds to the synthetic condition that particles cannot move against the direction of the imposed electric field into the previous periodic subunit of the slit-well; we will refer to this as the no-backflow condition. The location of this no-backflow boundary condition was fixed independently of σ .

Of course, in the actual slit-well device there is always a nonzero probability of particle backflow. The simplification was made here because it allows the g_0 equation to be posed in a much simpler domain (i.e., a single periodic subunit). However, as a result of this modeling choice, there will be discrepancies between the MFPT results reported in this paper and the results of previous studies of the slit-well device (such as Cheng *et al.* [7] and Wang *et al.* [8]), especially at low electric field strengths. Nevertheless, as the results in Sec. IV

will indicate, the major features of the slit-well system are preserved despite the no-backflow condition. Furthermore, the simplified model still contains several mathematical features that are expected to be common to many MNFDs and particularly difficult for the NNM to resolve: re-entrant corners, a nonuniform electric field, and nontrivial dependence on physical and geometric problem parameters. As stated above, the purpose of this paper is to study the performance of the NNM when solving a problem with the characteristic features of a typical MNFD problem. Certain features, such as the highly skewed geometry of the fully periodic slit-well and the singularities associated with the fully sharp re-entrant corners, are more technically challenging and relegated to future work.

III. METHODOLOGY

A. Neural network method

The NNM implementation used for this work was similar to that previously described by Magill *et al.* [57]. In the fixed parameter experiments (Sec. IV B 1), the true solution $g_0(\mathbf{x})$ of the g_0 equation [Eq. (6)] was approximated by a deep neural network $\tilde{g}_0(\mathbf{x})$ trained to minimize a composite loss functional

$$\mathcal{L} = \mathcal{L}_{\text{PDE}} + \mathcal{L}_{\text{BC}} + \mathcal{L}_{\text{norm}}. \quad (8)$$

The first loss term consisted of

$$\mathcal{L}_{\text{PDE}}[\tilde{g}_0] = \int_{\Omega} (F[\tilde{g}_0])^2 dA, \quad (9)$$

where F is the operator in the g_0 equation [Eq. (6)]. Thus, $\mathcal{L}_{\text{PDE}}[\tilde{g}_0]$ quantifies the extent to which \tilde{g}_0 satisfied the g_0 equation [Eq. (6)] throughout the domain Ω . Note that, as discussed in Sec. II, Ω depends on σ . The second loss term was defined as

$$\mathcal{L}_{\text{BC}}[\tilde{g}_0] = \int_{\partial\Omega} (B[\tilde{g}_0])^2 ds, \quad (10)$$

where $B[\tilde{g}_0]$ defines no-flux or absorbing boundary conditions (BCs), as appropriate, on each part of the boundary of the domain [see Fig. 2(c)]. Thus, $\mathcal{L}_{\text{BC}}[\tilde{g}_0]$ quantifies the extent to which \tilde{g}_0 satisfied the BCs over the domain boundary $\partial\Omega$.

The final term was given by

$$\mathcal{L}_{\text{norm}}[\tilde{g}_0] = \left[\int_{\Omega} (F[\tilde{g}_0]) dA \right]^2. \quad (11)$$

Similarly to \mathcal{L}_{PDE} , the last loss term $\mathcal{L}_{\text{norm}}$ quantifies the extent to which the approximate solution satisfies the PDE inside the domain Ω . However, whereas \mathcal{L}_{PDE} is a *local* measure of the residual of Eq. (6), $\mathcal{L}_{\text{norm}}$ is a *global* measure. Specifically, \mathcal{L}_{PDE} is the mean of the squared residual, while $\mathcal{L}_{\text{norm}}$ is the square of the mean residual. In theory, $\mathcal{L}_{\text{norm}}$ is a redundant loss term and simply setting \mathcal{L}_{PDE} to zero is sufficient to ensure that \tilde{g}_0 satisfies the g_0 equation [Eq. (6)]. In practice, however, training without $\mathcal{L}_{\text{norm}}$ was found to produce approximate solutions that captured the shape of the true solution fairly accurately, but struggled to converge on the correct magnitude (i.e., differed from the true solution by a small multiplicative factor).

We note that our use of $\mathcal{L}_{\text{norm}}$ is similar to the normalization process used by Al-Arabi *et al.* [58] in solving the time-dependent Smoluchowski equation. The use of $\mathcal{L}_{\text{norm}}$

can also be contrasted with previous work by Avrutskiy [59]. There, Avrutskiy [59] showed benefits to adding redundant loss terms that encourage the solution to satisfy the derivative of the PDE operator $F' = 0$ over the spatial domain. Here, the term $\mathcal{L}_{\text{norm}}$ is a redundant loss term that encourages \tilde{g}_0 to agree with the *integral* of the PDE operator F over the spatial domain Ω . These additional loss terms can be thought of as soft constraints on the training process, or equivalently as regularization terms constructed out of prior knowledge of the target problem.

In Secs. IV B 2–IV B 4, the neural network was parametrized with respect to field strength λ , particle size σ , or both. Thus, the loss terms were redefined as

$$\mathcal{L}_{\text{PDE}}[\tilde{g}_0] = \left\langle \int_{\Omega_\sigma} (F_{\lambda,\sigma}[\tilde{g}_0])^2 dA \right\rangle_{\lambda,\sigma}, \quad (12)$$

$$\mathcal{L}_{\text{BC}}[\tilde{g}_0] = \left\langle \int_{\partial\Omega_\sigma} (B_{\lambda,\sigma}[\tilde{g}_0])^2 ds \right\rangle_{\lambda,\sigma}, \quad (13)$$

$$\mathcal{L}_{\text{norm}}[\tilde{g}_0] = \left\langle \left[\int_{\Omega_\sigma} (F_{\lambda,\sigma}[\tilde{g}_0]) dA \right]^2 \right\rangle_{\lambda,\sigma}. \quad (14)$$

The notations Ω_σ , $F_{\lambda,\sigma}$, and $B_{\lambda,\sigma}$ indicate that the domain changes with σ , and the PDE and BC operators change with both λ and σ . The angled brackets indicate averages over the parameter values. In other words, the loss used for the parametrized neural networks is identical to that used for the fixed parameter experiments, with the additional step of averaging the loss over parameter space.

Note that the electric field \vec{E} was obtained by computing the electric potential u using the NNM methodology of Magill *et al.* [57]. Of course, this is not strictly necessary because u could just as easily be approximated by some other method (e.g., FEM). However, the intention was to illustrate the ease with which previously computed NNM solutions can be fed into the loss functional of new NNM solutions. A contour plot illustrating both u and \vec{E} is included in Appendix B (Fig. 7). Note that the electric potential is defined on the nominal domain Ω_0 corresponding to $\sigma = 0$, which differs from the actual domain Ω on which g_0 is defined.

All of the NNM experiments in this work were conducted with fully connected feedforward neural networks of depth $d = 3$ and width $w = 50$. The hyperbolic tangent was used for activation functions in the hidden layers, while the output layer was linear. To solve the nonparametrized problems (Sec. IV B 1), the approximate solution \tilde{g}_0 was constructed as

$$\tilde{g}_0(\mathbf{x}) = f_{d+1} \circ f_d \circ \cdots \circ f_1(\mathbf{x}), \quad (15)$$

with

$$f_1(\mathbf{x}) = \tanh(W_1 \mathbf{x} + \mathbf{b}_1), \quad (16)$$

$$f_i(\mathbf{x}) = \tanh(W_i f_{i-1}(\mathbf{x}) + \mathbf{b}_i), \quad i = 2, \dots, d, \quad (17)$$

$$f_{d+1}(\mathbf{x}) = \mathbf{W}_{d+1} f_d(\mathbf{x}) + b_{d+1}, \quad (18)$$

where $W_1 \in \mathbb{R}^{w \times 2}$, $W_i \in \mathbb{R}^{w \times w}$ for $i = 2, \dots, d$, and $W_{d+1} \in \mathbb{R}^{1 \times w}$ are the network's weight matrices, while $\mathbf{b}_i \in \mathbb{R}^w$ for $i = 1, \dots, d$, and $b_{d+1} \in \mathbb{R}$ are its biases.

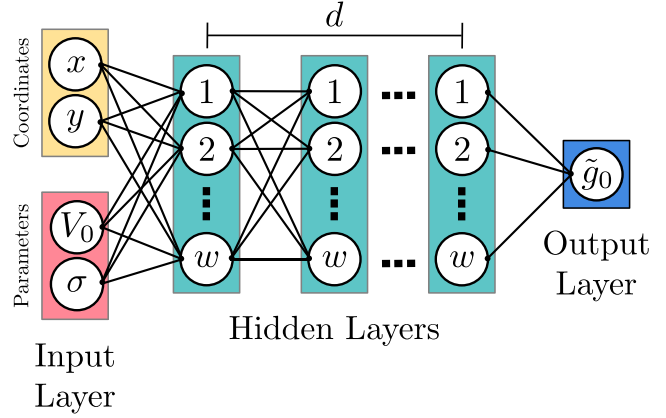


FIG. 3. Fully connected feedforward neural network of width w and depth d mapping coordinates (x, y) and problem parameters (λ, σ) to an output $\tilde{g}_0(x, y; \lambda, \sigma)$. At each node, a weighted sum of the incoming arrows and a bias is computed and passed through an activation function. The network's parameters are optimized such that $\tilde{g}_0(x, y; \lambda, \sigma)$ approximately satisfies the target PDE and BCs.

The experiments in Secs. IV B 2–IV B 4 considered parametrized neural networks, where one or both of the problem parameters λ and σ were included as additional inputs to the network. In these cases, the networks were defined as

$$\tilde{g}_0(\mathbf{x}; \mathbf{m}) = f_{d+1} \circ f_d \circ \cdots \circ f_1(\mathbf{x}; \mathbf{m}), \quad (19)$$

with f_i defined as before for $i = 2, \dots, d + 1$, but with f_1 adjusted to

$$f_1(\mathbf{x}) = \tanh(W_1^{(x)} \mathbf{x} + W_1^{(m)} \mathbf{m} + \mathbf{b}_1), \quad (20)$$

where $W_1^{(x)} \in \mathbb{R}^{w \times 2}$ and $W_1^{(m)} \in \mathbb{R}^{w \times m}$, where m is the length of the parameter vector \mathbf{m} . In other words, the parameters $(\lambda$ or σ or both) were concatenated to the end of the input vector of the network, and the weight matrices were adjusted accordingly. This is illustrated schematically in Fig. 3. The same approach was used by Sirignano and Spiliopoulos [28] and Hennigh *et al.* [42] but can be contrasted with the recently proposed DeepONet architecture of Lu *et al.* [60].

Training was conducted in TENSORFLOW [61] version 1.15 with all unspecified hyperparameters set to their default values. The weights were initialized using the Glorot method [62], and biases were initialized to zero. Weights were iteratively updated using the Adam optimizer [63] to minimize \mathcal{L} with the learning rate set to 10^{-3} in Sec. IV B 1, and to 10^{-4} in Secs. IV B 2–IV B 4. In each iteration, the integrals in \mathcal{L} were approximated by Monte Carlo sampling using the same procedure described in Magill *et al.* [57]. Specifically, rejection sampling was applied to 10 000 nominal samples generated in the bounding box $[-5, 5] \times [-3.125, 3.125]$, and each smooth subunit of the boundary was randomly sampled with a linear density of about 13 points per unit length. For the parametrized network experiments (Secs. IV B 2–IV B 4), the relevant problem parameters were also sampled randomly in each training iteration. These samples were generated uniformly at random, with λ drawn from [5,50] and σ drawn from [0.125,0.625]. In particular, it was necessary to sample the parameter σ before sampling points in Ω , since the extent

of Ω varies with σ . One random parameter vector was drawn per training iteration.

The testing loss was evaluated every 1000 training iterations, using ten times more samples than during a training step. In the parametrized network experiments (Secs. IV B 2–IV B 4), the testing loss was averaged across 100 random parameter vectors. Training was continued for a fixed number of iterations (600 000 epochs for the fixed parameter experiments, and 30 000 000 epochs for the parametrized network experiments). The final network was taken as that which achieved the lowest testing loss across all iterations.

B. Finite element method

The MFPT problem in the slit-well domain cannot be solved exactly in closed form due to the complex nature of the geometry. Instead, approximate ground truth solutions to the problem were obtained using the finite element method (FEM). Following Magill *et al.* [57], the problem for g_0 was solved using a mixed FEM formulation implemented in FEniCS [64]. The electric field \vec{E} included in the PDE [Eq. (6)] was obtained by also approximating the electric potential u by a mixed FEM formulation. As stated above, u is defined on the nominal domain Ω_0 , whereas g_0 is defined on a smaller domain depending on σ . Thus, for the FEM solutions it was necessary to first solve u and \vec{E} on a discretization of Ω_0 , project \vec{E} onto a discretization of the appropriate Ω , and then define the variational problem for g_0 on Ω .

The mesh decomposition of the domain was conducted using the MSHR package in FENICS. The resolution parameter was set to 200, and the re-entrant corners were approximated linearly by 400 segments each. The same mesh parameters were used for all values of σ , and for the nominal domain, Ω_0 , on which u was solved.

IV. RESULTS

This section details results obtained using the NNM to solve the g_0 equation modeling the MFPT of nanoparticles driven through the slit-well device (described in Sec. II). The focus throughout is on the relationship between key problem parameters and observables of physical interest, where g_0 acts as a proxy between the two. The first observable of interest is, naturally, the mean first passage time $\langle\tau\rangle$. As described in Eq. (4), $\langle\tau\rangle$ can be obtained by integrating g_0 over the domain Ω . Throughout this paper, the integration of g_0 to estimate $\langle\tau\rangle$ is accomplished using the same Monte Carlo procedure described for \mathcal{L}_{PDE} in Sec. III.

In practice, an observable of greater interest than the mean first passage time itself is the net electrophoretic mobility of the nanoparticles through the slit-well device over long timescales [7,8]. In particular, the electrophoretic mobility is typically defined as

$$\mu_{\text{electro}} := \lim_{t \rightarrow \infty} \frac{\langle x \rangle_t}{E_c t}, \quad (21)$$

where $\langle x \rangle_t$ is the ensemble average of the x position at time t , and E_c is a characteristic scale for the applied electric-field strength. It is not clear whether μ_{electro} can be inferred directly from the g_0 problem being solved here. Instead, the present

paper will investigate a similar observable of interest, which will be called the effective mobility

$$\mu_{\text{eff}} := \frac{L_0/\langle\tau\rangle}{E_c} = \frac{1}{\lambda\langle\tau\rangle}, \quad (22)$$

where L_0 is the mean horizontal distance from ρ_0 to the absorbing wall. The characteristic field strength is chosen of the form $E_c = V_c/L_c$, where V_c is a characteristic voltage drop and L_c is a characteristic length scale. Since the overall voltage drop across the system is of order one and proportional to the field strength λ , we choose $V_c = \lambda$. For numerical simplicity, we also choose $L_c = L_0$, thus obtaining the final equality in Eq. (22). The effective mobility is expected to exhibit similar features to the electrophoretic mobility because both consist of characteristic particle velocities divided by characteristic electric-field strengths. A comprehensive exploration of the relationship between the two mobility definitions is left to future work.

A. Characteristics of g_0

Figure 4 shows contour plots of g_0 solutions computed using the NNM, with the corresponding estimates of $\langle\tau\rangle$ and μ_{eff} shown in the legends. The four subplots correspond to the four essential parameter regimes alluded to in Fig. 1. Note that the magnitude of the color scale varies across the four subplots.

First, consider the solution of g_0 in Fig. 4(a) corresponding to small particles ($\sigma = 0.125$) driven by a weak field ($\lambda = 5.0$). Here, g_0 has a maximum in the left slit near the peak of the initial particle distribution ρ_0 . Naturally, since the particle positions are initialized according to ρ_0 , the average residence time in that region is relatively high; this feature is common to all four subplots in Fig. 4. Outside the left slit, g_0 decreases nearly monotonically from left to right, eventually reaching a value of zero on the absorbing boundary. The shape of this function is nearly visually indistinguishable from the solution with $\sigma = 0.125$ and $\lambda = 0$ (not shown) and is characteristic of predominantly diffusive dynamics in all regions of the domain.

Figure 4(b) again shows g_0 for small particles ($\sigma = 0.125$), but now driven by a much stronger field ($\lambda = 50.0$). In contrast with the monotonically decreasing solution in Fig. 4(a), in this scenario g_0 is relatively constant throughout most of the domain until a boundary layer near the absorber. In fact, here g_0 even exhibits some minor nonmonotonic features: a shadow is evident in the bottom-left of the well, and a local maximum is attained at the entrance to the right slit. Drift and diffusion effects are relatively balanced in this case, with the uniformity in x reflecting strongly driven motion in the horizontal direction, and the uniformity in y reflecting rapid diffusion in the vertical direction.

Figure 4(c) shows g_0 for large particles ($\sigma = 0.625$) driven by a weak field ($\lambda = 5.0$). Notice that the walls of the domain are shifted inward by 0.5σ , reflecting the reduced area that can be occupied by the center of mass of larger particles (Sec. II). In this scenario, the smaller diffusion coefficient of the larger particles balances the weaker field, resulting in a solution that more closely resembles that in Fig. 4(b) than that in Fig. 4(a). However, the solution in Fig. 4(c) is visibly

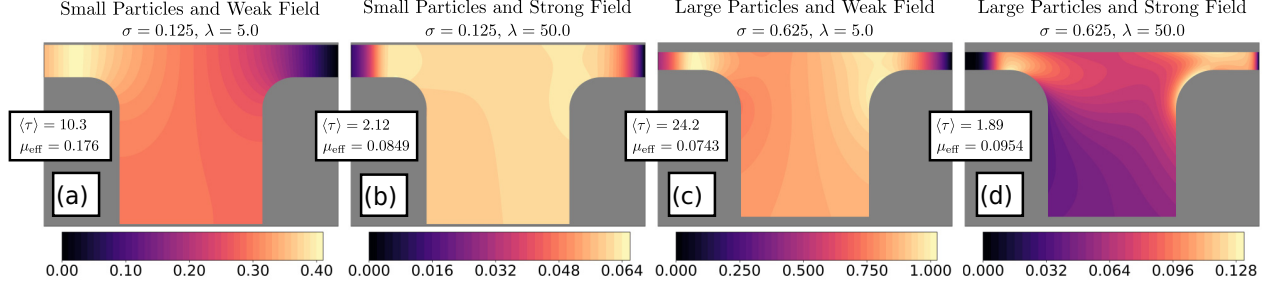


FIG. 4. NNM solutions to the g_0 equation subject to (a) a small particle size and a weak electric field, (b) a small particle size and a strong electric field, (c) a large particle size and a weak electric field, and (d) a large particle size and a strong electric field.

increasing from left to right across the well, in contrast with both the solutions in Figs. 4(a) and 4(b). As was the case in Fig. 4(b), drift and diffusion are of comparable importance; the differences between the two solutions are primarily due to the modifications to the domain geometry.

Finally, Fig. 4(d) shows g_0 computed for large particles ($\sigma = 0.625$) subject to a strong electric field ($\lambda = 50.0$). Here, the shape of the solution differs significantly from those in all of Figs. 4(a)–4(c). In Fig. 4(d), g_0 takes on very small values throughout the entire well, and decreases substantially from the top of the well to its bottom. The combination of the low diffusion coefficient and the very strong driving force causes the large particles to remain primarily streamlined in the upper region of the well as they move rapidly from ρ_0 to the absorber.

The MFPTs $\langle \tau \rangle$ and effective mobilities μ_{eff} in the four scenarios of Fig. 4 are consistent with the expected sorting mechanisms in each regime [7]. When the field is strong, smaller particles have a larger $\langle \tau \rangle$ and lower μ_{eff} than larger particles. The converse is true at weak fields.

Future work should explore the relationship of g_0 , $\langle \tau \rangle$, and μ_{eff} with standard explanations for these phenomena, such as the entrance effect [7,65]. The purpose of the discussion in this section was to illustrate the variety of complicated behavior that arise in g_0 solutions across the different physically meaningful parameter regimes in the slit-well. In Sec. IV B, parametrized NNM solutions will be trained to interpolate nonlinearly between all four solutions in Fig. 4. Ultimately, in Sec. IV B 4 this will yield continuously differentiable mappings between both problem parameters λ and σ and both key physical observables $\langle \tau \rangle$ and μ_{eff} , thereby capturing the entirety of this rich sorting mechanism in a single numerical solution.

B. Benchmarking the neural network method against the finite element method

In this section, g_0 will be leveraged as a proxy for the calculation of the metrics $\langle \tau \rangle$ and μ_{eff} . In practice, it is common in MNFD research and development (and scientific research more broadly) to study how such key metrics change in response to variations in the system parameters. The simplest approach to characterizing this variation is to compute or measure the metrics independently for a large number of parameter choices. In Sec. IV B 1, the NNM is applied to

precisely this task of calculating $\langle \tau \rangle$ and μ_{eff} for many combinations of particle size σ and field strength λ .

The above approach, however, requires repeated calculation of the key metrics which can be expensive when considering many independent parameters. As discussed in Sec. I, the NNM can be leveraged to solve such parametrized problems directly across continuous ranges of parameter values. The high-dimensional function $g_0(x, y; \lambda, \sigma)$ implicitly encodes $\langle \tau \rangle$ and μ_{eff} as continuously differentiable functions of σ and λ . The NNM is used to approximate this function directly in Secs. IV B 2–IV B 4, for g_0 solutions parametrized directly by λ , σ , or both simultaneously.

Throughout Sec. IV B, four quantities are used to characterize the performance of the NNM across parameter space. These quantities are all plotted in Fig. 5, with each column corresponding to one of the four NNM formulations discussed above. Naturally, both the MFPT $\langle \tau \rangle$ and the effective mobility μ_{eff} are included in the analysis. These are plotted in the first two rows [Figs. 5(a)–5(d) and 5(e)–5(h)], respectively, alongside the reference values computed using FEM. The NNM results are indicated by lines, and the corresponding FEM results are included as stars. Dotted lines in Figs. 5(a) and 5(e) connect values that are only computed at discrete parameter choices, whereas solid lines used everywhere else indicate values that are computed over continuous parameter ranges.

Next, in order to quantify the accuracy of the $\langle \tau \rangle$ values obtained via the NNM, the relative error ε with respect to the ground truth FEM solution is computed. Specifically, ε is defined as the relative error of $\langle \tau \rangle$ with respect to $\langle \tau \rangle_{\text{FEM}}$, i.e.,

$$\varepsilon = \frac{|\langle \tau \rangle - \langle \tau \rangle_{\text{FEM}}|}{\langle \tau \rangle_{\text{FEM}}}, \quad (23)$$

where $\langle \tau \rangle$ and $\langle \tau \rangle_{\text{FEM}}$ are the MFPTs computed by the NNM and FEM, respectively. The relative errors ε are plotted in Figs. 5(i)–5(l). Here, circular markers indicate the discrete parameter choices at which ε was computed. Additionally, the plots in Figs. 5(i)–5(l) contain a dotted black line at 10^{-2} , corresponding to a relative error of 1%. This is representative of a relative error threshold that is typically attainable and acceptable in MNFD research. Indeed, Appendix A describes standard particle simulations that were used to approximate $\langle \tau \rangle$ with relative errors comparable to or below 1% for all choices of parameters λ and σ .

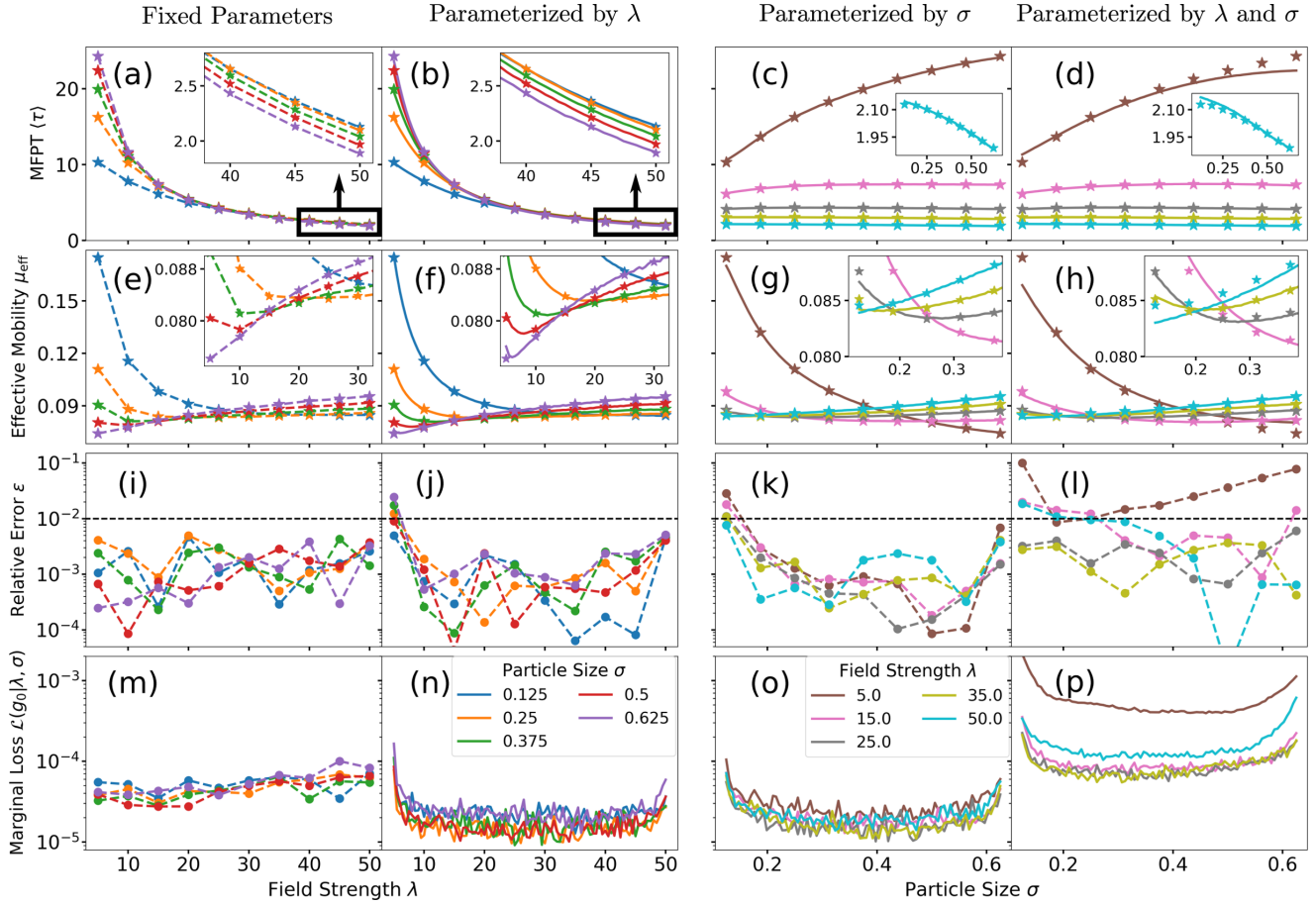


FIG. 5. Analysis of g_0 solutions computed using the NNM. Mean first passage times $\langle\tau\rangle$ for the NNM (a) with fixed parameters, and parametrized by (b) λ , (c) σ , and (d) both λ and σ . Star markers denote values obtained using the FEM, and insets display behavior at high field strengths. Effective mobilities μ_{eff} for the NNM (e) with fixed parameters, and parametrized by (f) λ , (g) σ , and (h) both λ and σ . Star markers denote values obtained using FEM, and insets zoom in on the minimum of the curves. Relative errors ε computed against the FEM for the NNM (i) with fixed parameters, and parametrized by (j) λ , (k) σ , and (l) both λ and σ . Dotted black line denotes 1% error baseline computed by particle simulations. (m) Testing loss of the NNM with fixed parameters, and marginal loss of the NNM parametrized by (n) λ , (o) σ , and (p) both λ and σ .

The final quantity included in the analysis is similar to the loss functional \mathcal{L} used during the NNM training process [Eq. (9)]. However, the total loss provides only a single characterization of a network's performance over its entire domain. When the NNM was used to solve parametrized g_0 problems, it was valuable to evaluate the relative performance of these solutions at different points in parameter space. To this end, we defined the marginal loss $\mathcal{L}(g_0|\lambda, \sigma)$, a parameter-dependent generalization of the total loss. As with the true loss, $\mathcal{L}(g_0|\lambda, \sigma)$ is the sum of the L_2 norms of the PDE-, BC-, and norm-based residuals. However, whereas the total loss averages these quantities over all choices of λ and/or σ [Eqs. (12)–(14)], the corresponding terms in $\mathcal{L}(g_0|\lambda, \sigma)$ were instead treated as functions of λ and σ . In the case of nonparametrized NNM solutions, the marginal loss definition simply reduces to the original total loss.

The marginal losses are plotted in Figs. 5(m)–5(p). The values in Fig. 5(m) correspond to NNM solutions trained at fixed parameter choices and are thus indicated by discrete circular markers. Conversely, since $\mathcal{L}(g_0|\lambda, \sigma)$ can be evaluated continuously for parametrized solutions, the corresponding

marginal loss values shown in Figs. 5(n)–5(p) are indicated by solid lines sampled finely throughout the parameter space.

1. Neural network method with fixed parameters

This section contains a discussion of the results in Figs. 5(a), 5(e), 5(i), and 5(m). Here, the NNM was applied repeatedly to solving the g_0 equation for fixed choices of the problem parameters: field strength λ and particle size σ . This NNM formulation is the same as the one used in Sec. IV A but is now applied to many more choices of the problem parameters. Specifically, the results are shown for ten choices of λ uniformly spaced from 5 to 50 and five choices of σ uniformly spaced from 0.125 to 0.625, with a distinct neural network used to approximate g_0 for each parameter combination.

For small values of λ in Fig. 5(a), $\langle\tau\rangle$ is monotonically increasing with σ , and for large values of λ (see the inset) the opposite is true. Moreover, the finer sampling of parameter space resolves new features that were not clear from examining only the four samples in Sec. IV A. For instance, Fig. 5(a)

shows that $\langle \tau \rangle$ decreases monotonically with λ for each choice of σ . In addition, the dependence of $\langle \tau \rangle$ on σ is much stronger at low field strengths.

The same sorting behavior can be viewed from a different perspective via μ_{eff} in Fig. 5(e). In addition, the crossover in sorting order around $\lambda \approx 25$ is better resolved by μ_{eff} than $\langle \tau \rangle$. Indeed, in Fig. 5(e) it is clear that there is no single value of λ for which $\langle \tau \rangle$ and μ_{eff} are entirely independent of σ . Of course, the results discussed above for $\langle \tau \rangle$ and μ_{eff} are not novel because they are consistent with published results on the slit-well device (see, e.g., Cheng *et al.* [7]). Rather, the purpose of this discussion is to illustrate two points: first, that valuable information can be extracted by studying the variation of key output metrics (here, $\langle \tau \rangle$ and μ_{eff}) as functions of the key input parameters (here, λ and σ) and, second, that the physical problem being studied in this paper (Sec. II) indeed captures essentially the same physical mechanisms expected for the actual slit-well system.

Before considering the benefits of the more ambitious parametrized NNM formulations, it is important to assess how accurately the NNM resolves $\langle \tau \rangle$ and μ_{eff} when applied to the simpler task of solving g_0 at a single point in parameter space. The accuracy is quantified in Fig. 5(i), which shows ε , the relative error in $\langle \tau \rangle$. In this plot, it appears that ε is roughly independent of both σ and λ , suggesting that the current implementation of the NNM is fairly robust throughout the problem parameter space. This is corroborated by the testing losses $\mathcal{L}(g_0|\lambda, \sigma)$ plotted in Fig. 5(m), which are also roughly independent of the problem parameters. Most importantly, for all choices of parameters λ and σ in Fig. 5(i), ε is well below the 1% error threshold indicated by the black line. In other words, the NNM is at least as effective at resolving $\langle \tau \rangle$ as the Brownian dynamics particle simulations included in Appendix A.

2. Neural network method parameterized by field strength

Whereas in Sec. IV B 1, 50 networks were used to obtain 50 different g_0 solutions, which were then integrated over their respective domains to produce 50 different $\langle \tau \rangle$ measurements, in this section only five networks are utilized to accomplish the same goal. Each of these five networks solves $g_0(x, y; \lambda)$ for $\lambda \in [5, 50]$ at a fixed choice of σ . As in Sec. IV B 1, the metrics $\langle \tau \rangle$, μ_{eff} , ε , and $\mathcal{L}(g_0|\lambda, \sigma)$ are computed from the solutions; these are plotted in Figs. 5(b), 5(f), 5(j), and 5(n), respectively. Comparing Figs. 5(b) with 5(a) and Fig. 5(f) with 5(e), it is clear that the NNM formulation parametrized by λ recovers the same results previously obtained by solving g_0 independently for many different parameter choices in Sec. IV B 1.

One advantage of the parametrized NNM formulation is evident in the inset of Fig. 5(f). For each choice of σ , there is a λ value for which μ_{eff} is minimal. When computing μ_{eff} only at discrete choices of the parameters [as in Fig. 5(e)], the exact location of these minima is not clear. Instead, the results in Fig. 5(f) illustrate that the parametrized NNM formulation naturally resolves the existence of local minima, since the solution is trained continuously for all parameter values in the training domain. The benefit of continuous mappings from problem parameters to key output metrics becomes more valu-

able as dimensionality of parameter space is increased (e.g., as explored in Sec. IV B 4).

The relative error ε and marginal loss $\mathcal{L}(g_0|\lambda, \sigma)$ in Figs. 5(j) and 5(n) quantify the accuracy of the $g_0(x, y; \lambda)$ solution. Here, both ε and $\mathcal{L}(g_0|\lambda, \sigma)$ are highest at the boundaries of the λ training range and fairly uniform throughout the majority of the interior of the training range. In particular, both are highest at the left boundary, $\lambda = 5$. This relationship between error and loss is similar to those studied in Magill *et al.* [57] and provide further justification for using the (marginal) loss as an *a posteriori* method for gauging the reliability of NNM solutions.

The deterioration in performance seen in Figs. 5(j) and 5(n) at the boundaries of the λ training range can likely be attributed to the uniform Monte Carlo sampling of λ during training. The exact endpoints have very low probabilities of being sampled directly; moreover, their neighborhoods are only sampled on one side, whereas the neighborhoods of points nearer to the middle of the λ training range are sampled thoroughly on both sides. This could effectively lead to an under-representation of the behavior near the endpoints in the training loss. Characterizing this tentative mechanism is beyond the scope of the present work.

Overall, only three of the 50 relative errors in Fig. 5(j) slightly exceed the 1% error threshold. Thus, the implementation of the parametrized NNM studied in this section meets the standard of accuracy typically attained by Brownian dynamics (BD) simulations. In the regions of parameter space where the relative error was not measured directly, the marginal loss [Fig. 5(n)] provides an *a posteriori* estimate of the error, suggesting that the NNM's performance is excellent except for λ values very close to the boundaries of the training range. Altogether, these results demonstrate that the NNM is a feasible technique for solving the g_0 problem over a continuous range of field strengths. Moreover, using g_0 as a proxy for $\langle \tau \rangle$ and μ_{eff} enables the NNM to resolve the behavior of these key output metrics continuously over the target parameter range.

3. Neural network method parameterized by particle size

To expand upon the unique strengths of the NNM, this section will consider the problem of solving g_0 as a function of the particle size σ . Here, since the diffusion coefficient is being modeled as $D = \sigma^{-1}$, the terms of the g_0 PDE depend directly on the parameter σ , just as they depend directly on λ . However, the location of the boundaries of the slit-well domain also depend explicitly on the parameter σ (Sec. II). Thus, whereas λ only modified the PDE terms, σ modifies both the PDE terms and the domain geometry. As described in Sec. I, it is challenging for classical reduced-order methods to deal with parametrized domain geometries. However, this section will demonstrate that the NNM can handle geometry-modifying parameters (σ) just as easily as parameters that do not modify the domain geometry (λ).

Once again, the MFPT $\langle \tau \rangle$, effective mobility μ_{eff} , relative error ε , and marginal loss $\mathcal{L}(g_0|\lambda, \sigma)$ are computed from the NNM solutions and plotted in Figs. 5(c), 5(g), 5(k), and 5(o), respectively. Whereas the results in Figs. 5(a), 5(e), 5(i), and 5(m) and Figs. 5(b), 5(f), 5(j), and 5(n) for Secs. IV B 1–IV B 2 were solved and plotted as functions of λ , the results

for this section are presented as functions of σ . Specifically, each curve in Figs. 5(c), 5(g), 5(k), and 5(o) represents a single neural network trained over the range $\sigma \in [0.125, 0.625]$ at a fixed choice of λ [indicated by the legend in Fig. 5(o)].

The $\langle \tau \rangle$ and μ_{eff} measurements in Figs. 5(c) and 5(g) indicate that the NNM parametrized by σ recovers the same physical properties observed for the NNM with fixed parameters (Sec. IV B 1) and the NNM parametrized by λ (Sec. IV B 2). For the $\langle \tau \rangle$ measurements in Fig. 5(c), the $\lambda = 5.0$ curve is monotonically increasing whereas the $\lambda = 50.0$ curve (enhanced in the inset) is monotonically decreasing. This implies that small particles have a lower MFPT at low field strengths, and large particles have a lower MFPT at high field strengths. Likewise, the $\lambda = 5.0$ curve for μ_{eff} in Fig. 5(g) indicates that small particles are more mobile at low field strengths, whereas the $\lambda = 50.0$ curve indicates that large particles are more mobile at high field strengths.

Visual comparison of the NNM results (solid lines) to the ground-truth FEM results (stars) in Figs. 5(c) and 5(g) suggests good agreement between the two through most of the parameter space. However, the effective mobilities computed by the NNM in Fig. 5(g) deviate noticeably from the FEM results at the left endpoint $\sigma = 0.125$. Accordingly, the relative errors and marginal losses plotted in Figs. 5(k) and 5(o) are also highest at $\sigma = 0.125$. In fact, ε and $\mathcal{L}(g_0|\lambda, \sigma)$ of the NNM solutions parametrized by σ [Figs. 5(k) and 5(o)] exhibit the same structure previously identified (Sec. IV B 2) in ε and $\mathcal{L}(g_0|\lambda, \sigma)$ of the NNM solutions parametrized by λ [Figs. 5(j) and 5(n)]. That is, ε and $\mathcal{L}(g_0|\lambda, \sigma)$ are roughly uniform for intermediate values of σ but increase sharply near the boundaries of the training domain. In particular, ε and $\mathcal{L}(g_0|\lambda, \sigma)$ are consistently higher at the low- σ endpoint than at the high- σ endpoint.

Overall, the relative error in Fig. 5(k) is well below the 1% error threshold for most of the training range. As was the case in Sec. IV B 2, at the few points where relative error exceeds 1%, it only does so by a small amount. The marginal loss continues to behave as an *a posteriori* measure of solution accuracy and suggests that the regions of high relative error are once again concentrated near the endpoints of the training range. Despite the fact that parameter σ directly changes the domain geometry in addition to modifying the terms of the PDE, the performance measured in this section is essentially the same as that reported in Sec. IV B 2, where the NNM was parametrized by the simpler parameter λ . Thus, it appears that the NNM can handle geometry-modifying parameters just as easily as parameters that do not modify domain geometry. This is particularly interesting given the difficulty of treating parametrized geometries with other reduced-order modeling techniques.

4. Neural network method parameterized by field strength and particle size

The results shown so far have established that the NNM can robustly solve the g_0 equation in the slit-well MNFD (Sec. IV B 1), and that the method can easily be extended to produce solutions parametrized by field strength λ (Sec. IV B 2) or particle size σ (Sec. IV B 3). Expanding upon this capability, in this section the NNM is used to approximate

g_0 as a function of both λ and σ simultaneously (Fig. 3). Specifically, a single neural network is trained to approximate the four-dimensional function $g_0(x, y; \lambda, \sigma)$ over the same parameter space previously spanned by five networks in Secs. IV B 2 and IV B 3 or 50 networks in Sec. IV B 1.

The MFPT $\langle \tau \rangle$, effective mobility μ_{eff} , relative error ε , and marginal loss $\mathcal{L}(g_0|\lambda, \sigma)$ are computed from the NNM solution $g_0(x, y; \lambda, \sigma)$ and plotted in Figs. 5(d), 5(h), 5(l), and 5(p). The lines are shown as functions of σ and evaluated at the same choices of λ used in Sec. IV B 3 [indicated by the legend in Fig. 5(o)]. The $\langle \tau \rangle$ and μ_{eff} values plotted in Figs. 5(d) and 5(h) closely match those in Figs. 5(c) and 5(g), demonstrating that the NNM parametrized by both λ and σ can resolve all the same major physical phenomena previously identified in Secs. IV B 2–IV B 3.

However, the accuracy of the solution $g_0(x, y; \lambda, \sigma)$ is slightly worse than that observed in the previous sections [Figs. 5(a)–5(c) and 5(e)–5(g)] as visible in $\langle \tau \rangle$ and μ_{eff} [Figs. 5(d) and 5(h)] and quantitatively confirmed by ε and $\mathcal{L}(g_0|\lambda, \sigma)$ [Figs. 5(l) and 5(p)]. This is not entirely surprising, since the four-dimensional problem here is intrinsically more difficult than the three-dimensional (Secs. IV B 2 and IV B 3) and two-dimensional formulations (Sec. IV B 1) of the problem. Moreover, the network depth and width were held constant over all experiments, and the training time was held constant for all the parametrized formulations (Sec. III A). Regardless, although $g_0(x, y; \lambda, \sigma)$ appears somewhat less accurate than the solutions from previous sections, it generally still meets the target 1% error threshold over most of its parameter training range.

An exception to this statement is presented by the results at $\lambda = 5$ [the brown lines in Figs. 5(d), 5(h), 5(l), and 5(p)], for which the error of $g_0(x, y; \lambda, \sigma)$ is greater than 1% over nearly the entire σ training range. The marginal loss also reflects this poor performance; for $\lambda = 5$, $\mathcal{L}(g_0|\lambda, \sigma)$ in Fig. 5(p) is more than an order of magnitude larger than $\mathcal{L}(g_0|\lambda, \sigma)$ from all previous experiments [i.e., those in Figs. 5(m)–5(o)], and several times larger than the other $\mathcal{L}(g_0|\lambda, \sigma)$ curves in Fig. 5(p). Conspicuously, the $\mathcal{L}(g_0|\lambda, \sigma)$ curves in Fig. 5(p) vary significantly with λ , whereas in Figs. 5(m)–5(o) very little variation was observed between the different $\mathcal{L}(g_0|\lambda, \sigma)$ curves.

Of course, the results in Fig. 5(p) differ fundamentally from those in Figs. 5(m)–5(o); whereas each curve in Figs. 5(m)–5(o) corresponds to one or more independent networks, all the curves in Fig. 5(p) are generated by a single network. In fact, the brown ($\lambda = 5$) and blue ($\lambda = 50$) curves, which exhibit the highest marginal losses in Fig. 5(p), lie directly on the boundary of the network's (λ, σ) training domain. When analyzing the NNM parametrized by λ or σ (Secs. IV B 2 and IV B 3), a substantial deterioration in accuracy was found to be highly localized near the boundaries of the parameter training range. If a similar boundary effect exists here for the $g_0(x, y; \lambda, \sigma)$ solution, then the results in Figs. 5(d), 5(h), 5(l), and 5(p) are not representative of the solution's overall accuracy over the entire problem parameter space, as essentially half of the data shown in those plots lie on the boundary of the network's parameter training space.

To investigate this possibility, the same metrics that are shown as discrete lines in Figs. 5(d), 5(h), 5(l), and 5(p) are

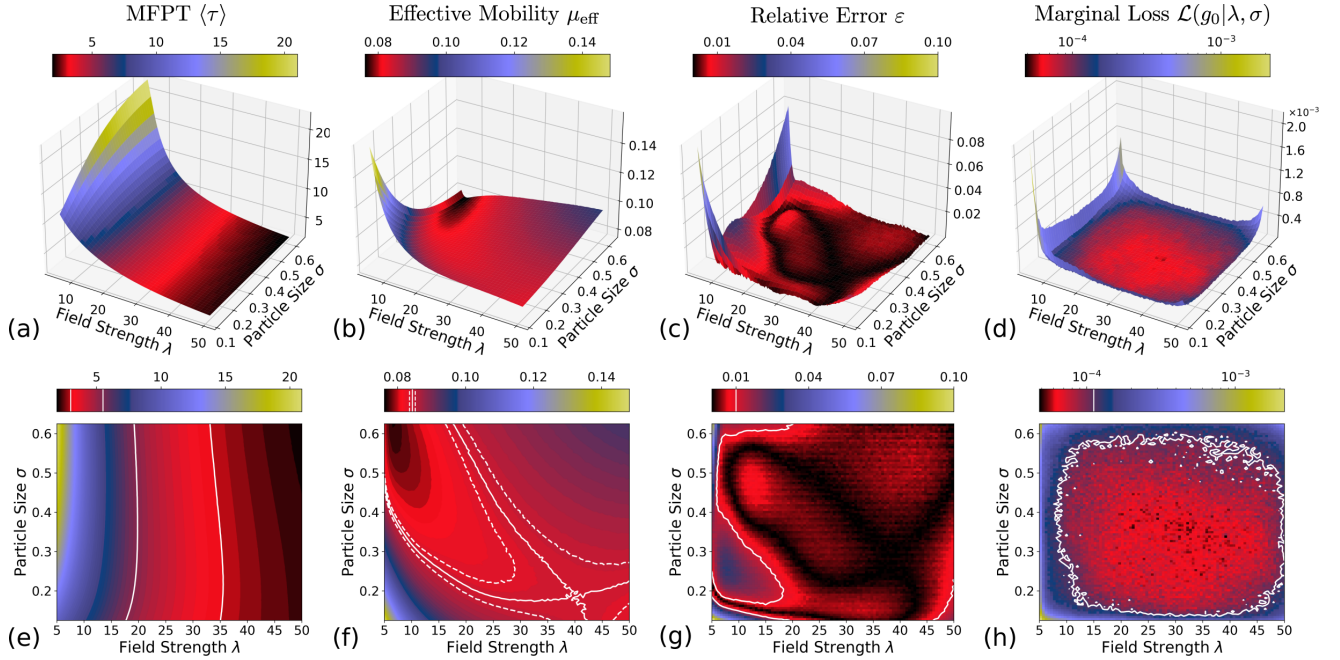


FIG. 6. Analysis of the $g_0(x, y; \lambda, \sigma)$ solution obtained using the NNM parametrized by both field strength λ and particle size σ . (a), (e) Mean first passage time $\langle \tau \rangle$ with white contours to show nonmonotonic sorting behavior. (b), (f) Effective mobility μ_{eff} with white contours to show saddle point. (c), (g) Relative error ε of the MFPT with white contour to denote 1% error threshold. (d), (h) Marginal loss $\mathcal{L}(g_0|\lambda, \sigma)$ with white contour to denote testing loss \mathcal{L} .

replotted in Fig. 6 as continuous functions of (λ, σ) . The first row [Figs. 6(a)–6(d)] shows three-dimensional plots of the metrics over parameter space, whereas the second row shows two-dimensional contour maps [Figs. 6(e) and 6(f)] and color maps [Figs. 6(g) and 6(h)] of the same metrics. As anticipated, the relative error ε [Figs. 6(c) and 6(g)] and marginal loss $\mathcal{L}(g_0|\lambda, \sigma)$ [Figs. 6(d) and 6(h)] are only large near the boundaries of the parameter training space. Indeed, ε in Fig. 6(g) is below the 1% threshold (indicated by the solid white line) throughout the majority of the parameter space, confirming the suspicion that the line plots in Fig. 5 provide a biased view of the $g_0(x, y; \lambda, \sigma)$ solution.

The boundary effect is particularly clear in $\mathcal{L}(g_0|\lambda, \sigma)$ in Fig. 6(d), which features a prominent convex shape. Here, $\mathcal{L}(g_0|\lambda, \sigma)$ is consistently higher along all the edges of the training parameter space and decreases monotonically and rapidly away from the boundary. In particular, $\mathcal{L}(g_0|\lambda, \sigma)$ is exceptionally large at the corners of the training space.

Note that the decay of marginal loss away from the boundaries of the parameter space is actually substantially sharper than it appears visually in Figs. 6(d) and 6(h). The color scales for Figs. 6(d) and 6(h) are logarithmic and the color map is not perceptually uniform: it exhibits far more variation in color and contrast near the lower end of the scale. These plotting choices make the subtle structure of the marginal loss more apparent, but give it the biased appearance of a gradual variation throughout the domain. In actuality, when plotted with a linear color scale and a perceptually uniform color map, the marginal loss appears essentially flat through most of the domain.

As expected, the relative error ε [Figs. 6(c) and 6(g)] is closely tied to the marginal loss $\mathcal{L}(g_0|\lambda, \sigma)$. Relative

error is uniformly low in the interior of the parameter space (roughly $(\lambda, \sigma) \in [15, 45] \times [0.2, 0.6]$), corresponding to the flat interior of $\mathcal{L}(g_0|\lambda, \sigma)$. Additionally, near the two corners at $\lambda = 5$ where $\mathcal{L}(g_0|\lambda, \sigma)$ is largest, ε also attains its highest values, approaching 10%. There is also a small peak in ε at the $(\lambda, \sigma) = (50, 0.125)$ corner, corresponding to an equally small peak in $\mathcal{L}(g_0|\lambda, \sigma)$ at the same corner. Surprisingly, although $\mathcal{L}(g_0|\lambda, \sigma)$ exhibits a clear peak at the $(\lambda, \sigma) = (50, 0.625)$ corner, ε does not. Therefore, the marginal loss $\mathcal{L}(g_0|\lambda, \sigma)$ once again appears to act as a conservative *a posteriori* estimator of relative error ε : high relative error occurs near regions of high marginal loss, although high marginal loss does not always imply high relative error.

As noted above, the performance of the $g_0(x, y; \lambda, \sigma)$ solution deteriorates even more significantly at the corners of the parameter training space than on its edges. This is more complicated than the boundary effect discussed for the solutions parametrized by just λ or σ , and can be accounted for by extending the postulated mechanism from Secs. IV B 2 and IV B 3. There, it was argued that the deterioration in performance arises because the stochastic sampling used during training under-represents boundary points: whereas the neighborhoods of interior points are thoroughly sampled on all sides, this is not true for boundary points. In the two-dimensional parameter training space considered here, the corners and the edges of the boundary are under-represented to different extents by the stochastic sampling process. Whereas parameter values on the edges of the domain only have 50% as many neighboring points inside the training space as interior points, parameter values on the corners have only 25% as many. This tentatively explains why performance

is so much worse at corners of the parameter training space than it is on the edges.

If this mechanism extends to higher dimensional parameter spaces, it may eventually prove to be a dominant source of error: for instance, the corners of an n -dimensional hypercube have only $1/2^n$ as many neighbors inside the training space as interior points, and the number of boundary segments (corners, edges, faces, ...) grows rapidly with n . In fact, the fraction of a parameter space lying within a given distance of its boundary also increases with dimensionality. Altogether, these observations suggest the need for further investigation into this boundary effect, its possible connection to Monte Carlo sampling of the loss during training, and methods (such as low-discrepancy sampling methods [66,67]) for resolving the problem.

In contrast to the line plots in Fig. 5, the plots in Fig. 6 highlight the richness of information available through the $g_0(x, y; \lambda, \sigma)$ solution compared with the solutions parametrized only by λ (Sec. IV B 2), σ (Sec. IV B 3), or neither (Sec. IV B 1). For instance, although the NNM solutions parametrized by λ or σ (Secs. IV B 2 and IV B 3) suggested a nonmonotonic dependence of $\langle \tau \rangle$ and μ_{eff} with respect to σ for certain values of λ , they did not provide sufficient information to estimate the exact range of λ over which this behavior persists. Just as the NNM solutions parametrized by λ or σ (Secs. IV B 2 and IV B 3) are more helpful than the fixed parameter solutions (Sec. IV B 1) in localizing one-dimensional critical points, so is the NNM solution parametrized by both λ and σ more useful for delineating the nonmonotonic regions of parameter space.

The range of nonmonotonic behavior can be estimated visually from μ_{eff} in Figs. 6(b) and 6(f). Using a vertical line test, it is easy to see that nonmonotonic dependence of μ_{eff} on σ is present at voltages as low as $\lambda \approx 10$. In fact, μ_{eff} is doubly nonmonotonic with respect to both λ and σ in the large- σ , low- λ range [black region in Figs. 6(b) and 6(f)]. Although the same trends were suspected from the solutions discussed in Secs. IV B 2 and IV B 3, $g_0(x, y; \lambda, \sigma)$ resolves the features more completely.

Despite the usefulness of $g_0(x, y; \lambda, \sigma)$ for resolving critical points, the solution predicts a false saddle point in μ_{eff} at $(\lambda, \sigma) \approx (40, 0.2)$ [highlighted by the white solid line in Fig. 6(f)]. Additional FEM results (not shown) confirm that there is no saddle point anywhere in the parameter space under consideration. This error can be attributed to the fact that the true μ_{eff} changes extremely little in the high- λ , low- σ region of the domain. For illustration, the two dotted white lines in Fig. 6(f) indicate contours for μ_{eff} values 1% greater and smaller, respectively, than the value of μ_{eff} on the solid white line passing through the saddle point. Despite this range corresponding to a very small fraction of the total variation of μ_{eff} over the domain, the area between the dotted white lines account for roughly 25% of the total parameter training space, demonstrating that μ_{eff} is extremely flat throughout this entire region.

Although the presence of a false saddle point is a noticeable qualitative error, it corresponds to a very small quantitative error in the key output metrics $\langle \tau \rangle$ and μ_{eff} . In fact, the visual appearance of the saddle point in Fig. 6(f) is intentionally accentuated by the choice of color map, as discussed for

the marginal loss above. It is quite feasible that an error of such small magnitude could be resolved simply by increasing network capacity and/or training time.

Still, the question arises of whether and how the NNM can be used reliably in applications where these types of incorrect or ill-conditioned features may occur. The marginal loss $\mathcal{L}(g_0|\lambda, \sigma)$ provides one possible resolution to this concern. The region of increased $\mathcal{L}(g_0|\lambda, \sigma)$ near $(\lambda, \sigma) = (50, 0.125)$ in Fig. 6(h) coincides fairly closely with the right half of the saddle point in Fig. 6(f). Thus, $\mathcal{L}(g_0|\lambda, \sigma)$ correctly reflects that the solution is less reliable in this region, drawing into question the validity of the predicted saddle point.

Future work should elaborate on what quantitative predictions of solution quality can be based on the marginal loss, along the lines of the investigations of Magill *et al.* [57]. In the interim, we propose using the total loss [as indicated in Fig. 6(h) by the solid white line] as an approximate threshold between regions of relatively high and low expected accuracy. In fact, the marginal loss $\mathcal{L}(g_0|\lambda, \sigma)$ as defined here is likely a suboptimal tool for the detection of false critical points in parameter space because it does not directly measure gradient information with respect to (λ, σ) . Rather, it is only indirectly sensitive to the error in the shape of $\langle \tau \rangle$ and μ_{eff} insofar as it emerges from errors in the shape of $g_0(x, y; \lambda, \sigma)$. For applications in which the localization of ill-conditioned critical points is of interest, modified loss functions that incorporate the derivatives of the target PDE with respect to λ and σ (e.g., like those explored by Avrutskiy [59]) might be more relevant error estimators. This notion illustrates the potential benefits of customizing the NNM for specific PDEs and research questions, just as flux- or energy-conserving numerical methods are preferred for applications where those features are particularly important.

In summary, the results in this section demonstrate that the NNM can produce a robust approximation to the $g_0(x, y; \lambda, \sigma)$ solution. Here, $g_0(x, y; \lambda, \sigma)$ enables higher-dimensional visualization of $\langle \tau \rangle$ and μ_{eff} over λ and σ , resolving features in parameter space more accurately and completely than the solutions parametrized by only λ or σ . Furthermore, $g_0(x, y; \lambda, \sigma)$ accurately predicts the magnitude of $\langle \tau \rangle$ and μ_{eff} to within the 1% error threshold simultaneously over the majority of the parameter training space. Although $g_0(x, y; \lambda, \sigma)$ exhibits some regions of high relative error, $\mathcal{L}(g_0|\lambda, \sigma)$ once again provides a robust *a posteriori* estimator of the solution's reliability throughout the parameter space.

V. CONCLUSIONS

This work investigated the use of the neural network method to solve a parametrized time-integrated Smoluchowski equation describing nanoparticle passage through the slit-well microfluidic device. The g_0 solutions were solved for a variety of fixed choices of field strength λ and particle size σ using both the NNM and a standard FEM implementation. Additionally, the NNM was used to solve the equation directly as a function of λ and/or σ . Mean first passage time $\langle \tau \rangle$ and effective mobility μ_{eff} were studied as the primary output metrics of interest, with relative error ε and marginal loss $\mathcal{L}(g_0|\lambda, \sigma)$ used to characterize solution performance.

The qualitative examinations of g_0 in Sec. IV A revealed a wide variety of functional behavior over the region of (λ, σ) parameter space studied here. The four primary regimes underlying nanoparticle sorting in the slit-well (i.e., low and high fields for small and large particles) correspond to four significantly different g_0 solution types, each reflecting different interplays of drift, diffusion, and geometry. This highlights the challenging nature of the parametrized PDE problem studied in this work. Additionally, this analysis suggested that the g_0 solutions themselves may encode interesting and useful qualitative information about biophysical processes. Future work should examine how information encoded in g_0 may be complementary to qualitative information derived from stochastic particle trajectories.

Of course, qualitative insights aside, the most salient feature of g_0 is that it integrates to yield the mean first passage time $\langle \tau \rangle$. As noted, although $\langle \tau \rangle$ is a quantity of widespread interest in all first passage problems and is relevant to many MNFD design problems, it appears that numerical solutions of g_0 have rarely been leveraged for such applications. The results of this paper support that g_0 may be an undervalued tool in computational biophysics.

Although g_0 can be computed using many methods, such as FEM or particle simulations, this work focused on resolving g_0 using the NNM. When applied to fixed choices of problem parameters, the NNM consistently estimated $\langle \tau \rangle$ with errors below 1%. In particular, the NNM values were at least as accurate as typical particle simulations, which are the most common tool for studying first passage problems in biophysics. However, a proper comparison of runtime was not conducted in this work, and should be a major focus of future investigations.

The main appeal of the NNM is the unique ease with which it can be applied to parametrized g_0 problems. Via integration of g_0 , these solutions yield a direct mapping from key problem inputs (e.g., λ, σ) to key problem outputs (e.g., $\langle \tau \rangle, \mu_{\text{eff}}$). This is particularly appealing for the application of MNFD research, where essential phenomena often depend nontrivially on the coupling of many system parameters. The results in the current work demonstrate that the NNM can learn accurate approximations of g_0 parametrized by λ, σ , or both, all using a modest network size and even without careful hyperparameter optimization. Whereas classical ROM techniques typically require special considerations to handle geometry-modifying parameters like σ , the NNM was found to resolve $g_0(x, y; \sigma)$ just as easily as $g_0(x, y; \lambda)$. As discussed, parametrized solutions can be quite useful in characterizing entire regions of parameter space.

Although the NNM is expected to perform well on highly parametrized PDEs, the careful error analysis presented in the current study revealed several points of caution for future efforts in this direction. First, all parametrized solutions studied here exhibited a deterioration in accuracy near the boundaries of their parameter training space. Nonetheless, the predicted values of $\langle \tau \rangle$ were still mostly within the 1% margin of error. Moreover, the marginal loss functional $\mathcal{L}(g_0|\lambda, \sigma)$ proposed here was found to act as a conservative *a posteriori* estimator of the solution accuracy throughout parameter space.

The second point of caution that must be considered when applying the NNM to parametrized PDEs concerns the

interpretation of key features, such as critical points, that are identified using these solutions. For instance, in Sec. IV B 4, the NNM solution exhibited an erroneous saddle point in a flat region of μ_{eff} , which was an artifact that arose due to the ill-conditioning of the gradients of $\mu_{\text{eff}}(\lambda, \sigma)$. In fact, plots of ε showed no indication of errors in this region, as the mistake only manifested in the curvature of the mapping. However, once again the marginal loss $\mathcal{L}(g_0|\lambda, \sigma)$ did indicate that the NNM solution lost fidelity in this region of parameter.

In summary, the parametrized NNM solutions were generally accurate far from the training boundaries, and $\mathcal{L}(g_0|\lambda, \sigma)$ provided robust regions of confidence. Altogether, these results highlight the specific appeal of the NNM as a method for studying parametrized first passage problems via the time-integrated Smoluchowski model. We hope this work prompts further investigation into the use of g_0 with or without the NNM, and into the relationship of $\langle \tau \rangle$ and μ_{eff} to more standard MNFD metrics. Regarding the application of the NNM to such problems, future work should address technical challenges such as singularities posed by sharp corners, training difficulties for highly skewed geometries, and achieving competitive runtime.

ACKNOWLEDGMENTS

The authors gratefully acknowledge funding from Mitacs under the Accelerate Entrepreneur program (Ref. IT21168) and from Smart Computing for Innovation (SOSCIP Ref. 3-076). H.W.d.H. also acknowledges funding from the Natural Sciences and Engineering Research Council (NSERC) in the form of Discovery Grant No. 2020-07145. M.M. also acknowledges funding from the Ontario Graduate Scholarship (OGS) Program and the Vector Institute Postgraduate Affiliate Program.

APPENDIX A: COMPUTING MEAN FIRST PASSAGE TIMES WITH PARTICLE SIMULATIONS

This section contains a description of standard Brownian dynamics (BD) simulations used to measure the mean first passage times of nanoparticles traversing the slit-well microfluidic device (Sec. II). The BD simulations were initialized with $N = 100\,000$ noninteracting particles placed according to the distribution ρ_0 [Eq. (7)]. The position of the i th particle \vec{x}_i was updated according to the discretized BD equation

$$\frac{\Delta \vec{x}_i}{\Delta t} = \sqrt{\frac{2D}{\Delta t}} \vec{R}(t) + \frac{q\lambda}{\gamma} \vec{E}_0 + \frac{1}{\gamma} \vec{F}_{\text{WCA}}. \quad (\text{A1})$$

In Eq. (A1), the particle properties are the diffusion coefficient D , the friction coefficient γ , and the particle charge q . As noted in Sec. II, both q and γ were set equal to the particle diameter σ , to capture free-draining behavior. The diffusion coefficient D was set to $1/\sigma$ and the time step was set $\Delta t = 10^{-5}$. The term $\vec{R}(t)$ in Eq. (A1) is a random driving force representing the thermal motion of an implicit solvent which was sampled from a uniform distribution of mean 0 and variance 1.

Rather than representing the interactions between particles and walls as perfectly rigid, the walls were implemented using

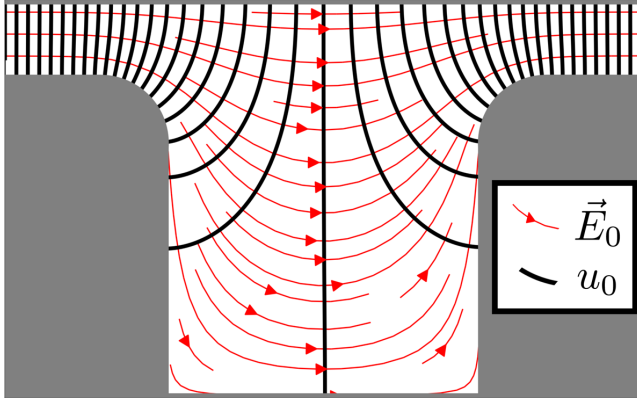


FIG. 7. Contour plots of the baseline electric potential u_0 (black) and field \vec{E}_0 (red) computed by the NNM.

a short-range repulsive shifted WCA force

$$\vec{F}_{\text{WCA}} = -\nabla U_{\text{WCA}}, \quad (\text{A2})$$

with

$$U_{\text{WCA}}(r_i) = \begin{cases} 4\epsilon \left[\left(\frac{\sigma_0}{r_i} \right)^{12} - \left(\frac{\sigma_0}{r_i} \right)^6 \right] + \epsilon, & r_i < r_{\text{cut}} \\ 0, & r_i \geq r_{\text{cut}} \end{cases} \quad (\text{A3})$$

where r_i is the minimum distance from particle i to the nearest reflective wall minus a distance $r_{\text{shift}} = 0.5(\sigma - \sigma_0)$. Here r_{shift} corresponds to the radius of the hard core of the particle, whereas $\sigma_0 = 0.125$ is the length over which the surface of the particle is partially compressible. The potential is zero beyond a cutoff distance $r_{\text{cut}} = 2^{1/6}\sigma_0$, so that if the center of the particle is farther than a distance $r_{\text{shift}} + r_{\text{cut}}$ from the wall there is no interaction. The energy scale of the repulsive interaction was set to $\epsilon = 0.125 = \sigma_0$.

Although this type of model is commonly used for particle-wall interactions, due to its improved numerical stability relative to perfectly rigid interactions, it introduces a small difference between the underlying physics of the BD simulations and the PDE models being solved in this work. For this reason, the MFPTs determined using particle simulations should not be expected to agree exactly with those obtained using the NNM and FEM methods, even in the limit of small Δt and large N . Nonetheless, as our results corroborate, the effect of this difference between the models is small.

The term \vec{E}_0 in Eq. (A1) corresponds to the baseline electric field in the slit-well domain (denoted by red in Fig. 7). This was solved for a voltage drop of two units from the leftmost to rightmost boundaries, as in Magill *et al.* [57]. The net electric-field strength was set by the parameter λ . \vec{E}_0 used here was the same one described in Sec. III B. As shown in Magill *et al.* [57], particle simulations conducted using an electric field solved with the NNM are nearly statistically indistinguishable from those conducted using a field solved with the FEM, so long as the NNM electric field exhibits a sufficiently small loss. The purpose of the present study is not to replicate this result, but to explore the computational advantages of the NNM over other techniques in parametrized problems. Thus, the particle simulations are conducted using

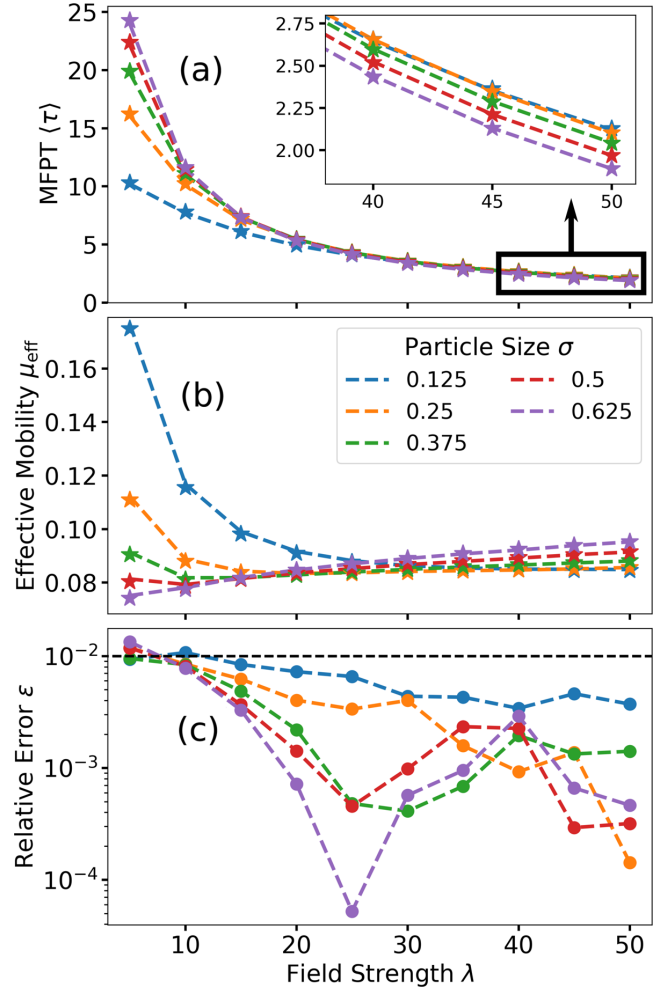


FIG. 8. Passage time properties of particles escaping the slit-well model computed using Brownian dynamics simulations. Star markers denote values obtained via FEM. (a) Mean first passage time $\langle \tau \rangle$. (b) Effective mobility μ_{eff} . (c) Relative error of $\langle \tau \rangle$ computed against the ground truth FEM solution.

the FEM electric field, which is taken as the reference ground truth.

Parallel to the analysis conducted in Sec. IV B, the mean first passage time $\langle \tau \rangle$ and effective mobility μ_{eff} are computed using the BD simulations for various choices of field strength λ and particle size σ . These values are plotted with dashed lines in Fig. 8(a) and 8(b) with star markers to denote $\langle \tau \rangle$ and μ_{eff} values obtained by FEM. Note that $\langle \tau \rangle$ and μ_{eff} are only solved for the same discrete choices of λ and σ that are also computed using FEM.

In addition, the relative error ϵ is computed using Eq. (23) where $\langle \tau \rangle$ and $\langle \tau \rangle_{\text{FEM}}$ are the MFPTs computed by BD and FEM, respectively. The values are plotted in Fig. 8(c) with circular markers denoting the parameter choices where the relative error was computed. All of the relative errors in Fig. 8(c) fall below 2%, with majority of the values being within 1% error. This establishes a 1% error baseline against the ground truth MFPT values computed by FEM, for which to benchmark the performance of the NNM.

TABLE I. Comparison of computational time (in minutes) of the NNM, BD, and FEM methods used to compute the MFPT at fixed parameter values.

Parameters (λ, σ)	Runtime (minutes)		
	NNM	BD	FEM
(5.0, 0.125)	136.95	5.02	2.25
(50.0, 0.125)	138.37	1.03	2.15
(5.0, 0.625)	126.57	12.80	2.18
(50.0, 0.625)	133.63	1.12	2.13

APPENDIX B: CONTOUR PLOTS OF ELECTRIC POTENTIAL AND FIELD

The baseline electric field \vec{E}_0 used to drive particle motion in the slit-well device [Eq. (6)] was computed using the NNM, as described in Magill *et al.* [57]. That is, the baseline electric potential u_0 was solved using Laplace's equation over a voltage drop of two units from the left slit wall to the right slit wall. The electric field was then computed using the relation $\vec{E}_0 = \nabla u_0$. The red and black contour lines in Fig. 7 correspond to the electric field \vec{E}_0 and electric potential u_0 , respectively, inside the slit-well MNFD.

APPENDIX C: RUNTIME COMPARISON

The MFPT and effective mobility of nanoparticles traversing the slit-well MNFD were obtained using BD simulations, the FEM, and the NNM. Table I shows the runtime, in minutes, of each method used to solve the MFPT at fixed choices parameter values. Four choices of the parameters are included,

TABLE II. Comparison of computational time (in days) of the various methods used to compute the MFPT over ranges of parameter space.

Method	Mean Runtime (days)
NNM parameterized by λ	6.33
NNM parameterized by σ	7.79
NNM parameterized by (λ, σ)	7.66
High-resolution FEM sampling	12.18

illustrating that runtimes were fairly independent of parameters for NNM and FEM but depended strongly on parameters for BD. Table II shows the runtime, in days, of each method used to solve the MFPT over large regions of parameter space. As implemented, the various parametrized NNM methods all have runtimes comparable to one another. Moreover, the total runtime of the high-resolution FEM sampling exceeds the mean runtime for the parametrized NNM methods. However, this runtime obviously depends on the number of points sampled. Here, 8099 parameter combinations were utilized in order to produce high-resolution maps of error over parameter space.

Optimizing runtime was not a goal of the current work. The implementations of each of the algorithms studied here (NNM, BD, FEM) can undoubtedly be improved upon to substantially decrease the runtimes from those reported in Tables I and II. Moreover, judicious use of parallelization across GPUs and/or CPUs, as applicable, could provide further improvements to each of the methods. Thus, the runtimes included here are provided for reference only, and a more careful comparison is left to future work.

- [1] S. L. Levy and H. G. Craighead, DNA manipulation, sorting, and mapping in nanofluidic systems, *Chem. Soc. Rev.* **39**, 1133 (2010).
- [2] K. D. Dorfman, DNA electrophoresis in microfabricated devices, *Rev. Mod. Phys.* **82**, 2903 (2010).
- [3] P. Abgrall and N. T. Nguyen, Nanofluidic devices and their applications, *Anal. Chem.* **80**, 2326 (2008).
- [4] H. Gardeniers and A. Van Den Berg, Micro-and nanofluidic devices for environmental and biomedical applications, *Int. J. Environ. Anal. Chem.* **84**, 809 (2004).
- [5] R. Mulero, A. S. Prabhu, K. J. Freedman, and M. J. Kim, Nanopore-based devices for bioanalytical applications, *J. Assoc. Lab. Anim.* **15**, 243 (2010).
- [6] J. Han and H. G. Craighead, Entropic trapping and sieving of long DNA molecules in a nanofluidic channel, *J. Vac. Sci. Technol., A* **17**, 2142 (1999).
- [7] K.-L. Cheng, Y.-J. Sheng, S. Jiang, and H.-K. Tsao, Electrophoretic size separation of particles in a periodically constricted microchannel, *J. Chem. Phys.* **128**, 101101 (2008).
- [8] H. Wang, H. W. de Haan, and G. W. Slater, Electrophoretic ratcheting of spherical particles in well/channel microfluidic devices: Making particles move against the net field, *Electrophoresis* **41**, 621 (2020).
- [9] J. Fu, P. Mao, and J. Han, Nanofilter array chip for fast gel-free biomolecule separation, *Appl. Phys. Lett.* **87**, 263902 (2005).
- [10] J. Fu, J. Yoo, and J. Han, Molecular Sieving in Periodic Free-Energy Landscapes Created by Patterned Nanofilter Arrays, *Phys. Rev. Lett.* **97**, 018103 (2006).
- [11] M. Langecker, D. Pedone, F. C. Simmel, and U. Rant, Electrophoretic time-of-flight measurements of single DNA molecules with two stacked nanopores, *Nano Lett.* **11**, 5002 (2011).
- [12] J. A. Berkenbrock, R. Grecco-Machado, and S. Achenbach, Microfluidic devices for the detection of viruses: Aspects of emergency fabrication during the covid-19 pandemic and other outbreaks, *Proc. R. Soc. London, Ser. A* **476**, 20200398 (2020).
- [13] S. J. Shepherd, C. C. Warzecha, S. Yadavali, R. El-Mayta, M.-G. Alameh, L. Wang, D. Weissman, J. M. Wilson, D. Issadore, and M. J. Mitchell, Scalable mRNA and siRNA lipid nanoparticle production using a parallelized microfluidic device, *Nano Lett.* **21**, 5671 (2021).
- [14] T. Chou and M. R. D'Orsogna, First passage problems in biology, in *First-passage Phenomena and their Applications* (World Scientific, Singapore, 2014), pp. 306–345.
- [15] M. H. Lam, K. Briggs, K. Kastitis, M. Magill, G. R. Madejski, J. L. McGrath, H. W. de Haan, and V. Tabard-Cossa, Entropic trapping of DNA with a nanofiltered nanopore, *ACS Appl. Nano Mater.* **2**, 4773 (2019).
- [16] K. Briggs, G. Madejski, M. Magill, K. Kastitis, H. W. de Haan, J. L. McGrath, and V. Tabard-Cossa, DNA translocations

- through nanopores under nanoscale preconfinement, *Nano Lett.* **18**, 660 (2018).
- [17] H. W. de Haan and G. W. Slater, Mapping the variation of the translocation α scaling exponent with nanopore width, *Phys. Rev. E* **81**, 051802 (2010).
 - [18] M. Magill, Ed Waller, and H. W. de Haan, A sequential nanopore-channel device for polymer separation, *J. Chem. Phys.* **149**, 174903 (2018).
 - [19] S. Redner, *A Guide to First-Passage Processes* (Cambridge University Press, Cambridge, 2001).
 - [20] V. Kurella, J. C. Tzou, D. Coombs, and M. J. Ward, Asymptotic analysis of first passage time problems inspired by ecology, *Bull. Math. Biol.* **77**, 83 (2015).
 - [21] S. Iyer-Biswas and A. Zilman, First passage processes in cellular biology, *Advances in Chemical Physics* **160**, 261 (2016).
 - [22] Z. Ma, A. W. Krings, and R. C. Millar, Introduction of first passage time (FPT) analysis for software reliability and network security, in *Proceedings of the 5th Annual Workshop on Cyber Security and Information Intelligence Research: Cyber Security and Information Intelligence Challenges and Strategies* (Association for Computing Machinery, New York, 2009), pp. 1–6.
 - [23] E. D. Filho and M. Araújo, Synaptic transmission and Fokker-Planck equation, in *ICORES* (SciTePress, Setúbal, Portugal, 2012), pp. 59–63.
 - [24] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, ImageNet: A large-scale hierarchical image database, In *2009 IEEE Conference on Computer Vision and Pattern Recognition* (Institute of Electrical and Electronics Engineers, New Jersey, 2009), pp. 248–255.
 - [25] Y. LeCun, Y. Bengio, and G. Hinton, Deep learning, *Nature (London)* **521**, 436 (2015).
 - [26] R. Collobert, J. Weston, L. Bottou, M. Karlen, K. Kavukcuoglu, and P. Kuksa, Natural language processing (almost) from scratch, *J. Mach. Learn. Res.* **12**, 2493 (2011).
 - [27] Weinan E, J. Han, and A. Jentzen, Deep learning-based numerical methods for high-dimensional parabolic partial differential equations and backward stochastic differential equations, *Commun. Math. Stat.* **5**, 349 (2017).
 - [28] J. Sirignano and K. Spiliopoulos, DGM: A deep learning algorithm for solving partial differential equations, *J. Comput. Phys.* **375**, 1339 (2018).
 - [29] V. R. Royo and C. Tomlin, Recursive regression with neural networks: Approximating the HJI PDE solution, [arXiv:1611.02739](https://arxiv.org/abs/1611.02739).
 - [30] V. I. Avrutskiy, Neural networks catching up with finite differences in solving partial differential equations in higher dimensions, *Neural Computing and Applications* (Springer, New York, 2020), pp. 1–16.
 - [31] J. Han, A. Jentzen, and E. Weinan, Solving high-dimensional partial differential equations using deep learning, *Proc. Natl. Acad. Sci. USA* **115**, 8505 (2018).
 - [32] C. Huré, H. Pham, and X. Warin, Deep backward schemes for high-dimensional nonlinear PDEs, *Mathematics of Computation* (American Mathematical Society, Providence, 2020).
 - [33] S. Karumuri, R. Tripathy, I. Bilonis, and J. Panchal, Simulator-free solution of high-dimensional stochastic elliptic partial differential equations using deep neural networks, *J. Comput. Phys.* **404**, 109120 (2020).
 - [34] H. Mutuk, Neural network study of hidden-charm pentaquark resonances, *Chin. Phys. C* **43**, 093103 (2019).
 - [35] M. A. Nabian and H. Meidani, A deep learning solution approach for high-dimensional random differential equations, *Probab. Eng. Mech.* **57**, 14 (2019).
 - [36] D. Zhang, L. Guo, and G. Em Karniadakis, Learning in modal space: Solving time-dependent stochastic PDEs using physics-informed neural networks, *SIAM J. Sci. Comput.* **42**, A639 (2020).
 - [37] C. Beck, Weinan E, and A. Jentzen, Machine learning approximation algorithms for high-dimensional fully nonlinear partial differential equations and second-order backward stochastic differential equations, *J. Nonlinear Sci.* **29**, 1563 (2019).
 - [38] Q. Wei, Y. Jiang, and J. Z. Y. Chen, Machine-learning solver for modified diffusion equations, *Phys. Rev. E* **98**, 053304 (2018).
 - [39] P. Grohs, F. Hornung, A. Jentzen, and P. von Wurstemberger, A proof that artificial neural networks overcome the curse of dimensionality in the numerical approximation of Black-Scholes partial differential equations, [arXiv:1809.02362](https://arxiv.org/abs/1809.02362).
 - [40] A. Jentzen, D. Salimova, and T. Welti, A proof that deep artificial neural networks overcome the curse of dimensionality in the numerical approximation of Kolmogorov partial differential equations with constant diffusion and nonlinear drift coefficients, *Commun. Math. Sci.* **19**, 1167 (2021).
 - [41] M. Hutzenthaler, A. Jentzen, T. Kruse, and T. A. Nguyen, A proof that rectified deep neural networks overcome the curse of dimensionality in the numerical approximation of semilinear heat equations, *SAM Research Report 2019* (ETH, Zürich, Switzerland, 2019).
 - [42] O. Hennigh, S. Narasimhan, M. Amin Nabian, A. Subramaniam, K. Tangsali, Z. Fang, M. Rietmann, W. Byeon, and S. Choudhry, NVIDIA SimNet: An AI-accelerated multi-physics simulation framework, in *International Conference on Computational Science* (Springer, 2021), pp. 447–461.
 - [43] J. Berner, M. Dablander, and P. Grohs, Numerically solving parametric families of high-dimensional Kolmogorov partial differential equations via deep learning, [arXiv:2011.04602](https://arxiv.org/abs/2011.04602).
 - [44] K. Glau and L. Wunderlich, The deep parametric PDE method: application to option pricing, [arXiv:2012.06211](https://arxiv.org/abs/2012.06211).
 - [45] Y. Khoo, J. Lu, and L. Ying, Solving parametric pde problems with artificial neural networks, *Eur. J. Appl. Math.* **32**, 421 (2021).
 - [46] M. Abid Bazaz, S. Janardhanan *et al.*, A review of parametric model order reduction techniques, in *2012 IEEE International Conference on Signal Processing, Computing and Control* (IEEE, 2012), pp. 1–6.
 - [47] K. Lu, K. Zhang, H. Zhang, X. Gu, Y. Jin, S. Zhao, C. Fu, and Y. Yang, A review of model order reduction methods for large-scale structure systems, *Shock and Vibration* (2021).
 - [48] N. R. Franco, A. Manzoni, and P. Zunino, A deep learning approach to reduced order modelling of parameter dependent partial differential equations, [arXiv:2103.06183](https://arxiv.org/abs/2103.06183).
 - [49] M. F. Kasim, D. Watson-Parris, L. Deaconu, S. Oliver, P. Hatfield, D. H. Froula, G. Gregori, M. Jarvis, S. Khatiwala, J. Korenaga *et al.*, Building high accuracy emulators for scientific simulations with deep neural architecture search, *Mach. Learn.: Sci. Technol.* **3**, 015013 (2022).
 - [50] H. Eivazi, H. Veisi, M. Hossein Naderi, and V. Esfahanian, Deep neural networks for nonlinear model order reduction of unsteady flows, *Phys. Fluids* **32**, 105104 (2020).

- [51] T. Daniel, F. Casenave, N. Akkari, and D. Ryckelynck, Model order reduction assisted by deep neural networks (rom-net), *Adv. Model. Simul. Eng. Sci.* **7**, 16 (2020).
- [52] S. Fresca *et al.*, A comprehensive deep learning-based approach to reduced order modeling of nonlinear time-dependent parametrized PDEs, *J. Sci. Comput.* **87**, 61 (2021).
- [53] V. De La Rubia, A. Lamecki, and M. Mrozowski, Geometry parametric model order reduction with randomly generated projection bases, in *2016 IEEE MTT-S International Conference on Numerical Electromagnetic and Multiphysics Modeling and Optimization* (IEEE, 2016), pp. 1–2.
- [54] P. Heydari and M. Pedram, Model reduction of variable-geometry interconnects using variational spectrally-weighted balanced truncation, in *IEEE/ACM International Conference on Computer Aided Design. ICCAD 2001. IEEE/ACM Digest of Technical Papers (Cat. No. 01CH37281)* (IEEE, 2001), pp. 586–591.
- [55] L. Daniel, O. C. Siong, L. S. Chay, K. H. Lee, and J. White, A multiparameter moment-matching model-reduction approach for generating geometrically parameterized interconnect performance models, *IEEE Trans. Comput. Aid. D.* **23**, 678 (2004).
- [56] H. Gao, L. Sun, and J.-X. Wang, Phygeonet: Physics-informed geometry-adaptive convolutional neural networks for solving parameterized steady-state pdes on irregular domain, *J. Comput. Phys.* **428**, 110079 (2021).
- [57] M. Magill, A. M. Nagel, and H. W. de Haan, Neural network solutions to differential equations in nonconvex domains: Solving the electric field in the slit-well microfluidic device, *Phys. Rev. Research* **2**, 033110 (2020).
- [58] A. Al-Aradi, A. Correia, D. Naiff, G. Jardim, and Y. Saporito, Solving nonlinear and high-dimensional partial differential equations via deep learning, [arXiv:1811.08782](https://arxiv.org/abs/1811.08782).
- [59] V. I. Avrutskiy, Enhancing function approximation abilities of neural networks by training derivatives, *IEEE Trans. Neural Netw. Learn. Syst.* **32**, 916 (2020).
- [60] Lu Lu, P. Jin, and G. Em Karniadakis, DeepOnet: Learning nonlinear operators for identifying differential equations based on the universal approximation theorem of operators, [arXiv:1910.03193](https://arxiv.org/abs/1910.03193).
- [61] M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G. S. Corrado, A. Davis, J. Dean, M. Devin, S. Ghemawat, I. Goodfellow, A. Harp, G. Irving, M. Isard, Y. Jia, R. Jozefowicz, L. Kaiser *et al.*, TensorFlow: Large-scale machine learning on heterogeneous systems, [arXiv:1603.04467](https://arxiv.org/abs/1603.04467) (2015).
- [62] X. Glorot and Y. Bengio, Understanding the difficulty of training deep feedforward neural networks, in *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics* (Proceedings of Machine Learning Research, Sardinia, Italy, 2010), pp. 249–256.
- [63] D. Kingma and J. Ba, Adam: A method for stochastic optimization, [arXiv:1412.6980](https://arxiv.org/abs/1412.6980).
- [64] M. Alnæs, J. Blechta, J. Hake, A. Johansson, B. Kehlet, A. Logg, C. Richardson, J. Ring, M. E. Rognes, and G. N. Wells, The FEniCS project version 1.5, *Archive of Numerical Software* **3**, 9 (2015).
- [65] I. V. Grigoriev, Y. A. Makhnovskii, A. M. Berezhkovskii, and V. Yu Zitserman, Kinetics of escape through a small hole, *J. Chem. Phys.* **116**, 9574 (2002).
- [66] J. Chen, R. Du, P. Li, and L. Lyu, Quasi-Monte Carlo sampling for machine-learning partial differential equations, [arXiv:1911.01612](https://arxiv.org/abs/1911.01612).
- [67] S. Mishra and T. K. Rusch, Enhancing accuracy of deep learning algorithms by training with low-discrepancy sequences, *SIAM J. Numer. Anal.* **59**, 1811 (2021).