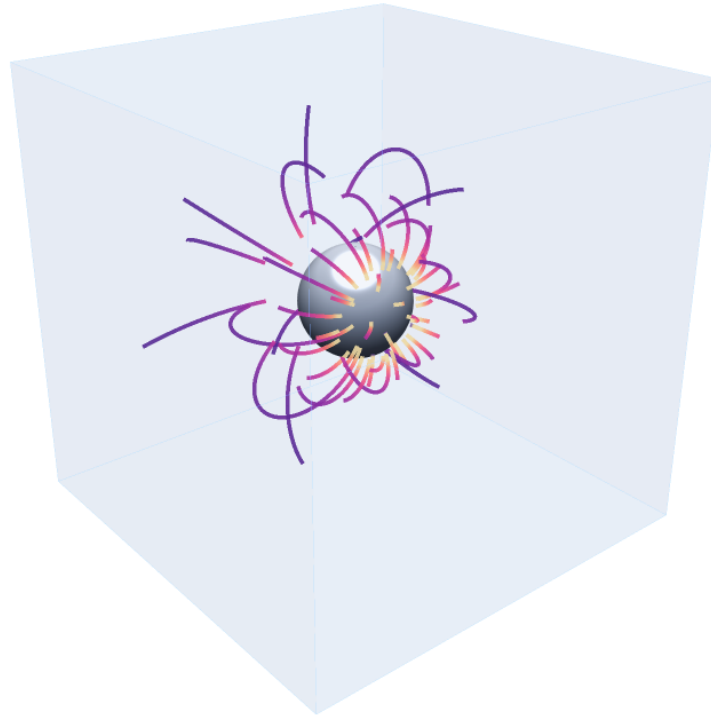




CHALMERS
UNIVERSITY OF TECHNOLOGY



Hydrodynamic Memory Kernel Extraction of a Microfluidic Particle with Machine Learning Based Kernel Estimation

Master's thesis in Complex Adaptive Systems, Msc
Master's thesis in Applied Mechanics, Msc

NILS STRÖMÖ
EESHAN GANLA

DEPARTMENT OF MECHANICS AND MARITIME SCIENCES

CHALMERS UNIVERSITY OF TECHNOLOGY
Gothenburg, Sweden 2026
www.chalmers.se

MASTER'S THESIS IN APPLIED MECHANICS MSc/COMPLEX
ADAPTIVE SYSTEMS MSc

Hydrodynamic Memory Kernel Extraction of a Microfluidic Particle with Machine Learning Based Kernel Estimation

NILS STRÖMÖ
EESHAN GANLA



CHALMERS
UNIVERSITY OF TECHNOLOGY

Department of Mechanics and Maritime Sciences
Division of Fluid Dynamics
CHALMERS UNIVERSITY OF TECHNOLOGY
Gothenburg, Sweden 2026

Hydrodynamic Memory Kernel Extraction of a Microfluidic Particle with Machine Learning Based Kernel Estimation
NILS STRÖMÖ
EESHAN GANLA

© NILS STRÖMÖ, EESHAN GANLA, 2026.

Supervisor: Anand Joseph Michael, Department of Mechanics and Maritime Sciences, Division of Fluid Dynamics
Examiner: Henrik Ström, Department of Mechanics and Maritime Sciences, Division of Fluid Dynamics

Master's Thesis 2026
Department of Mechanics and Maritime Sciences
Chalmers University of Technology
SE-412 96 Gothenburg
Sweden
Telephone +46 31 772 1000

Cover: Fluid visualization constructed in Python showing a particle moving through a fluid constrained by a cubic geometry.

Typeset in L^AT_EX
Gothenburg, Sweden 2026

Hydrodynamic Memory Kernel Extraction of a Microfluidic Particle with Machine Learning Based Kernel Estimation

NILS STRÖMÖ

EESHAN KEDAR

Department of Mechanics and Maritime Sciences

Division of Fluid Dynamics

Chalmers University of Technology

Abstract

The accurate simulation of Brownian particles in confined microfluidic environments requires resolving hydrodynamic memory effects, which introduce a nonlocal dependence between the particle’s velocity history and the instantaneous hydrodynamic force. This memory is captured by a convolution integral involving a memory kernel tensor extracted from direct numerical simulations (DNS). However, both DNS and the subsequent kernel optimization are computationally expensive, and the optimization can suffer from numerical artifacts that compromise physical realizability. In this work, we improved the existing kernel optimization framework by refining the simultaneous optimization of the full kernel tensor and implementing corrective measures that enhance accuracy for problematic cases where the original optimization produced erroneous results. Using these improvements, we perform DNS in two confined geometries at three confinement levels. Particle-wall distances are sampled to prioritize resolution of the variation of hydrodynamic memory near walls, edges, and corners.

A comprehensive library of memory kernels was generated across 84 unique positions for the cube and 28 positions for the channel. Finally, we developed a machine learning surrogate model that predicts the memory kernel directly from the particle’s position relative to confining walls, bypassing the need for expensive DNS. The surrogate model was trained on the DNS-generated kernel library and demonstrates accurate prediction of particle dynamics at a fraction of the computational cost. A comparison between different models and hyperparameter search led to Long-short term memory unit being the most accurate. This approach provides a pathway toward efficient, physically consistent simulations of particle motion in geometrically complex microfluidic systems.

Keywords: Direct Numerical Simulation, Brownian Motion, Machine Learning, Lagrangian Particle Tracking.

Preface

This report presents the outcome of our master's thesis project carried out at the Department of Mechanics and Maritime Sciences at Chalmers University of Technology during the spring of 2026.

Acknowledgements

The computations in this work were enabled by resources provided by the National Academic Infrastructure for Supercomputing in Sweden (NAISS), partially funded by the Swedish Research Council (Vetenskapsrådet) through a grant agreement.

I would like to express my gratitude to all those who have helped or supported me throughout my education and to those who aiding me in the writing of this thesis presented before you.

I would first like to thank Henrik Ström for lecturing an undergraduate course in continuum mechanics. His interest in Fluid Dynamics reflected back on me as I ended up enjoying the Fluid dynamics exercises much more interesting. It is the reason for picking up the topic of Hydrodynamics and Fluid dynamics for my thesis. Secondly in particular, I wish to thank Anand. For both inspiring me to pick this master thesis but also for you continued support regarding the methods, aims and help for the duration of the thesis.

At last, I would like to personally thank Eeshan and Zahran for aiding my studies throughout the period of this thesis. For the late nights, coffee breaks and the wonderful discussions.

Nils Strömö, Gothenburg, June 2026

I would like to sincerely thank Anand and Henrik for providing me with the opportunity to undertake this thesis. Their guidance, expertise, and continuous support were invaluable throughout the project. This work would not have been possible without their encouragement and insightful feedback. I am also grateful to have had Nils as a teammate, whose collaboration made the project both more productive and a great deal more enjoyable.

I would also like to thank Zahran Ghazy and our other colleagues at the Department of Fluid Dynamics, Chalmers, for creating a welcoming and enjoyable work environment. Their assistance, discussions, and camaraderie greatly enriched this experience.

Finally, I am deeply grateful to my family and friends for their unwavering support, encouragement, and motivation throughout my studies. Their belief in me has been a constant source of strength and inspiration.

Eeshan Ganla, Gothenburg, June 2026

List of Acronyms

Below is the list of acronyms that have been used throughout this thesis listed in alphabetical order:

CFD	Computational Fluid Dynamics
CLE	Classical Langevin Equation
DNS	Direct Numerical Simulation
DPD	Dissipative Particle Dynamics
FDT	Fluctuation–Dissipation Theorem
FSI	Fluid–Structure Interaction
GLE	Generalized Langevin Equation
GRU	Gated Recurrent Unit
IBM	Immersed Boundary Method
LPT	Lagrangian Particle Tracking
LSTM	Long Short-Term Memory
ML	Machine Learning
MLP	Multi-Layer Perceptron
MSD	Mean Squared Displacement
MSE	Mean Squared Error
NN	Neural Network
PINN	Physics-Informed Neural Network
PSD	Positive Semi-Definite
RANS	Reynolds-Averaged Navier–Stokes
VACF	Velocity Autocorrelation Function

Nomenclature

Below is the nomenclature of parameters, and variables that have been used throughout this thesis.

Latin symbols

$C(t)$	Velocity autocorrelation function
D	Diffusion coefficient
$\mathbf{F}_H$	Hydrodynamic force
$\tilde{\mathbf{F}}$	Random (fluctuating) force
h	Distance from the particle centre to the wall
K	Memory kernel tensor
K_{ij}	Memory kernel components
k_B	Boltzmann constant
L	Domain size
m_p	Particle mass
p	Pressure
Re_p	Particle Reynolds number
r_p	Particle radius
S	Spectral intensity of the Brownian force
T	Temperature
u_i	Fluid velocity
v	Particle velocity
V_{rms}	Root-mean-square thermal velocity
x_i	Spatial coordinate

Greek symbols

Γ	Noise strength of the fluctuating force
δ_{ij}	Kronecker delta
ζ	Stokes drag coefficient
λ_i	Loss function hyperparameter
μ_f	Dynamic viscosity of the fluid
ν	Kinematic viscosity of the fluid
ρ_f	Fluid density
ρ_p	Particle density
σ_{ij}	Viscous stress tensor
τ_p	Particle response time
τ_f	Viscous time scale
τ_w	Vorticity propagation time to the wall
τ_A	Time scale of finite-domain effects

Machine learning symbols

$\mathcal{L}_{\text{Total}}$	Total loss function
$\mathcal{L}_{\text{Residual}}$	Residual loss function
$\mathcal{L}_{\text{History}}$	Physics-inspired loss function

Contents

List of Acronyms	ix
Nomenclature	xi
List of Figures	xv
1 Introduction	1
1.1 Background	1
1.2 Previous Work and Literature Study	2
1.2.1 Langevin equation	2
1.2.2 Generalized Langevin equation	3
1.2.3 Hydrodynamic memory under confinement	4
1.2.4 Memory-kernel extraction from DNS	4
1.3 Aim	5
1.3.1 Optimisation of the DNS Framework	5
1.3.2 Confinement Effects Across Domain Sizes	5
1.3.3 Replacing Direct Numerical Simulation with Machine Learning	6
2 Theory	7
2.1 Generalized Langevin Equation and Memory Kernels	7
2.1.1 Classical Langevin Equation	7
2.1.1.1 Fluctuation-dissipation theorem	8
2.1.1.2 Velocity autocorrelation function	8
2.1.2 Relevant Time Scales	9
2.1.3 Generalized Langevin Equation	9
2.1.3.1 Memory Kernel Formulation	10
2.1.4 Characteristics of the memory kernels	10
2.2 Immersed Boundary Method	11
2.2.1 Governing Equations	12
2.2.2 Coupling Strategies for Immersed Boundaries	12
2.2.3 The Mirroring Immersed Boundary Method	13
2.2.4 IBOFlow Solver	13
2.3 Model Architecture	14
2.3.1 Long-short term memory	14
2.4 Loss functions	14
2.4.1 Logarithmic loss function	14
2.4.2 Physics Informed Neural Networks	14

3	Methodology	17
3.1	Generating Memory Kernels	17
3.1.1	Direct Numerical Simulation Framework	17
3.1.1.1	Computational Domain and Mesh	18
3.1.1.2	Boundary Conditions	18
3.1.1.3	Fluid and Particle Properties	18
3.1.1.4	Excitation Sequence and Temporal Discretisation . .	19
3.1.1.5	Numerical Implementation	20
3.1.1.6	Simulation Workflow	20
3.2	Memory Kernel Extraction from DNS	21
3.2.1	Optimisation Framework	21
3.2.2	Sampling of Particle Positions	23
3.2.3	Excitation Sequence	23
3.2.3.1	Numerical Instability for the [1,1,1] Forcing Direction	23
3.2.4	Data Processing and Kernel Library	24
3.3	Predicting Memory Kernels through Machine Learning	24
3.3.1	Data structure and processing	25
3.3.2	Network parameters and hyperparameters	25
3.3.3	Long-short term memory	25
3.3.4	Continuous-Time Decoder (MLP)	26
4	Results	31
4.1	Correction of Kernel Optimisation Instability	31
4.2	Hindrance Effects	33
4.2.1	Characteristics of the memory kernels	33
4.2.1.1	General Shape and Trends	33
4.2.2	Approach to a Wall	34
4.2.3	Approach to an Edge	36
4.2.4	Motion Parallel to a Wall	38
4.2.5	Visualisation of Zero Crossings	40
4.2.6	Verification and Validation	41
4.3	Machine Learning	41
4.3.1	Choice of Machine Learning model architecture	41
4.3.2	Memory Kernels across geometry	42
4.3.2.1	Physics-Informed Neural Network Results	43
4.3.3	Validation of particle trajectory	45
5	Conclusion	47
5.1	Development of memory kernels	47
5.2	Machine Learning Surrogate	48
5.3	Validation and Limitations	48
5.4	Future Outlook	48
	References	51

List of Figures

2.1	Representative memory kernel for a spherical particle confined between two parallel walls, generated from data provided by Michael et al. [25]. The peak at zero time lag, the negative dip, and the decaying tail are the generic features described in this section.	11
3.1	Flowchart of the DNS and kernel optimisation procedure. The kernel is optimized at each time step using <code>mk_pred_multi_kernel()</code> with <code>L-BFGS-B</code> or <code>trust-constr</code> , enforcing positive semi-definiteness of the covariance matrix.	22
3.2	Collapse of off-diagonal kernel components ($\gamma_{12}, \gamma_{13}, \gamma_{23}$) after $120 \tau_p$ time steps	24
3.3	Diagram showing the distribution of points where DNS simulations are performed. The points positions start at the origin and are distributed such that they reach the wall, edge and corner of a subsection of the cube. The domain of the shown diagram is $30h/r_p$	26
3.4	Diagram showing the importance of logarithmic scaling. The two sub diagrams 3.4a and 3.4b show Memory Kernels that start(left) at the origin and end(right) close to a wall. In plot 3.4a Memory Kernels are plotted plainly whilst in plot 3.4b is plotted in logarithmic scale.	27
3.5	Flowchart of the LSTM network architecture. The sequence is expanded via <code>RepeatVector</code> and processed through deep LSTM layers utilizing orthogonal initializers, regularized by intermediate <code>Dropout</code> layers, and finalized with sequential <code>TimeDistributed</code> dense projections.	28
3.6	Flowchart of the Continuous-Time Coordinate MLP. By injecting the time coordinate t , the network maps initial spatial constraints to temporal outputs independently at each step, bypassing recurrent layers.	29
4.1	Peak region (first 10 time steps) of the diagonal and off-diagonal kernel components obtained with the original (blue) and corrected (orange) optimisation methods. The diagonal component K_{11} agrees between the two methods, whereas the off-diagonal components collapse under the original method.	31
4.2	Evolution of the diagonal kernel components (K_{11}, K_{22}, K_{33}) over the optimisation blocks.	32
4.3	Evolution of the off-diagonal kernel components (K_{12}, K_{13}, K_{23}) over the optimisation blocks.	32

4.4	Surface plot of $\log_{10}(K_{33})$ for the square channel at $10r_p$ domain. Time index on the x-axis, Positions are along the diagonal from centre to wall on the y-axis.	33
4.5	Sampling direction for the wall-approach case, shown for the cube. The particle starts at the domain centre and is moved towards the wall (red arrow). The kernel component reported is the diagonal component aligned with this direction, here K_{33} (green arrow).	34
4.6	Surface plots of $\log_{10}(K)$ for the wall-normal diagonal component, K_{33} for the cube, K_{22} for the square channel, and K_{33} for the two-wall geometry. The time-lag index is on the x -axis and the sampled position on the y -axis, and the dashed green lines mark sign changes of the kernel.	35
4.7	Sampling direction for the edge-approach case, shown for the square channel. The kernel is sampled at successive positions running from the domain centre towards the edge (red arrow), where two walls meet, and the green arrow labelled K_{xy} indicates the off-diagonal component aligned with this direction.	36
4.8	Diagonal and off-diagonal kernel surfaces for the edge-approach sampling in the $10r_p$ domain: K_{11} (square channel) and K_{22} (cube) on the top row, and the off-diagonal components K_{12} (square channel) and K_{23} (cube) on the bottom row. The y -axis is the sampled position along the diagonal from the centre to the edge, and the x -axis is the time-lag index. Dashed green lines mark sign changes.	37
4.9	Sampling direction for the parallel-motion case, shown for the cube, where the particle sits at the centre of the z -wall and the sampling proceeds towards the y -wall along the red arrow, while the green arrow labelled K_{33} marks the diagonal component in the z direction that stays perpendicular to the sampling direction.	38
4.10	Diagonal and off-diagonal kernel surfaces for motion parallel to a wall in the $15r_p$ domain, showing K_{22} for the square channel and K_{33} for the cube on the top row and the off-diagonal components K_{12} and K_{23} on the bottom row, where the x -axis is the sampled position along the line and the y -axis is the time-lag index. Dashed green lines mark sign changes.	39
4.11	Sign-coloured surface plots of the square-channel memory kernel components. The height shows $\log_{10}(K)$ and the colour encodes the sign of K : red is positive, blue is negative, and the white band marks the zero crossing.	40
4.12	Particle position over time from DNS (orange, dashed) and LPT (blue, solid) for the $10r_p$ cube. The LPT trajectory ends earlier because the particle reaches the wall sooner.	41
4.13	Comparison between different types of model architectures. The vertical axis shows the Mean Square Error for the during of training. The horizontal axis shows the number of epochs.	42

4.14	The principal diagonal components K_{11} (xx) and K_{22} (yy) of the memory kernel. The magnitudes are evaluated along a diagonal trajectory extending from the origin to the domain corner.	43
4.15	The longitudinal component K_{33} (zz) and the first transverse cross-coupling component K_{12} (xy). Both are plotted as a function of the time index and spatial position along the diagonal.	44
4.16	The remaining transverse cross-coupling memory kernel components, K_{13} (xz) and K_{23} (yz), representing the interactions between the z-axis and the orthogonal plane along the trajectory.	44
4.17	In this diagram the MSE loss is compared by the extent the PINN loss term $\mathcal{L}_{\text{History}}$ for a LSTM based Machine Learning model. The figure shows three lines, brown depicting $\lambda_2 = 1$, cyan depicting $\lambda_2 = 0.1$ and navy showing $\lambda_2 = 10$, differently weighted $\mathcal{L}_{\text{History}}$ terms. For all training sets, the Residual loss term hyperparameter $\lambda_1 = 1$ remained constant.	45
4.18	Comparison of particle position over time from DNS (solid red line) and LPT (dashed blue line). The diagram displayed the X, Y and Z positions over time. The LPT trajectory ends earlier because it reached the wall earlier, which ends the simulation.	46

1

Introduction

1.1 Background

Microfluidic devices have become increasingly prevalent in applications ranging from chemical reaction engineering to biomedical diagnostics [1], [2]. These systems manipulate small volumes of fluids within channels of sub-millimetre dimensions, where physical phenomena such as diffusion and surface forces dominate over inertial effects [1]. The design and optimisation of microfluidic devices require accurate predictive models of particle transport, particularly when particles approach confining walls or interact with surrounding boundaries [3]. At these small scales, particle transport is no longer purely deterministic. Particles suspended in such devices exhibit Brownian motion, which becomes significant when their size is comparable to the length scales of the flow.

Brownian motion can be described as the random motion of particles when they are suspended in a medium. As the motion occurs at very small length and time scales, it plays a crucial role in transport phenomena at micro- and nano-scales, and its cumulative effect can influence the performance of microfluidic systems [4]. In applications such as lab-on-chip devices, biomedical diagnostics, and particle manipulation technologies, Brownian-induced diffusion can alter particle trajectories and overall device functionality [2], [5].

The motion of a Brownian particle generates a disturbance in the surrounding fluid that evolves over finite time scales as momentum diffuses through the fluid. Consequently, the hydrodynamic force acting on the particle depends not only on its instantaneous velocity but also on its velocity history, giving rise to hydrodynamic memory effects [6]. As a particle approaches a solid wall, edge, or corner, these hydrodynamic memory effects are modified due to the presence of the boundary. Understanding how confinement alters the hydrodynamic response is essential for predicting particle behaviour in microfluidic devices, where particles frequently interact with nearby walls [3]. The history-dependent hydrodynamic force can be expressed as a convolution integral between a memory kernel tensor $\mathbf{K}(t)$ and the particle velocity history $\mathbf{V}(t)$:

$$\mathbf{F}_H(t) = - \int_0^t \mathbf{K}(t - \tau) \mathbf{V}(\tau) d\tau. \quad (1.1)$$

The memory kernel encapsulates the combined effects of drag, added mass, and history forces. Extracting accurate memory kernels for various positions and geometries is therefore a prerequisite for developing efficient and predictive models of Brownian particle transport in microfluidic systems.

1.2 Previous Work and Literature Study

The theoretical description of Brownian motion has evolved considerably since the seminal works of Einstein and Langevin, with the generalized Langevin equation (GLE) now providing a rigorous framework for incorporating hydrodynamic memory effects through the memory kernel formalism. This section reviews the key developments in this area, beginning with the classical Langevin equation and its limitations for neutrally buoyant particles, followed by the extension to the GLE and the role of the memory kernel in capturing unsteady drag forces. Finally, the work of Michael et al. is examined in detail, as their methodology provides the foundation for the present study.

1.2.1 Langevin equation

The pure-diffusion model of Einstein, while successful at long times, suffers from several deficiencies: it neglects particle inertia, predicts no ballistic regime at short times, and gives rise to an unphysical velocity autocorrelation function [7], [8]. These deficiencies were addressed by Langevin in 1908, who proposed an equation of motion that includes both a deterministic friction force and a stochastic random force, enabling a more complete description of Brownian dynamics across time scales [9]. Langevin's approach is based on Newton's second law and treats the Brownian particle as a massive object whose velocity is well-defined. The surrounding fluid exerts two distinct types of forces on the particle: a deterministic drag force proportional to the instantaneous velocity, and a fluctuating force that represents the cumulative effect of molecular collisions. The resulting equation of motion is

$$m \frac{d^2 x}{dt^2} = -\zeta \frac{dx}{dt} + \tilde{F}(t), \quad (1.2)$$

where m is the particle mass, ζ is the friction coefficient, and $\tilde{F}(t)$ is the random force. The random force is modelled as a Gaussian white noise process with zero mean.

The limitation of this description first became apparent with the advent of computer simulation. In a series of molecular dynamics studies through the 1960s, Alder and Wainwright were able to follow the motion of individual particles directly and track the decay of the Velocity Autocorrelation Function (see Section 2.1.1.2) over time. Rather than the rapid exponential decay implied by Langevin's instantaneous drag, they observed that the correlation persisted, vanishing only slowly at long times [10]. The matter was settled in 1970, when they established that this long-time behaviour follows an algebraic decay, $C(t) \sim t^{-d/2}$, where d denotes the spatial dimension of the domain [11], a scaling shortly confirmed by independent numerical solutions of the Navier–Stokes equations. This finding exposed a fundamental limitation of Langevin's description. Its memoryless friction term cannot reproduce the slow decay of the VACF.

The discovery of the algebraic long-time decay of the VACF motivated several theoretical approaches aimed at explaining the role of fluid inertia and hydrodynamic

interactions. These include hydrodynamic formulations based on the linearised Navier–Stokes equations, generalized Langevin equation (GLE) formulations derived from fluctuating hydrodynamics, mode-coupling theory, and kinetic theory [12], [13]. Although these approaches differ in their underlying assumptions and mathematical formulation, they all account for the inertia of the surrounding fluid and predict the same asymptotic algebraic decay of the VACF.

In the hydrodynamic approach, the fluid is treated as a continuum governed by the linearised Navier–Stokes equations, and the hydrodynamic force acting on the particle is formulated as a frequency-dependent generalisation of Stokes’ law [14].

In the mode-coupling approach, the slow decay of the VACF is attributed to the coupling between the tagged particle’s motion and the long-wavelength collective modes of the fluid, whose momentum relaxes only on the slow viscous time scale. This coupling produces the algebraic $t^{-d/2}$ tail, with an amplitude that can be expressed entirely in terms of the fluid transport coefficients and equilibrium thermodynamic quantities [15], [16]. Ernst, Hauge and van Leeuwen further showed this description to be formally equivalent to the hydrodynamic picture based on the linearised Navier–Stokes equations.

In the kinetic-theory approach, the long-time tail is derived directly from the microscopic collision dynamics. By summing correlated sequences of collisions (ring collisions), the same $t^{-d/2}$ decay is recovered, with an amplitude expressed in terms of the collision frequency and the reduced density [17].

Of these, the present study adopts the generalized Langevin equation, as its memory-kernel formulation captures the history-dependent hydrodynamic force directly and extends naturally to the confined geometries considered here.

1.2.2 Generalized Langevin equation

The slow decay of the VACF described above reflects the unsteady nature of the hydrodynamic force acting on a moving particle, an effect that the classical Langevin equation omits. This force was first characterised by Boussinesq and Basset, who showed that the force on a spherical particle in unsteady motion comprises three contributions, namely the steady Stokes drag, an added-mass force arising from fluid inertia, and the Basset history (or memory) force, which accounts for the viscous diffusion of momentum in the surrounding fluid [6], [18]. At low particle-to-fluid density ratios the history force becomes significant, and the hydrodynamic force can no longer be represented by the steady Stokes drag alone. The classical Langevin equation is therefore insufficient to describe Brownian motion whenever fluid inertia and hydrodynamic memory effects are appreciable.

The generalized Langevin equation (GLE) addresses these effects within the framework of fluctuating hydrodynamics [12], [13]. By replacing the constant drag coefficient with a memory kernel, it captures the history dependence of the hydrodynamic force while preserving the fluctuation-dissipation theorem (Section 2.1.1.1) through a temporally correlated, or coloured, stochastic force.

1.2.3 Hydrodynamic memory under confinement

The preceding results describe a particle in an unbounded fluid, whereas the particles of interest in microfluidic devices move close to solid boundaries. The presence of a boundary breaks the symmetry of the surrounding flow and modifies the hydrodynamic force acting on the particle. Kim and Karrila note that the appropriate analytical treatment depends on the ratio of two length scales, the particle radius and its distance from the wall [19]. When the particle is far from the wall compared with its own size, the flow can be built up by the method of reflections. The wall is accounted for by adding image singularities that correct the boundary conditions in successive steps. Lorentz introduced the image technique for a sphere near a plane wall, which gives the leading correction to the mobility [20]. Faxén later applied the method of reflections to the same problem and obtained a higher-order series for motion parallel to the wall, reducing to Lorentz’s result when truncated at leading order [21]. Happel and Brenner collect these and related results for particles of arbitrary shape and for a range of wall geometries [22]. Both treatments show that the wall reduces the mobility of the particle, and that motion perpendicular to the wall is hindered more than motion parallel to it. Liron and Mochon, and later Hackborn, obtained analytical solutions for the flows produced by Stokeslets and rotlets in channels bounded by two parallel plane walls [23], [24], results that have since served as a basis for numerical work [19]. These analytical treatments are restricted to steady motion and to simple planar wall configurations, and they do not describe the history-dependent hydrodynamic force acting on a particle in the more complex geometries encountered in microfluidic devices. This has motivated approaches in which the memory kernel is instead determined directly from resolved simulations of the confined flow.

1.2.4 Memory-kernel extraction from DNS

Recently, Michael et al. [25] proposed a continuum-based framework for simulating Brownian motion using direct numerical simulation (DNS). Their multiphase DNS resolves the hydrodynamic forces acting on a particle, from which the hydrodynamic memory kernel is identified using the particle velocity history. This kernel is subsequently used to generate a coloured Brownian force consistent with the fluctuation-dissipation theorem, while the generalized Langevin equation is employed to evolve the particle trajectory.

Michael et al. further extended this methodology to confined geometries by determining hydrodynamic memory kernels for Brownian particles near solid boundaries [26]. These kernels were incorporated into a Lagrangian particle tracking (LPT) framework, enabling particle trajectories to be predicted without relying on analytical force models or empirical wall-correction factors. The methodology was demonstrated for a Brownian particle migrating towards a wall in a microfluidic device.

1.3 Aim

This project has two aims. The first is to study how confinement modifies hydrodynamic memory while improving the optimisation framework used to extract memory kernels from DNS data. Extracting memory kernels in this way is computationally expensive, since a separate resolved simulation is required for every particle position, so building a library that covers a confined domain demands many such simulations. Using the library in a reduced-order particle tracking model adds a further cost, because the kernels must be interpolated between the sampled positions, which introduces error where the kernel varies most rapidly. A model that predicts the kernel directly from the particle's position and environment would address both issues, replacing the per-position simulations with a single trained predictor and generating kernels on demand rather than by interpolation. The second aim is therefore to develop a machine learning model that predicts these memory kernels in place of direct numerical simulation (DNS).

1.3.1 Optimisation of the DNS Framework

While the existing kernel optimisation framework identifies the memory kernel that best matches the simulated hydrodynamic force and velocity history in the cases with planar wall boundaries, significant errors in the extracted kernels were observed due to deficiencies in the optimisation process when it came to domains where the particle was confined in multiple directions. The present work addresses these issues through a range of modifications and improvements to the optimisation framework, aimed at improving robustness, physical consistency, and the overall accuracy of the extracted memory kernels.

1.3.2 Confinement Effects Across Domain Sizes

To study how confinement modifies hydrodynamic memory, simulations are performed in a cubic domain and a square channel at three sizes: $30r_p$ (negligible wall effects, unbounded reference case), $10r_p$ (moderate confinement), and $5r_p$ (strong confinement). Within each domain, the particle is simulated at specific positions to capture the effects of being at the centre, near a single wall, near an edge (two walls), and near a corner (three walls). These positions are selected by recursively dividing the distance from the wall by $h^{1/3}$, with h being the wall correction factor. Starting from the centre of the domain, successive positions are calculated as $d_2 = d_1/h^{1/3}$, with d being the distance to the wall, until the wall is reached. This procedure is applied independently in all three spatial coordinates, generating a grid of positions that resolves the rapid variation of hydrodynamic memory near confining surfaces. At each position, the particle is subjected to an excitation sequence of seven distinct forcing directions to study its hydrodynamic response to Brownian forces. Comparing the extracted memory kernels across positions and domain sizes quantifies how confinement progressively influences particle dynamics.

1.3.3 Replacing Direct Numerical Simulation with Machine Learning

The final aim of this project is to create a Machine Learning model that learns from generated DNS data and is able to replicate the computational behaviour of the DNS solver and extrapolate to further geometries. The benchmark for such a model is the ability to replicate the Direct Numerical Simulation spatial evolution of the particle. That is, that the DNS solver and the ML model share similar paths for the particles movement through space. A smaller or very small difference between paths would mean that the ML model is robust and applicable.

When one aims to simply replace a DNS solver completely, the consistency, accuracy and adaptability is what makes a good ML model great. This project aims to engineer an ML model that is able to replicate the DNS solvers Memory Kernels accurately. Further, it aims to reproduce the complexity and cross correlation that is found in the data. The ML model should be able to consistently be able to generate memory kernels similar or exactly to those generated from the DNS solver. Moreover, it should be able to be adaptable to different environment geometries, particle sizes and fluid properties.

2

Theory

This chapter presents the theoretical background for the work in three parts. It first establishes how hydrodynamic memory arises in Brownian motion and how it is captured by the generalized Langevin equation through a memory kernel. It then describes the immersed boundary method used to resolve the fluid flow around the particle and extract the memory kernel. Finally, the machine learning framework used to predict the memory kernels is introduced, providing a means of obtaining them without performing a full simulation at every particle position.

2.1 Generalized Langevin Equation and Memory Kernels

The Langevin equation provides a fundamental description of Brownian motion by balancing particle inertia with a velocity proportional drag force and a random force [9]. From a hydrodynamic perspective, this equation assumes that the fluid responds instantaneously to the particle's motion. This assumption is valid only when the particle is much denser than the fluid [6]. For particles with density comparable to that of the fluid, the flow field induced by past motions decays slowly and creates a memory effect. The present hydrodynamic force then depends on the entire velocity history. This history dependence is captured by the generalized Langevin equation (GLE), which replaces the instantaneous drag term with a convolution integral involving a memory kernel [27]. In this section we first review the classical Langevin equation and then extend it to include hydrodynamic memory, a crucial step for simulating Brownian particles in confined microfluidic environments.

2.1.1 Classical Langevin Equation

Langevin described the Brownian particle as a massive object obeying Newton's second law, subject to two forces exerted by the surrounding fluid. The first is a deterministic drag proportional to the instantaneous velocity, and the second is a rapidly fluctuating force representing the cumulative effect of molecular collisions. In one dimension the equation of motion reads

$$m \frac{d^2 x}{dt^2} = -\zeta \frac{dx}{dt} + \tilde{F}(t), \quad (2.1)$$

or equivalently

$$m \frac{dv(t)}{dt} = -\zeta v(t) + \tilde{F}(t), \quad (2.2)$$

where m is the particle mass, $\zeta = 6\pi\mu_f r_p$ is the Stokes friction coefficient, $v(t)$ is the particle velocity, and $\tilde{F}(t)$ is the random force exerted by the surrounding fluid [9].

The random force $\tilde{F}(t)$ is modelled as a Gaussian white noise process, which is characterised by

$$\langle \tilde{F}(t) \rangle = 0, \quad \langle \tilde{F}(t) \tilde{F}(t') \rangle = \Gamma \delta(t - t'), \quad (2.3)$$

with Γ the noise strength, to be determined below. The noise is further assumed to be independent, so that $\langle \tilde{F}(t) x(t') \rangle = \langle \tilde{F}(t) v(t') \rangle = 0$ for $t \geq t'$. The delta correlation in (2.3) states that the force retains no memory between successive instants, which is justified when the collision time scale of the fluid molecules is far shorter than any time scale resolved in the particle dynamics.

2.1.1.1 Fluctuation-dissipation theorem

Writing (2.1) in terms of the velocity gives

$$m \frac{dv}{dt} = -\zeta v + \tilde{F}(t), \quad (2.4)$$

a first-order inhomogeneous equation with the formal solution

$$v(t) = v(0) e^{-t/\tau_p} + \frac{1}{m} \int_0^t e^{-(t-\tau)/\tau_p} \tilde{F}(\tau) d\tau. \quad (2.5)$$

Squaring (2.5) and averaging, the cross term vanishes because the noise is independent, and the remaining contribution is evaluated using (2.3), giving

$$\langle v^2(t) \rangle = \langle v^2(0) \rangle e^{-2t/\tau_p} + \frac{\Gamma}{2\zeta m} (1 - e^{-2t/\tau_p}). \quad (2.6)$$

For $t \gg \tau_p$ the exponential terms vanish and the mean-squared velocity approaches the constant value $\Gamma/2\zeta m$. Equating this to the equipartition result $\langle v^2 \rangle = k_B T/m$ fixes the noise strength as

$$\Gamma = 2\zeta k_B T. \quad (2.7)$$

Equation (2.7) is the fluctuation-dissipation theorem for the Langevin model [28], and it requires the magnitude of the fluctuating force to be balanced against the strength of the dissipation so that a well defined temperature is maintained. The relation is a physical necessity rather than a mathematical coincidence, since the drag and the random force both originate from collisions with the surrounding fluid molecules, and together they act as a thermostat on the particle [27].

2.1.1.2 Velocity autocorrelation function

The velocity autocorrelation function (VACF) measures the extent to which the velocity of the particle at one instant remains correlated with its velocity a time t later, and is defined as

$$C(t) = \langle v(0) \cdot v(t) \rangle, \quad (2.8)$$

where the angular brackets denote an equilibrium ensemble average. It therefore quantifies the rate at which the particle loses memory of its initial motion through

interaction with the fluid. Multiplying (2.5) by $v(0)$ and averaging, the contribution from the random force again vanishes, and applying equipartition to the remaining term gives

$$C(t) = \langle v^2(0) \rangle e^{-\zeta t/m} = \frac{3k_B T}{m} e^{-\zeta t/m}. \quad (2.9)$$

The classical Langevin equation therefore predicts that velocity correlations decay exponentially, on the time scale m/ζ , which is the particle response time τ_p introduced in Section 2.1.2.

2.1.2 Relevant Time Scales

In an unhindered scenario, the steady drag force on a spherical particle is $\zeta = 6\pi\mu_f r_p$, where r_p is the particle radius and μ_f the fluid viscosity [29]. With the particle mass $m_p = \frac{4}{3}\pi r_p^3 \rho_p$, one defines the particle response time as

$$\tau_p = \frac{m_p}{\zeta} = \frac{2}{9} \frac{r_p^2 \rho_p}{\mu_f}, \quad (2.10)$$

which sets the time scale over which the particle velocity relaxes under steady drag. The fluid momentum diffuses over the particle radius on a separate time scale

$$\tau_f = \frac{r_p^2 \rho_f}{\mu_f} = \frac{r_p^2}{\nu}, \quad (2.11)$$

where $\nu = \mu_f/\rho_f$ is the kinematic viscosity of the fluid. Their ratio, $\tau_p/\tau_f = 2\rho_p/(9\rho_f)$, is governed entirely by the particle-to-fluid density ratio and determines whether unsteady effects are significant.

When the particle is much denser than the fluid, $\tau_p/\tau_f \gg 1$, the fluid history effects act on time scales much shorter than the particle response time and have negligible impact. The hydrodynamic force is then dominated by steady drag, and the classical Langevin equation is adequate. When the densities are comparable, $\tau_p/\tau_f \approx 1$, as is the case for the neutrally buoyant particles considered here, the unsteady drag operates on time scales comparable to τ_p and must be retained.

When the particle is close to a solid wall, the time scale $\tau_w = h^2/\nu$ characterises the moment when the particle induced vorticity reaches the wall. One then expects wall effects to influence particle motion. Similarly, the time scale $\tau_A = L^2/\nu$ represents when finite size effects become apparent in a simulation intended to mimic an unbounded domain [30].

2.1.3 Generalized Langevin Equation

For particles with density comparable to that of the fluid, the instantaneous drag of the classical Langevin equation is insufficient, and the full history dependence of the hydrodynamic force must be retained. The generalized Langevin equation (GLE) achieves this by replacing the steady drag term with a convolution between a memory kernel and the velocity history,

$$m \frac{dv(t)}{dt} = - \int_0^t K(t-\tau) v(\tau) d\tau + \tilde{F}(t), \quad (2.12)$$

in which the random force is correspondingly coloured rather than white, and remains linked to the kernel through the fluctuation-dissipation theorem [28]. The memory kernel encapsulates the steady drag, added-mass, and history (Basset) contributions in a single object. Because it is a continuum representation of the fluid response, it can be obtained from a hydrodynamic description of the flow around the particle, without recourse to the expensive molecular dynamics simulations from which it may in principle also be derived [31].

2.1.3.1 Memory Kernel Formulation

At low Reynolds numbers ($\text{Re} \ll 1$), the hydrodynamic force is expressed as a convolution integral between a memory kernel tensor K and the particle velocity history [12], [32],

$$F_{H,i}(t) = - \int_0^t K_{ij}(h, \tau) v_j(t - \tau) d\tau. \quad (2.13)$$

Because the kernel is a tensor, the force in a given direction depends on the velocity history in all three orthogonal directions; for each lag time τ , $K_{ij}(h, \tau)$ is a 3×3 matrix quantifying how the velocity components at a past time contribute to the present force. The kernel further depends parametrically on the particle's distance h from any bounding surface, so that it encodes not only the motion history but also the local confinement.

2.1.4 Characteristics of the memory kernels

The memory kernel obtained from the methodology of Michael et al. [25] exhibits a characteristic shape that reflects the interplay of the steady drag, added-mass, and history contributions to the hydrodynamic force. A representative kernel, for a particle confined between two parallel walls, is shown in Figure 2.1, and it displays the peak, dip, and decaying tail described below.

At zero time lag the kernel shows a sharp peak corresponding to the instantaneous steady drag, which appears as a discretised Dirac delta of magnitude $2\zeta/\Delta\tau$. When the added-mass contribution is included, this peak is augmented to

$$K(0) = \frac{2}{\Delta\tau} \left(\zeta + \frac{m_p \rho_f}{2\rho_p \Delta\tau} \right), \quad (2.14)$$

so that its height grows as the particle-to-fluid density ratio ρ_p/ρ_f decreases and the added-mass effect becomes more prominent.

At small but finite time lag the kernel exhibits a negative dip. This arises from the added-mass force, which is proportional to the particle acceleration and therefore contributes a negative dependence on the velocity at earlier times. The dip is most pronounced at low density ratios, where the added mass is largest. The history (Basset) force provides a positive contribution that partially offsets the dip, and at higher density ratios can lift the second kernel value back above zero.

At longer time lags the kernel decays towards zero as the vorticity generated by the particle dissipates into the fluid. For a spherical particle in an unbounded fluid this tail follows a $t^{-3/2}$ scaling, in agreement with the analytical kernel derived by

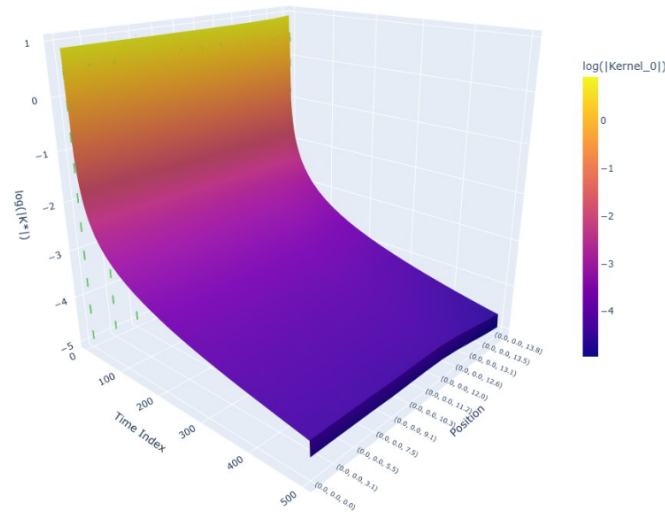


Figure 2.1: Representative memory kernel for a spherical particle confined between two parallel walls, generated from data provided by Michael et al. [25]. The peak at zero time lag, the negative dip, and the decaying tail are the generic features described in this section.

Dufty [33]. The strength of the tail is set by the viscous time scale τ_f , which becomes small relative to the particle response time at high density ratios, so that the tail magnitude diminishes, and in the limit $\rho_p/\rho_f \rightarrow \infty$ the kernel reduces to the pure steady-drag form $2\zeta \delta(\tau)$ characteristic of white-noise Brownian motion.

In confined geometries the wall modifies the kernel in a manner that directly reflects the altered hydrodynamic environment. Close to a wall the peak shifts upwards, indicating an increase in the drag and added-mass contributions, and this increase is more pronounced in the wall-normal direction. This reshaping of the memory kernel under confinement, and its extension to the edge and corner geometries encountered in microfluidic devices, is a central concern of the present work.

2.2 Immersed Boundary Method

Flows around arbitrary, moving, and interacting bodies are encountered across many fields of science and engineering, yet their behaviour remains difficult to predict. To gain fundamental insight into such flows, direct numerical simulations (DNS) at small scales are required, where the flow around each particle is fully resolved without introducing empirical models. The results from such resolved simulations can then be used to develop and improve large-scale modelling approaches.

Traditional body-fitted mesh methods require the computational grid to conform to the moving boundaries, necessitating costly remeshing at each time step. Immersed boundary methods (IBMs) offer an alternative by decoupling the fluid mesh from the geometry of the moving bodies. In these methods, the fluid is discretised on a regular Eulerian grid, while the boundary of each body is represented by a Lagrangian mesh. The coupling between the two representations enforces the no-slip

and no-penetration conditions at the fluid-solid interface without requiring the grid to conform to the body geometry [34].

2.2.1 Governing Equations

The fluid motion is governed by the incompressible Navier–Stokes equations,

$$\frac{\partial u_i}{\partial x_i} = 0, \quad (2.15)$$

$$\rho_f \left(\frac{\partial u_i}{\partial t} + u_j \frac{\partial u_i}{\partial x_j} \right) = -\frac{\partial p}{\partial x_i} + \mu_f \frac{\partial^2 u_i}{\partial x_j^2} + f_i, \quad (2.16)$$

where u_i is the fluid velocity, x_i is the spatial coordinate, p is the pressure, ρ_f is the fluid density, μ_f is the dynamic viscosity, and f_i represents any applied body force. The particle–fluid interface is enforced through an immersed boundary formulation in which no-slip and no-penetration conditions are imposed implicitly at the particle surface. The fluid velocity at the immersed boundary is constrained to match the local particle velocity, ensuring fully coupled momentum exchange between the solid and fluid phases. The hydrodynamic force acting on the particle is obtained by integrating the pressure and viscous stresses over the particle surface,

$$F_{H,i}(t) = \int_S (-p\delta_{ij} + \sigma_{ij}) n_j dS, \quad (2.17)$$

where δ_{ij} is the Kronecker delta, σ_{ij} is the viscous stress tensor, and n_j is the outward surface normal vector. The viscous stress tensor is given by

$$\sigma_{ij} = \mu_f \left(\frac{\partial u_i}{\partial x_j} + \frac{\partial u_j}{\partial x_i} \right). \quad (2.18)$$

In a body-fitted method these equations would be solved on a grid conforming to the particle surface. In the immersed boundary formulation, by contrast, the fluid is solved on a fixed Eulerian grid that does not conform to the particle, and the presence of the particle is imposed instead through the interface condition, either by means of the body force f_i in the momentum equation or by a direct constraint on the velocity field in the boundary cells. The particular way in which this constraint is realised distinguishes the various immersed boundary methods, which are compared in the following section.

2.2.2 Coupling Strategies for Immersed Boundaries

Immersed boundary methods differ chiefly in how the interface condition is transmitted between the Eulerian fluid grid and the Lagrangian representation of the body. The main strategies are summarised below.

- **Lagrange multipliers.** The constraint is incorporated into the weak form of the Navier–Stokes equations through a Lagrange multiplier. Glowinski et al. [35] introduced this fictitious-domain approach in a finite element setting, where it is implicit and second-order accurate but does not carry over readily to finite volume discretisations. Sharma and Patankar [36], [37] recast the

method for finite volume solvers, though the interface force remained explicitly coupled.

- **Explicit forcing.** The influence of the body is represented by a force added to the momentum equation. In Peskin’s original formulation [38] this force is spread across neighbouring grid nodes by a regularised Dirac delta, which limits the method to first-order accuracy. Mohd-Yusof [39], [40] instead applied the forcing directly at the boundary cells to hold the fluid at the interface velocity, achieving second-order accuracy; the resulting mass-conservation error was later remedied by Kim and Choi [41]. This class of methods is widely adopted for its simplicity.
- **Implicit boundary conditions:** Rather than adding a force, these methods constrain the interface velocity within the discretised equations themselves. Majumdar et al. [42] formulated such a condition for stationary bodies coupled to a RANS solver. As noted by Mark and van Wachem [43], this type of scheme can be sensitive to the interpolation weights and may permit a non-physical mass flux through the interface.

2.2.3 The Mirroring Immersed Boundary Method

The present work adopts the mirroring immersed boundary method together with a hybrid interface condition [44]. Whereas the original condition only reflects the velocity across the boundary, the hybrid condition additionally extrapolates the resolved field onto it, introducing extra points at which the fluid is held to the local body velocity. These additional constraint points lower the interpolation error along the interface while preserving second-order accuracy.

Mark et al. [44] validated the method on two configurations, namely steady flow past a fixed sphere across a range of Reynolds numbers and a sphere settling under gravity with the refined grid tracking its descent. In both cases the computed drag and terminal velocities matched analytical and semi-empirical data, and no oscillations appeared in the surface forces as the body moved through the mesh.

2.2.4 IBOFlow Solver

The simulations in the present work were carried out in IBOFlow (Immersed Boundary Octree Flow Solver), an incompressible finite-volume code in which the mirroring immersed boundary method is implemented [44]. The governing equations are discretised on a Cartesian grid organised as an octree, which can be locally refined and coarsened during the simulation. This structure allows the mesh to be concentrated around each body and to follow its motion as it moves through the domain, so that high resolution is maintained near the interface without refining the entire grid.

Coupling between pressure and velocity is handled with the SIMPLEC algorithm [45]. The flow variables are stored at a common set of cell-centred locations, and pressure-weighted flux interpolation [46] is applied to prevent the pressure and velocity fields from decoupling on this co-located arrangement. The octree data structure also supports efficient search operations, which are used to associate the immersed boundaries with the surrounding grid cells. The unsteady terms of the Navier–Stokes

equations are advanced with an implicit backward Euler scheme, while the motion of the immersed bodies is integrated using a second-order Crank–Nicolson scheme [47].

2.3 Model Architecture

The objective of the model is to generate Memory Kernels such that the physical trajectory of the particle is aligned with the generated results from DNS. With that purpose in mind, multiple different models were developed, improved and compared. The most promising model architectures which were tested were Long-short term unit, Principal Component Analysis and a functional basis approach. The loss function used while training these models were also altered, changed and compared. Moreover, the selection of data was both considered and of great importance. The size of the networks was altered to be well suited to the size of the dataset.

2.3.1 Long-short term memory

Recurrent neural networks use feedback connections to store representations of recent input events in form of activations, a form of short-term memory. Long-short term memory (LSTM) can learn to bridge the gap between long and short term memory with a novel gradient-based method. By enforcing constant error flow through constant error carousels within special units, multiplicative gate units learn to open and close access to the constant error flow [48]. LSTM is able to solve complex and long time lag tasks and is thus well-suited towards this projects long and dynamic memory kernels.

2.4 Loss functions

This chapter pertains to the loss functions which are used and compared throughout this project.

2.4.1 Logarithmic loss function

The logarithmic based function used in this project allows the training of the network more equally valuing the memory kernels temporal steps.

$$\mathcal{L} = \log(1 + |K_{pred} - K_{measured}|) \quad (2.19)$$

2.4.2 Physics Informed Neural Networks

Gaussian-process regression and physics-informed neural networks (PINNS) [49] can seamlessly integrate multifidelity/multimodality experimental data with various incompressible and biomedical flows [50]. PINNs are typically used for solving PDEs and comprises of modifying the loss function which is used during the optimization of neural networks. A history loss term $\mathcal{L}_{\text{History}}$ which accounts for the convolution between the memory kernel and the history of the particle’s velocity (2.13) is

added to the loss function. This results in the total loss($\mathcal{L}_{\text{Total}}$) being the sum of the residual loss $\mathcal{L}_{\text{Res}}$ and a second physics-inspired term $\mathcal{L}_{\text{History}}$, see equation (2.20)

$$\begin{aligned} \mathcal{L}_{\text{Total}} = & \underbrace{\lambda_1 \|K_{\text{pred}} - K_{\text{measured}}\|^2}_{\text{Kernel Prediction Error}} \\ & + \underbrace{\lambda_2 \|F_H - \int_0^t K_{\text{pred}}(h, \tau) \cdot V(t - \tau) d\tau\|^2}_{\text{Hydrodynamic Force Term}} \end{aligned} \quad (2.20)$$

where λ_1 and λ_2 are hyperparameters.

3

Methodology

This chapter presents the methodology in which hydrodynamic memory kernels were obtained from high-fidelity fluid–structure interaction simulations. Further, where memory kernels are approximated using machine learning and incorporated into a reduced-order generalized Langevin particle solver. In the first stage, hydrodynamic memory kernels were extracted directly from high-fidelity immersed boundary finite element simulations for a spherical particle at discrete spatial locations within different confined domains. These kernels were then assembled into a spatially indexed kernel library that captures the position-dependent hydrodynamic response of the system. In the second stage, the memory kernel library was used to develop a machine learning surrogate model. Multiple architectures and model types were tested to identify a suitable predictor for the memory kernels across the sampled positions. After evaluating various candidates, a two-model system was adopted: one model was dedicated to predicting the initial peaks of the kernel, while the other focused on the longer tails. This separation allowed each component to capture the distinct temporal features of the kernel more effectively. The chosen model was then integrated into a Lagrangian particle tracking solver in the subsequent stage to determine particle trajectories.

3.1 Generating Memory Kernels

The extraction of hydrodynamic memory kernels from direct numerical simulations constitutes the first stage of the methodology. This section outlines the DNS setup, encompassing the computational domain, boundary conditions, fluid and particle properties, and numerical implementation. The optimisation framework used to extract the memory kernels from the simulation data is then presented, along with the corrective measures implemented to improve robustness and physical consistency.

3.1.1 Direct Numerical Simulation Framework

The hydrodynamic memory kernels used in this work are extracted from fully resolved direct numerical simulations (DNS) of a spherical particle suspended in a viscous fluid. The fluid–structure interaction was resolved using the mirroring immersed boundary method [43], implemented in the finite-volume solver IBOFlow [44], coupled to a rigid-body structural solver for the particle. At each time step the fully resolved pressure and velocity fields were used to evaluate the hydrodynamic force on the particle by surface integration, providing the force history required for kernel

extraction.

3.1.1.1 Computational Domain and Mesh

The simulations considered a spherical particle of radius $r_p = 2 \times 10^{-5}$ m suspended in two types of confined geometries: a cubic domain and a square channel. In both cases, the computational domain had characteristic dimensions of $L = 5r_p$, $10r_p$, or $30r_p$. For the cubic domain, the particle was confined in all three Cartesian directions, while for the square channel, the particle was confined only in the x - and y -directions and free to move along the z -axis. The corresponding boundary conditions for each geometry are described in detail in Section 3.1.1.2.

The computational domain was discretised using a Cartesian octree grid. A base grid resolution corresponding to approximately 5 cells across the particle diameter was employed, giving a base cell size of $r_p/0.4$. To accurately resolve the near-particle flow, a refinement level of six was applied within a cubic region of side length $2r_p$ centred on the particle, resulting in a local grid spacing of $r_p/(0.4 \times 2^6)$. This refinement region translated with the particle throughout the simulation, ensuring that the boundary layer remained highly resolved. Away from the particle, the octree mesh was progressively coarsened, significantly reducing the total number of computational cells and the associated computational cost.

3.1.1.2 Boundary Conditions

Two sets of boundary conditions were applied, corresponding to the two geometries investigated:

1. **Cubic domain** For the cubic geometry, no-slip wall conditions were imposed on all six faces of the domain, i.e., the fluid velocity was set to zero at every boundary. This configuration confined the particle in all three Cartesian directions. The domain size varied as $L = 5r_p$, $10r_p$, or $30r_p$, with the particle positioned at specified distances from the walls.
2. **Square channel** For the square channel geometry, the same overall domain dimensions were used, but the boundaries normal to the z -axis were treated as outflow (zero-gradient) conditions, with the static pressure set to zero gauge. No-slip wall conditions were retained on the four faces normal to the x - and y -axes ($x = \pm L/2$, $y = \pm L/2$). Consequently, the particle was confined only in the x and y directions and free to move along the z -axis without interacting with walls in that direction.

3.1.1.3 Fluid and Particle Properties

The working fluid was water and the particle a rigid sphere with density matched to the fluid, so that the Brownian motion studied was that of a neutrally buoyant particle. The governing physical parameters are listed in Table 3.1.

From these, the particle mass follows as $m_p = \frac{4}{3}\pi r_p^3 \rho_p$ and the Stokes drag coefficient as $\zeta = 6\pi\mu_f r_p$.

Table 3.1: Fluid and particle properties used in the simulations.

Property	Symbol	Value
Fluid density	ρ_f	1000 kg/m ³
Fluid viscosity	μ_f	1×10^{-3} Pa s
Particle density	ρ_p	1000 kg/m ³
Particle radius	r_p	20 μ m
Temperature	T	293.15 K

Two characteristic time scales govern the dynamics. The particle response time,

$$\tau_p = \frac{\rho_p(2r_p)^2}{18\mu_f}, \quad (3.1)$$

sets the rate at which the particle velocity relaxes under steady drag, while the viscous time scale,

$$\tau_f = \frac{r_p^2}{\nu} = \frac{r_p^2 \rho_f}{\mu_f}, \quad \text{where } \nu = \frac{\mu_f}{\rho_f}. \quad (3.2)$$

gives the time for momentum to diffuse across the particle. For the neutrally buoyant case their ratio reduces to $\tau_p/\tau_f = 2\rho_p/(9\rho_f) \approx 0.22$, so that the two scales are of the same order and unsteady hydrodynamic effects are significant, which is precisely the regime in which the generalized Langevin description is required.

The strength of the thermal forcing is set by the fluctuation-dissipation theorem. The force applied in each excitation has spectral intensity

$$S = \frac{216 \mu_f k_B T}{\pi^2 (2r_p)^5 \rho_p^2}, \quad (3.3)$$

where k_B is the Boltzmann constant, and the root-mean-square thermal velocity is $V_{\text{rms}} = \sqrt{k_B T / m_p}$. The corresponding particle Reynolds number, $\text{Re}_p = 2r_p V_{\text{rms}} / \nu \approx 4 \times 10^{-4}$, confirms that the motion lies deep in the Stokes regime.

3.1.1.4 Excitation Sequence and Temporal Discretisation

The memory kernel is a symmetric 3×3 tensor with six independent components, and recovering all of them requires exciting the particle along enough independent directions that every component appears in the resulting force response. Each forcing direction is written as a vector whose entries give the relative components of the applied force along the x , y and z axes, so that $[1, 0, 0]$ drives the particle along the x axis alone, $[1, 1, 0]$ drives it diagonally in the x - y plane, and $[1, 1, 1]$ drives it along the body diagonal of the domain. The particle was subjected to the seven directions:

$$[1, 0, 0], [0, 1, 0], [0, 0, 1], [0, 1, 1], [1, 0, 1], [1, 1, 0], [1, 1, 1].$$

which combine the three axial excitations, needed for the diagonal components of the tensor, with the four diagonal excitations that couple pairs of axes and so reveal the off-diagonal components.

For each excitation, a constant point force of magnitude

$$F_{\text{mag}} = m_p \sqrt{\frac{\pi S}{\Delta t}},$$

was applied at the particle centre in the direction opposite to the corresponding excitation vector, where S is the Brownian spectral intensity defined in Equation (3.3). The force remained constant for a duration of $20 \tau_p$, generating a sustained velocity and force history for kernel identification.

The particle was initially at rest at the prescribed location (centre, near wall, edge, or corner). The seven excitation blocks were applied consecutively, giving a total simulation time of $140 \tau_p$, which was found to be sufficient for convergence of the identified kernel.

3.1.1.5 Numerical Implementation

The DNS framework is implemented through a modular architecture comprising three main solvers, instantiated in the `kernel_build()` function:

- **Fluid solver** (`create_iboflow_solver()`): the incompressible Navier-Stokes equations are discretised and solved in IBOFlow on an adaptive Cartesian oct-tree grid. A base cell size of $r_p/0.4$ is used, with the grid refined to level six within a region of side $4r_p$ around the particle to resolve the near-surface flow.
- **Structural solver** (`create_lastfem_solver()`): the particle is represented as a rigid body whose surface mesh is read from file and translated to its initial position. Its motion is advanced with an explicit Newmark scheme ($\beta = 0$, $\gamma = 1/2$), consistent with the Itô interpretation of the stochastic integration.
- **Fluid-structure coupling** (`ipsfsi.FSISimulation()`): the two solvers are coupled explicitly through the mirroring immersed boundary method, which enforces the no-slip and no-penetration conditions at the particle surface. Each time step is iterated to a relative tolerance of 10^{-2} , up to a maximum of five coupling iterations.

At each step the converged pressure and velocity fields are used to evaluate the hydrodynamic force on the particle by surface integration, which then drives the memory kernel extraction described in the following section.

3.1.1.6 Simulation Workflow

Figure 3.1 presents a schematic overview of the simulation and kernel optimisation workflow. The algorithmic steps are as follows:

1. **Initialization:** The fluid and structural solvers are instantiated, and the memory kernel tensor $\mathbf{K}(t, \tau)$ is initialised as a zero tensor with its diagonal seed value set to 10.

2. **Time-stepping loop:** For each global time step (index i , up to $\mathbf{taus} \times \mathbf{Nts}$), the following substeps are performed:
 - (a) **Fluid time step:** the fluid solver advances by one step with the particle position held fixed.
 - (b) **Hydrodynamic force extraction:** the current fluid force on the particle is obtained through `get_fluid_force()`.
 - (c) **Kernel optimisation:** the memory kernel is updated by minimising the difference between the measured hydrodynamic force and the convolution of the trial kernel with the velocity history,

$$\mathbf{F}_{\text{pred}}(t) = - \int_0^t \mathbf{K}(t, \tau) \mathbf{V}(t - \tau) d\tau, \quad (3.4)$$

evaluated over the five most recent force samples simultaneously. The minimisation proceeds in up to three stages:

- unconstrained **L-BFGS-B**;
 - if the covariance matrix is not positive semi-definite, the final quarter of the kernel is set to zero and **L-BFGS-B** is repeated;
 - if it remains non-positive-semi-definite, a constrained **trust-constr** optimisation is applied.
- (d) **Structure time step:** the structural solver advances, updating the particle position, velocity and acceleration.
 - (e) **Data logging:** position, velocity, acceleration, forces and kernel values are written to file for post-processing.
3. **Excitation sequence:** the full procedure is repeated for seven forcing directions, $[1, 0, 0]$, $[0, 1, 0]$, $[0, 0, 1]$, $[0, 1, 1]$, $[1, 0, 1]$, $[1, 1, 0]$ and $[1, 1, 1]$ (each applied in the negative sense; see Section 3.2.3.1), to ensure identifiability of all kernel tensor components.

3.2 Memory Kernel Extraction from DNS

3.2.1 Optimisation Framework

For each simulation run, the memory kernel tensor $\mathbf{K}(t)$ was extracted from the DNS data by solving an inverse problem. The hydrodynamic force $\mathbf{F}_H(t)$ and the particle velocity $\mathbf{V}(t)$ were recorded throughout the simulation. The kernel was then found by minimising the discrepancy between the measured force and the convolution integral

$$\mathbf{F}_H(t) = - \int_0^t \mathbf{K}(t - \tau) \mathbf{V}(\tau) d\tau. \quad (3.5)$$

The optimisation was performed using the L-BFGS-B algorithm [51], [52]. The full 3×3 kernel tensor was optimised simultaneously, capturing cross-couplings between different velocity components. The kernel also governs the covariance of the coloured Brownian force via the fluctuation-dissipation theorem, which requires its covariance matrix to be positive semi-definite. To enforce this, we first set the last quarter of the kernel to zero and re-ran the optimisation whenever the condition

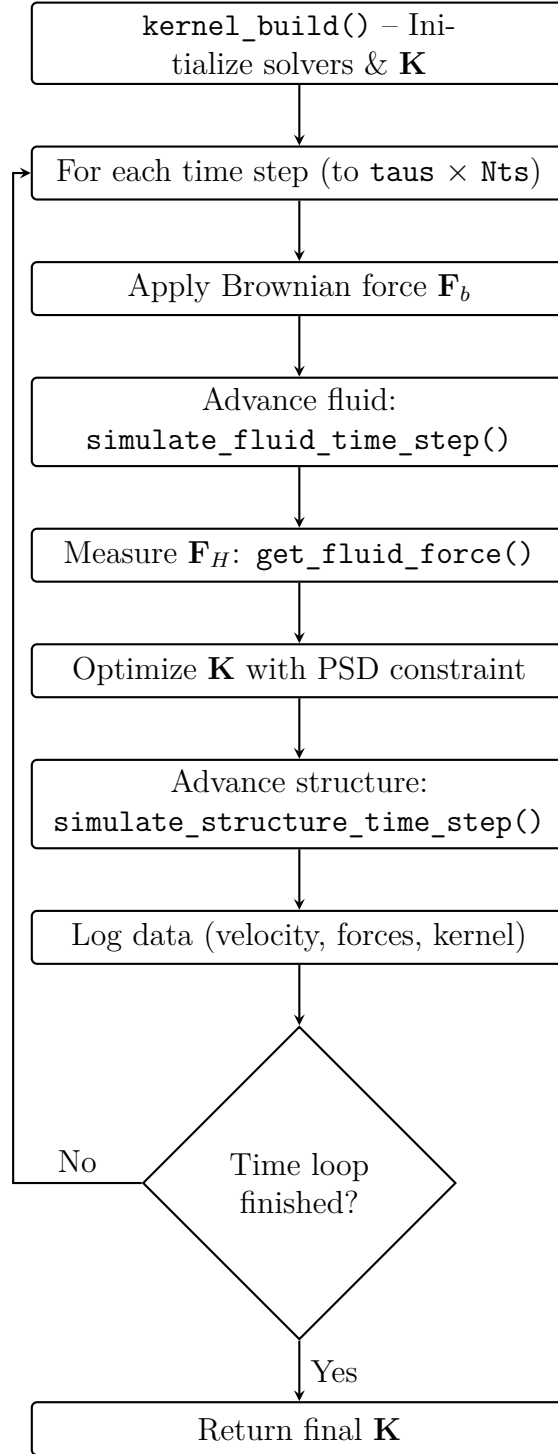


Figure 3.1: Flowchart of the DNS and kernel optimisation procedure. The kernel is optimized at each time step using `mk_pred_multi_kernel()` with L-BFGS-B or `trust-constr`, enforcing positive semi-definiteness of the covariance matrix.

was violated. If the violation remained, we then applied a constrained optimisation using `trust-constr`. The final 500 time steps of each converged kernel were saved for each particle position.

3.2.2 Sampling of Particle Positions

Within each domain, the particle was placed at a set of positions chosen to resolve the variation of the kernel with distance to the walls, edges and corners. The sampling followed the scheme of Michael et al. [26], in which the positions are spaced according to the near-wall growth of the added-mass force. As the particle approaches a wall, the added mass increases in proportion to $(r_p/h)^3$, where h is the distance from the particle centre to the wall. The sampled distances were therefore chosen so that this quantity doubled from one position to the next, which concentrates the sampling in the near-wall region where the kernel varies most rapidly.

This criterion is met by the recurrence

$$h_{k+1} = \frac{h_k}{2^{1/3}}, \quad (3.6)$$

which reduces h by a factor of $2^{1/3}$ at each step so that $(r_p/h)^3$ doubles. Starting from the domain centre ($h_0 = L/2$, for example $h_0 = 15r_p$ in the $30r_p$ domain), twelve wall distances were generated per confined direction, ranging from the centre to a closest approach of $h \approx 1.18r_p$.

For the cubic geometry the particle was confined in all three directions, and positions were generated from combinations of the sampled distances (h_x, h_y, h_z) , giving 84 unique positions. For the square channel, confinement acted only in x and y , with outflow boundaries in z , giving 28 positions.

3.2.3 Excitation Sequence

The particle was excited using seven distinct forcing directions (see Section 3.1.1.4 for details). Each forcing was applied for a duration of $20\tau_p$ (where $\tau_p = \rho_p(2r_p)^2/(18\mu_f)$ is the particle relaxation time), and the simulation was then restarted from the same initial position with the next forcing direction. This procedure produced a set of velocity and force histories from which the full kernel tensor could be determined uniquely.

3.2.3.1 Numerical Instability for the [1,1,1] Forcing Direction

For highly confined positions, such as those near corners in the cubic domain, the off-diagonal kernel components (K_{12} , K_{13} , K_{23}) were found to collapse abruptly to zero after roughly $120\tau_p$ time steps, under the final forcing direction $[1, 1, 1]$. The collapse is shown in Figure 3.2, where all three off-diagonal terms drop sharply to zero while the diagonal components (not shown) remain physically plausible throughout.

The instability was traced to the orientation of the applied force relative to the confining geometry. Under the original (positive) forcing, the $[1, 1, 1]$ direction drove the particle diagonally towards the corner where the walls meet, into the most strongly confined region of the domain. In this configuration the off-diagonal optimisation

became ill-conditioned and collapsed. Reversing every forcing vector to its negative sense drove the particle diagonally away from the corner instead, keeping it in a better-resolved region while exciting exactly the same set of directions. This global sign reversal removed the collapse entirely, and the corrected framework produced stable off-diagonal kernels across all positions and forcing directions, as shown in the Results.

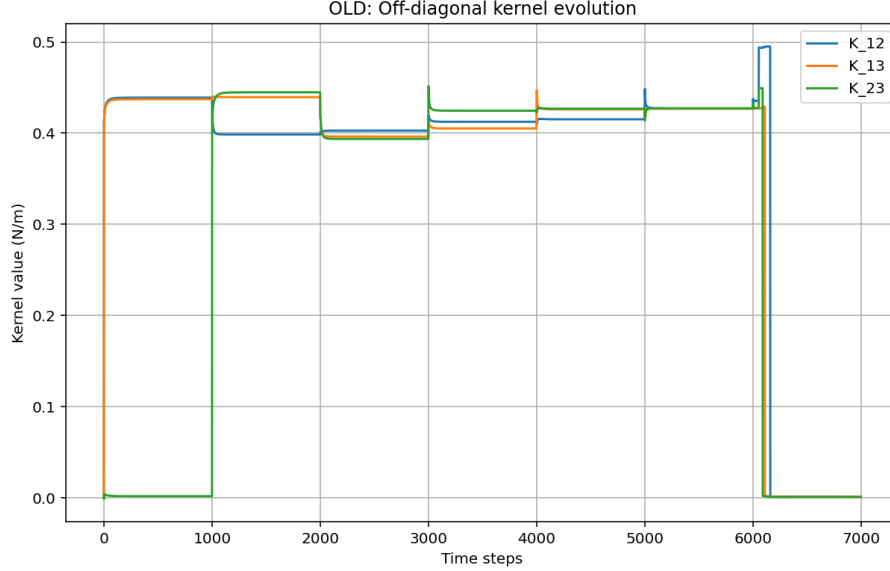


Figure 3.2: Collapse of off-diagonal kernel components ($\gamma_{12}, \gamma_{13}, \gamma_{23}$) after $120 \tau_p$ time steps

3.2.4 Data Processing and Kernel Library

For each position, geometry and domain size, the final 500 time steps of the extracted kernel were stored to file. A post-processing step then compiled all runs into a single dataset containing the kernel components, the particle position, the time index, and the corresponding velocities and forces. This library of memory kernels formed the training data for the machine learning surrogate described in the following section.

3.3 Predicting Memory Kernels through Machine Learning

The goal in this chapter is to replicate the results from the DNS framework described in previous chapters. The ML model predicts the memory kernel based on an input reference position, domain properties, and fluid properties τ_d . Validating this model is partially done by just comparing the predicted kernel to the DNS kernel. Further validation is done by comparing the actual particle trajectories by running a Lagrangian Particle Tracking simulation using the ML-predicted memory kernels and comparing it to the DNS results. An accurate model consistently replicates the hydrodynamics of the particle confined to the given geometry.

3.3.1 Data structure and processing

As stated previously, the hydrodynamic memory kernel $\mathbf{K}$ is a 3×3 tensor that quantifies how the velocity components at a past time τ contribute to the force component in each direction. For each system defined by its size and geometry, each position in that system is described by a tensor seen in equation 3.7.

$$\mathbf{K}(t) = \underbrace{\begin{bmatrix} K_{xx}^{(1)} & K_{xy}^{(1)} & K_{xz}^{(1)} \\ K_{yx}^{(1)} & K_{yy}^{(1)} & K_{yz}^{(1)} \\ K_{zx}^{(1)} & K_{zy}^{(1)} & K_{zz}^{(1)} \end{bmatrix}, \begin{bmatrix} K_{xx}^{(2)} & K_{xy}^{(2)} & K_{xz}^{(2)} \\ K_{yx}^{(2)} & K_{yy}^{(2)} & K_{yz}^{(2)} \\ K_{zx}^{(2)} & K_{zy}^{(2)} & K_{zz}^{(2)} \end{bmatrix}, \dots, \begin{bmatrix} K_{xx}^{(500)} & K_{xy}^{(500)} & K_{xz}^{(500)} \\ K_{yx}^{(500)} & K_{yy}^{(500)} & K_{yz}^{(500)} \\ K_{zx}^{(500)} & K_{zy}^{(500)} & K_{zz}^{(500)} \end{bmatrix}}_{\text{Sequence length is 500 steps}} \quad (3.7)$$

For a cubic geometry, where a particle is confined inside a cube, the distribution of positions is shown in figure 3.3. To save computation cost, the points do not cover the entire volume of the cube. Instead, they cover all possible cases, wall, edge and corner. Which are then reflected to cover the entirety of the geometry and do not need to be simulated. Each point has a Memory Kernel tensor $\mathbf{K}$ and velocity history $\mathbf{V}$, these points data are stored in a large file with positional-encoding and time-encoding.

Moreover, the Memory Kernel will be split up into peak and tail for numerous reasons. The peak describes the first few initial time steps of the Memory Kernel and are numerically much larger than the succeeding time steps. The tail of the Memory Kernel is significantly smaller and is defined as after the peak. In figure 3.4 the K_{zz} is plotted from the origin to the wall along the z-axis. In figure 3.4a on the left the Memory Kernels are plotted plainly, while in figure 3.4b the same Memory Kernels are plotted in logarithmic scale. In the figure, the early time steps 0-5 are defined as being or being part of the peak. While the succeeding and significantly smaller values are part of the tail. For visualizing and for further reasons, a log scale is used to increase the valuation of the tail during optimisation.

3.3.2 Network parameters and hyperparameters

For the training of the NN the Adam optimizer was chosen. It is both standard practice and well suited towards this project. The learning rate of the NN was set to 0.005 at first and was tested uniformly from ranges of 0.05 to 0.0005.

3.3.3 Long-short term memory

LSTMs were chosen for testing because of their ability to handle long and complex time dependencies well [48], which suits this project even if the data isn't a traditional time series. Figure 3.5 shows the sequential architecture of the model. The inputs consists of the position, domain size, and τ_d —passes through a sequence of NN layers to return a predicted memory kernel.

First, a `Dense(128)` layer uses a Leaky ReLU activation function. Leaky ReLU allows positive values to pass unchanged and sets negative values to near zero with a small ability to turn positive. This helps reduce the vanishing gradient problem,

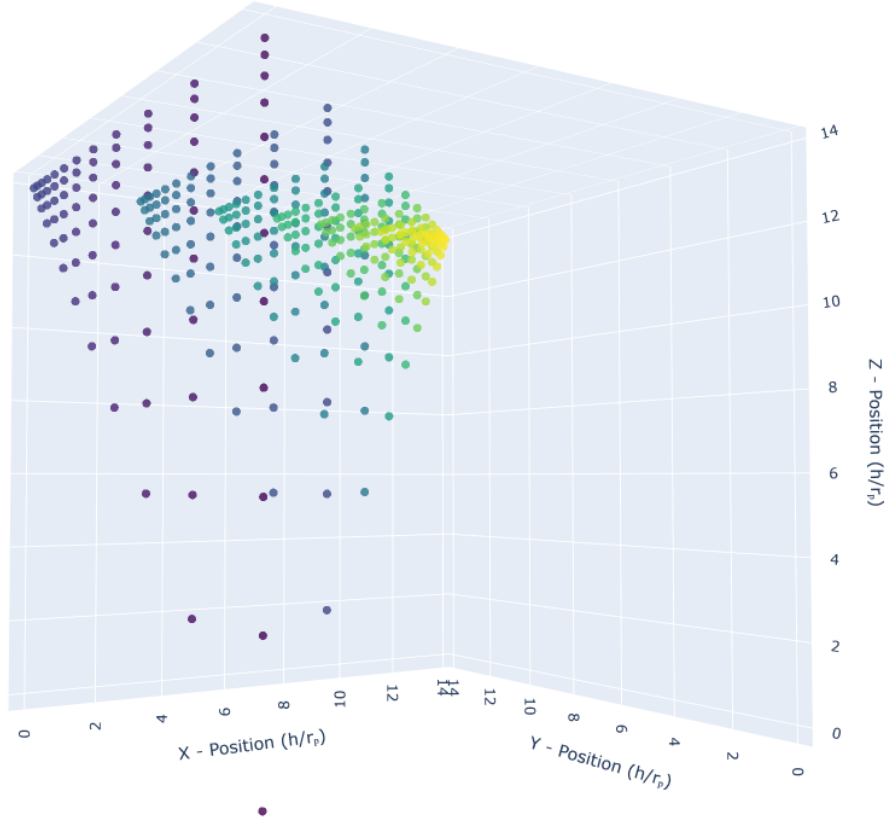


Figure 3.3: Diagram showing the distribution of points where DNS simulations are performed. The points positions start at the origin and are distributed such that they reach the wall, edge and corner of a subsection of the cube. The domain of the shown diagram is $30h/r_p$.

making the network to predict future time steps better and improving the fit for the kernel’s tail. Next, a `RepeatVector` layer makes 500 copies of the previous layer’s output so the network can predict all 500 time steps.

Following this, the data moves through a strictly sequential block of LSTM and Dropout layers: an `LSTM(256)`, a `Dropout(0.2)`, an `LSTM(128)`, and another `Dropout(0.2)`. Stacking two LSTM layers sequentially improves the model’s ability to learn patterns that evolve over long time periods. Finally, a `TimeDistributed` dense layer formats the output to match the shape of the kernel tensor.

3.3.4 Continuous-Time Decoder (MLP)

In figure 3.6 the model architecture for the Continuous-Time decoder. It is another form of NN used and tested throughout this project. By setting up the NN as will be shown below, a form of continuous functions of time estimates the kernel as an entire picture of a series. As previously seen, the Dense layer converts the input into a latent space and the `RepeatVector` duplicates this latent space to match the length of the memory kernels. The Lambda function concatenates a normalized time vector such that the time steps are time-encoded. The set of fully connected layers result

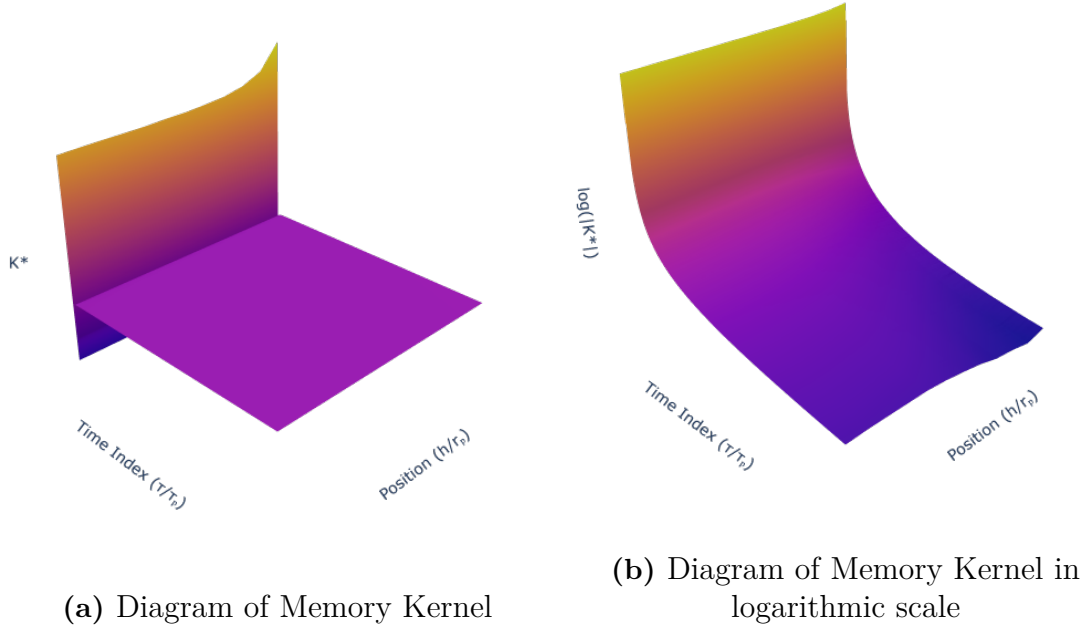


Figure 3.4: Diagram showing the importance of logarithmic scaling. The two sub diagrams 3.4a and 3.4b show Memory Kernels that start(left) at the origin and end(right) close to a wall. In plot 3.4a Memory Kernels are plotted plainly whilst in plot 3.4b is plotted in logarithmic scale.

in a deep neural network in the shape of a decoder. The width w determines in the width of the network as `Dense()` is a set of neurons. The `TimeDistributed()` function isolates each time step and passes each time step through the `Dense()` layers. This results in a continuous function which returns the output for a given time t , which was encoded using the `Lambda(inject_time)` function.

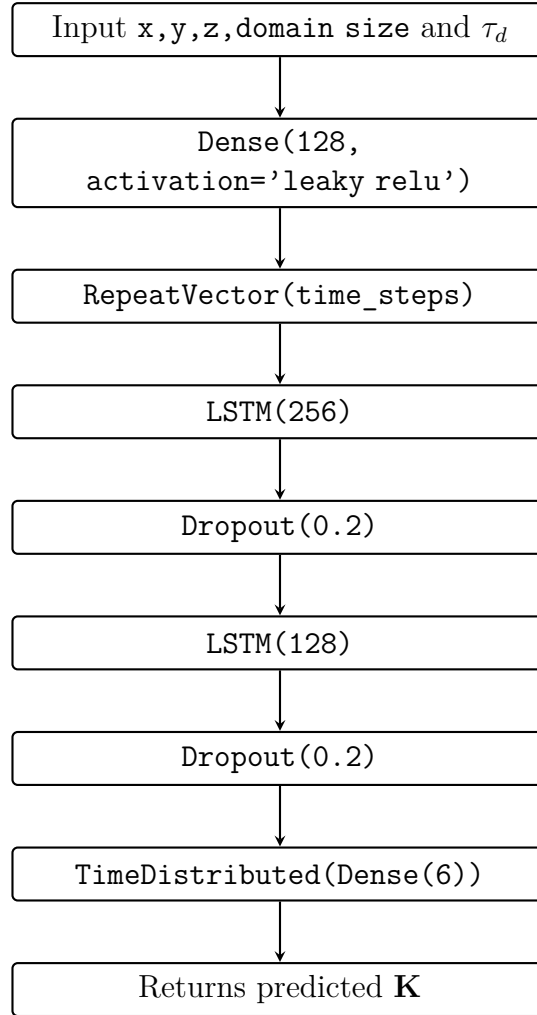


Figure 3.5: Flowchart of the LSTM network architecture. The sequence is expanded via `RepeatVector` and processed through deep LSTM layers utilizing orthogonal initializers, regularized by intermediate `Dropout` layers, and finalized with sequential `TimeDistributed` dense projections.

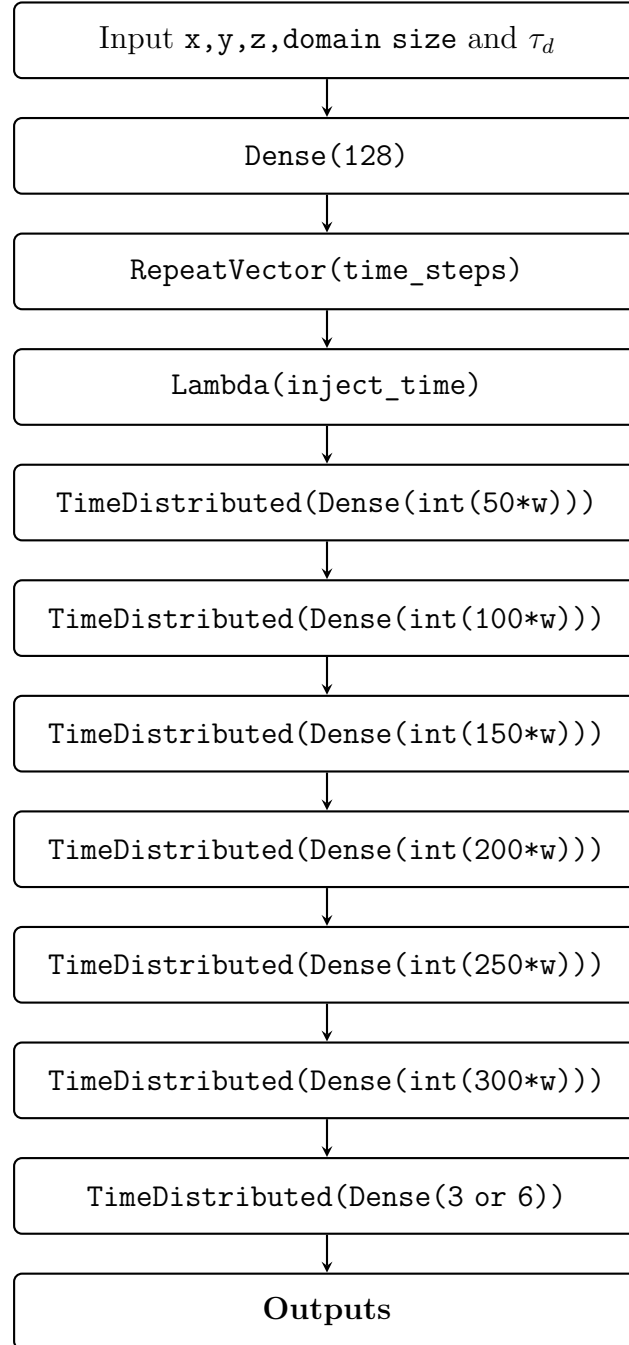


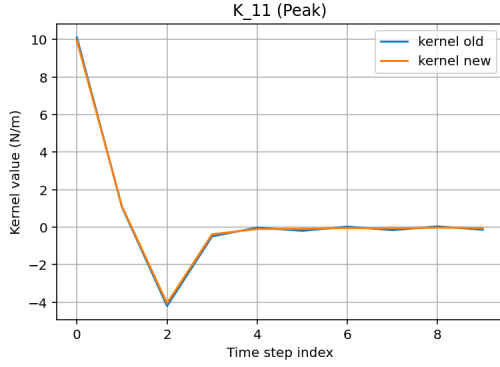
Figure 3.6: Flowchart of the Continuous-Time Coordinate MLP. By injecting the time coordinate t , the network maps initial spatial constraints to temporal outputs independently at each step, bypassing recurrent layers.

4

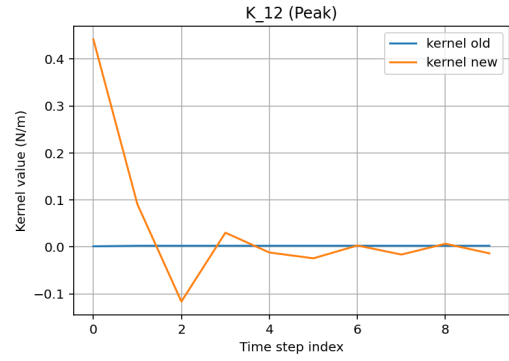
Results

4.1 Correction of Kernel Optimisation Instability

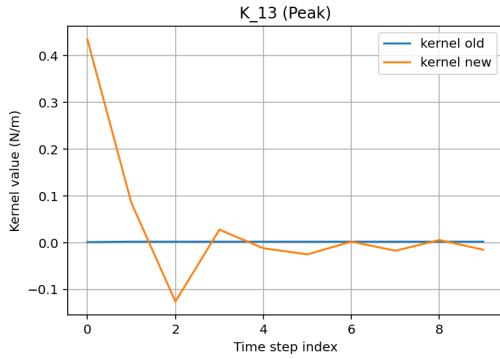
As described in the Methodology (Section 3.2.3.1), the original optimisation framework produced unphysical off-diagonal kernel components for highly confined positions when the forcing direction $[1, 1, 1]$ was applied. Figure 3.2 illustrates the collapse of K_{12} , K_{13} and K_{23} after approximately $120 \tau_p$ time steps.



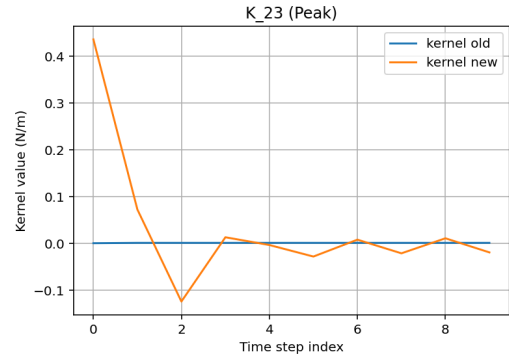
(a) K_{11} (diagonal): good agreement.



(b) K_{12} (off-diagonal): collapse.



(c) K_{13} (off-diagonal): collapse.



(d) K_{23} (off-diagonal): collapse.

Figure 4.1: Peak region (first 10 time steps) of the diagonal and off-diagonal kernel components obtained with the original (blue) and corrected (orange) optimisation methods. The diagonal component K_{11} agrees between the two methods, whereas the off-diagonal components collapse under the original method.

Figure 4.1 compares the first ten time steps (the peak region) of the diagonal and off-diagonal kernel components obtained with the original and corrected optimisation

methods. The diagonal components K_{11} , K_{22} and K_{33} agree well between the two methods. The off-diagonal components from the original method deviate markedly from the corrected ones, having collapsed to near zero, whereas the corrected components retain physically meaningful magnitudes and shapes.

Figures 4.2 and 4.3 show the evolution of the kernel components over the optimisation blocks for the two frameworks. The corrected optimisation remains stable throughout, with the off-diagonal terms converging to non-zero values that are consistent across the entire run. In contrast, the original off-diagonal terms (K_{12} , K_{13} , K_{23}) drop to zero after roughly 120 τ_p time steps and never recover, while the diagonal components remain largely unchanged between the two methods. This confirms that the instability affected only the cross-coupling terms.

The remedy, described in Section 3.2.3.1, was to reverse the sign of every forcing direction. This removed the collapse entirely, and all subsequent kernel extractions used the corrected forcing sequence, providing a physically consistent library of memory kernels for the machine learning surrogate.

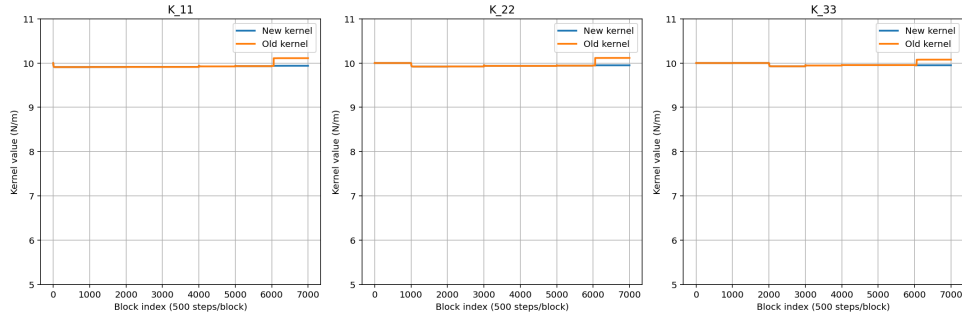


Figure 4.2: Evolution of the diagonal kernel components (K_{11} , K_{22} , K_{33}) over the optimisation blocks.

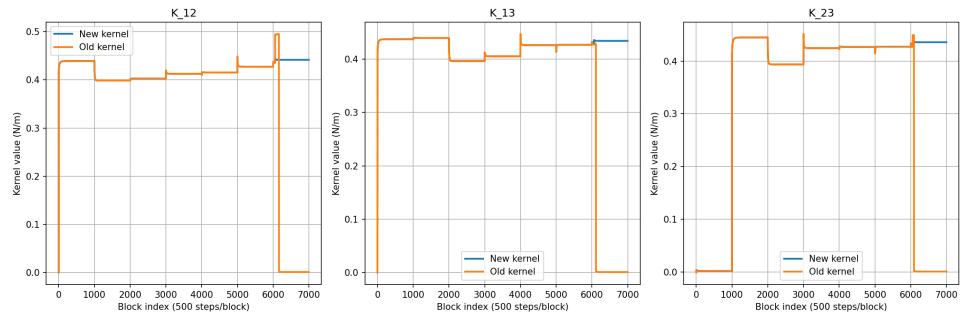


Figure 4.3: Evolution of the off-diagonal kernel components (K_{12} , K_{13} , K_{23}) over the optimisation blocks.

4.2 Hindrance Effects

4.2.1 Characteristics of the memory kernels

4.2.1.1 General Shape and Trends

As discussed in Section 2.1.4 (Theory), the peak of the memory kernel is associated with drag and added mass effects, while the tail reflects the decay of hydrodynamic memory. To characterise how hindrance modifies the kernel, its components were evaluated at a sequence of positions running from the domain centre towards the confining boundary. The kernels obtained at the different particle positions were stitched together to construct a three-dimensional kernel map describing the spatial variation of the hydrodynamic response. As shown in Figure 4.4, the kernel peaks increase as the particle approaches the wall, indicating an increase in both drag and added mass. This behaviour is consistent with the physical expectation that these effects intensify near a confining boundary and agrees with the behaviour reported by Michael et al. [25], [26].

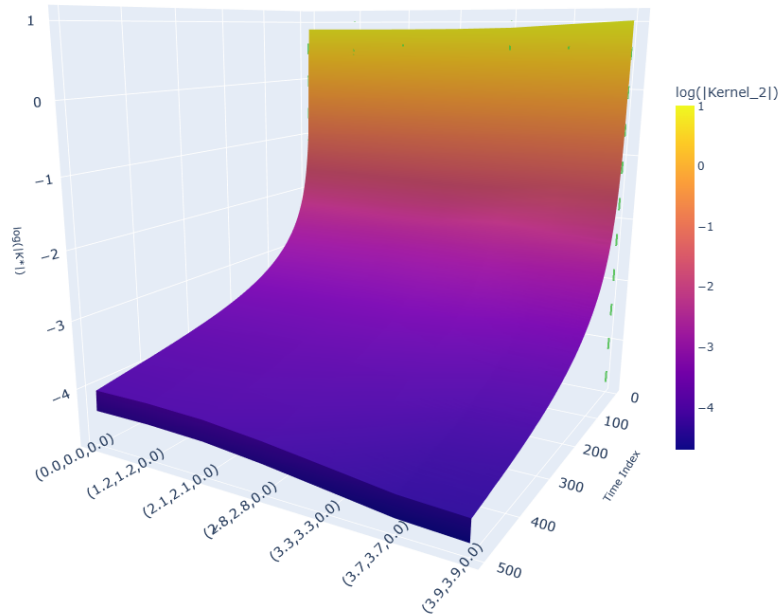


Figure 4.4: Surface plot of $\log_{10}(|K_{33}|)$ for the square channel at $10r_p$ domain. Time index on the x-axis, Positions are along the diagonal from centre to wall on the y-axis.

4.2.2 Approach to a Wall

We first compare the diagonal components of the memory kernel for particle positions moving from the centre of the domain towards a single wall. Three geometries are considered: a two-wall configuration (slab, confined only in the direction perpendicular to the walls), a square channel (confined in two orthogonal directions), and a cube (confined in all three directions). In each case the particle approaches the wall along a line of sampled positions, as illustrated in Figure 4.5 for the cube.

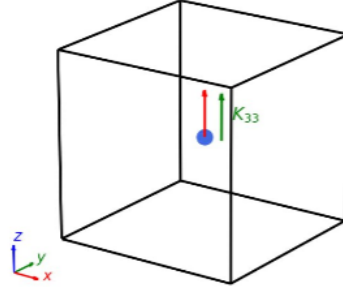


Figure 4.5: Sampling direction for the wall-approach case, shown for the cube.

The particle starts at the domain centre and is moved towards the wall (red arrow). The kernel component reported is the diagonal component aligned with this direction, here K_{33} (green arrow).

For each geometry we report the diagonal component aligned with the direction of motion. The wall-normal direction is taken as z for the cube and two-wall cases and as y for the square channel, so that the relevant components are K_{33} , K_{33} and K_{22} respectively. The three surfaces, for the $10r_p$ domain in which the particle starts at the centre ($5r_p$ from each wall), are shown in Figure 4.6.

In Figure 4.6, distinct valleys, seen as sharp downward dips in the log-magnitude, appear in the square channel and cube geometries as the particle approaches the wall. These valleys coincide with the dashed green lines marking sign changes of the kernel, where the contribution of the earlier velocity to the present hydrodynamic force reverses. No such valleys appear in the two-wall slab. For both the square channel and the cube, two sign changes occur at positions close to the wall, indicating that the reversal is a signature of multi-directional confinement: two or more orthogonal walls reshape the hydrodynamic memory so that the kernel changes sign as the particle nears the wall, a feature that a single pair of parallel walls does not produce.

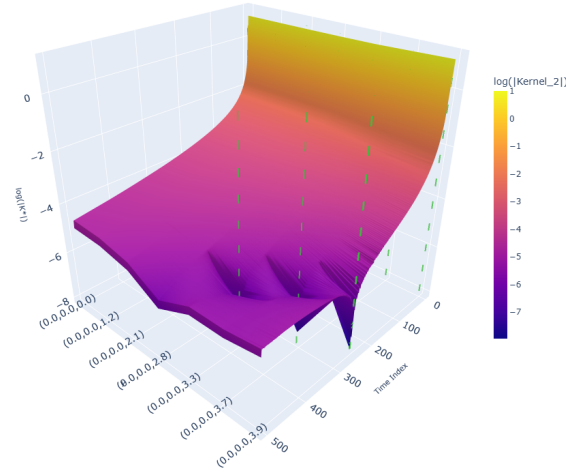
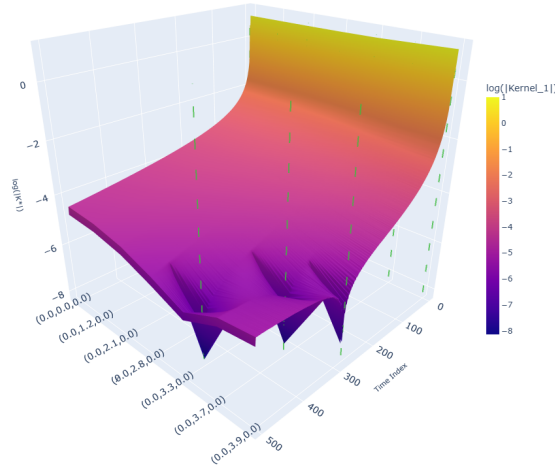
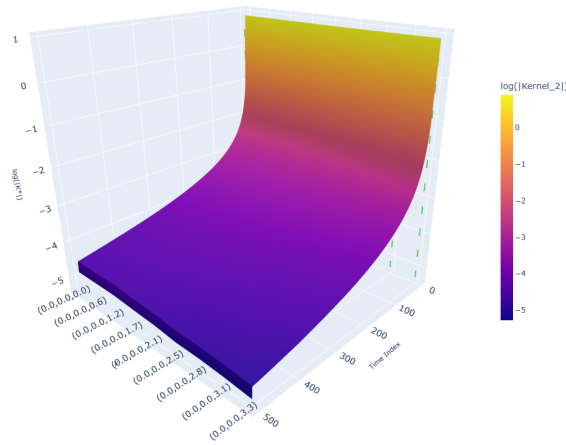
(a) Cube, $10r_p$ domain, K_{33} (b) Square channel, $10r_p$ domain, K_{22} (c) Two-wall, $10r_p$ domain, K_{33}

Figure 4.6: Surface plots of $\log_{10}(|K|)$ for the wall-normal diagonal component, K_{33} for the cube, K_{22} for the square channel, and K_{33} for the two-wall geometry. The time-lag index is on the x -axis and the sampled position on the y -axis, and the dashed green lines mark sign changes of the kernel.

4.2.3 Approach to an Edge

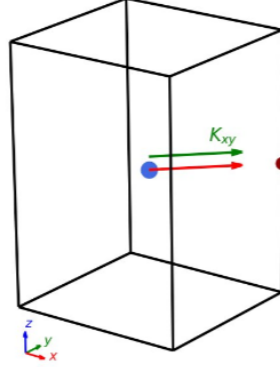


Figure 4.7: Sampling direction for the edge-approach case, shown for the square channel. The kernel is sampled at successive positions running from the domain centre towards the edge (red arrow), where two walls meet, and the green arrow labelled K_{xy} indicates the off-diagonal component aligned with this direction.

We now consider how the memory kernel varies as the particle is positioned progressively closer to an edge, the line along which two walls intersect, in the square channel and the cube. The kernel was extracted at a sequence of static positions lying along the diagonal of the confining plane, running from the domain centre towards the edge. This sampling direction is illustrated in Figure 4.7 for the square channel. In the square channel the sampled positions lie along the diagonal of the x - y plane, and in the cube they run towards an edge formed by two orthogonal walls. For each geometry we report the diagonal component associated with the sampling direction, K_{11} for the square channel and K_{22} for the cube, together with the corresponding off-diagonal components aligned with this direction, K_{12} and K_{23} respectively.

Several trends become apparent in Figure 4.8. In the diagonal components K_{11} and K_{22} , sign changes appear once the sampled position is close to the edge, much as they do in the wall-approach case. The off-diagonal components K_{12} and K_{23} behave differently, since they show sign changes at every sampled position from the centre out to the edge, and their peak magnitude grows sharply as the edge is approached. The same overall pattern holds in both cases, because as the sampled position moves closer to the edge and the confinement strengthens, the sign change shifts to earlier times so that the reversal happens sooner in the evolution of the kernel.

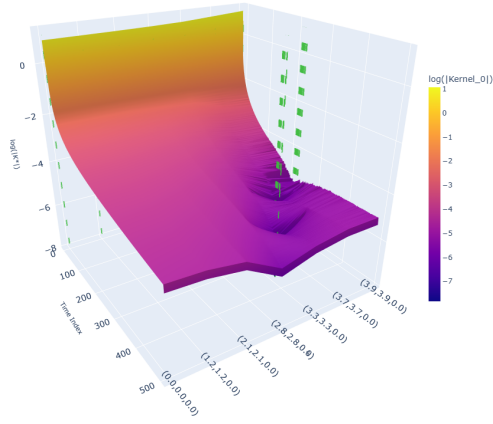
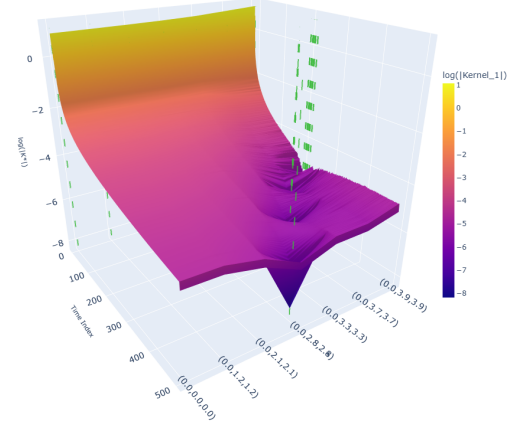
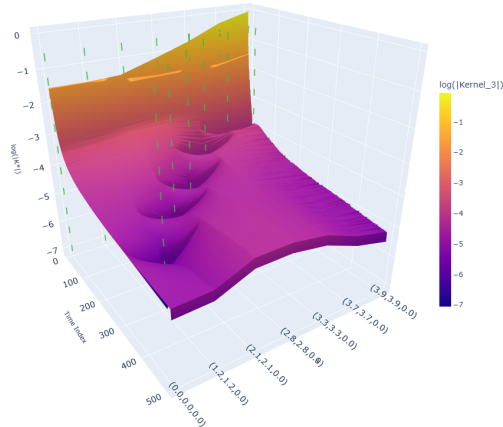
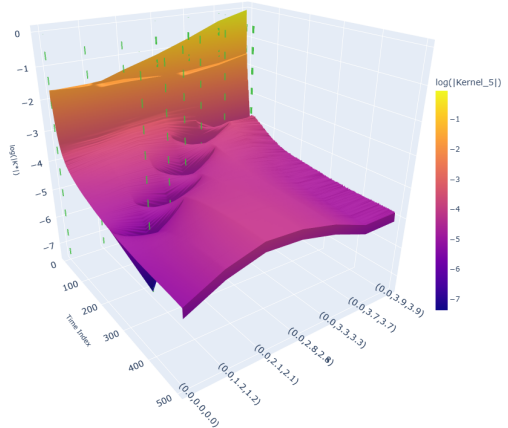
(a) Square channel: K_{11} (diagonal)(b) Cube: K_{22} (diagonal)(c) Square channel: K_{12}
(off-diagonal)(d) Cube: K_{23} (off-diagonal)

Figure 4.8: Diagonal and off-diagonal kernel surfaces for the edge-approach sampling in the $10r_p$ domain: K_{11} (square channel) and K_{22} (cube) on the top row, and the off-diagonal components K_{12} (square channel) and K_{23} (cube) on the bottom row. The y -axis is the sampled position along the diagonal from the centre to the edge, and the x -axis is the time-lag index. Dashed green lines mark sign changes.

4.2.4 Motion Parallel to a Wall

Figure 4.10 shows how the kernel behaves when the particle is sampled along a line at constant distance from one wall while its distance to the orthogonal wall decreases. The direction is illustrated for the cube in Figure 4.9, where the particle sits at the centre of the z -wall and the sampling proceeds towards the y -wall along the red arrow, while the green arrow labelled K_{33} marks the diagonal component in the z direction. For the square channel the sampling runs in the x direction along a line parallel to the wall normal to y , and we report K_{22} , the diagonal component in the y direction, together with K_{12} , the off-diagonal component coupling x and y . For the cube the sampling runs in the y direction parallel to the wall normal to z , and we report K_{33} , the diagonal component in the z direction, together with K_{23} , the off-diagonal component coupling y and z . In both cases the particle therefore stays a fixed distance from one wall while being sampled progressively closer to a second, orthogonal wall, so that the confinement it feels grows in only one direction as the sampling proceeds.

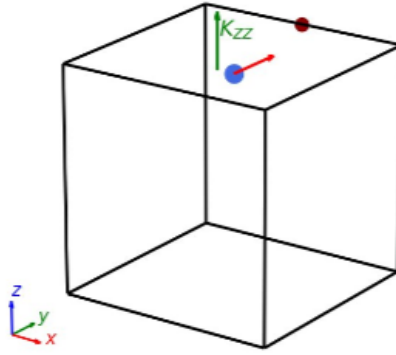


Figure 4.9: Sampling direction for the parallel-motion case, shown for the cube, where the particle sits at the centre of the z -wall and the sampling proceeds towards the y -wall along the red arrow, while the green arrow labelled K_{33} marks the diagonal component in the z direction that stays perpendicular to the sampling direction.

In the diagonal components K_{22} for the square channel and K_{33} for the cube, the sign change appears at the same temporal location regardless of how far the sampled position has moved, because the plotted component is perpendicular to the sampling direction and the distance to the wall that hinders it stays constant, so the sign change is governed only by that fixed wall distance.

The off-diagonal components K_{12} and K_{23} behave differently, since they are aligned with the sampling direction, and as the sampled position approaches the orthogonal wall the confinement in that direction strengthens, thus showing that the sign change occurs earlier and the valley shifts towards smaller time-lag indices. The off-diagonal kernels also show a sharp increase in peak magnitude near the wall, much

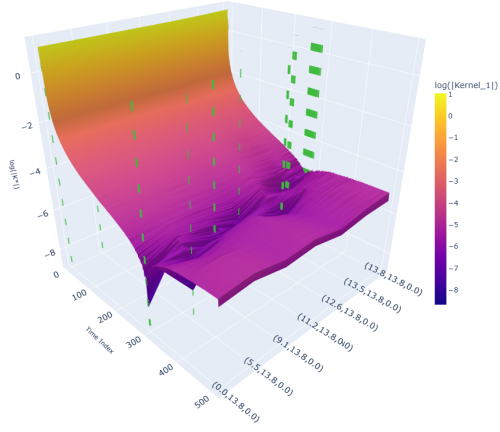
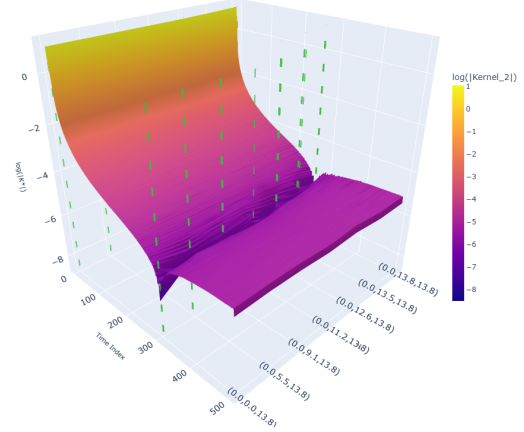
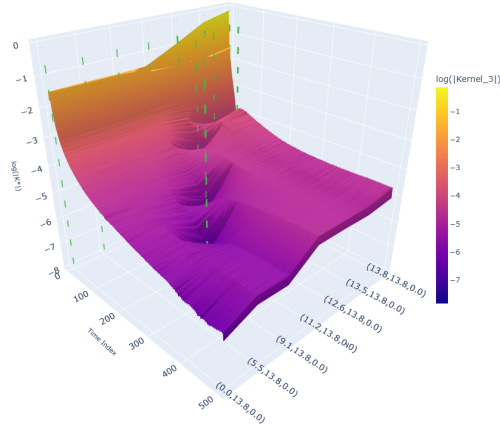
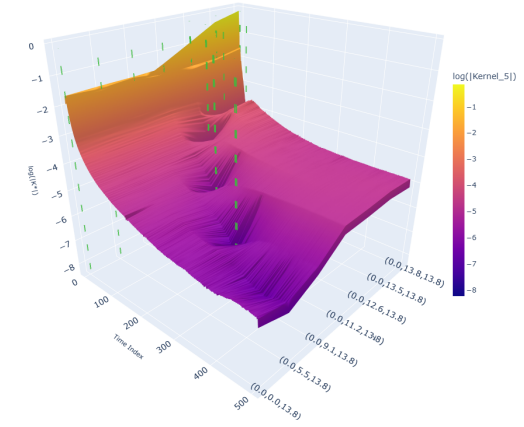
(a) Square channel, K_{22} (diagonal)(b) Cube, K_{33} (diagonal)(c) Square channel, K_{12}
(off-diagonal)(d) Cube, K_{23} (off-diagonal)

Figure 4.10: Diagonal and off-diagonal kernel surfaces for motion parallel to a wall in the $15r_p$ domain, showing K_{22} for the square channel and K_{33} for the cube on the top row and the off-diagonal components K_{12} and K_{23} on the bottom row, where the x -axis is the sampled position along the line and the y -axis is the time-lag index. Dashed green lines mark sign changes.

as in the edge-approach case, which reinforces the earlier pattern where stronger confinement makes the sign reversal happen sooner and amplifies the kernel peak in the component aligned with the direction of increasing hindrance.

4.2.5 Visualisation of Zero Crossings

To confirm the location of the sign changes in the memory kernel, we construct surface plots in which the height represents $\log_{10}(|K|)$ and the surface colour encodes the sign of the kernel value K . Positive values are shown in red and negative values in blue, so that a zero crossing appears as a sharp transition from blue to red through a white band. Combining the logarithmic magnitude with the sign colour provides an unambiguous way to confirm that the observed valleys correspond to genuine zero crossings rather than numerical artifacts.

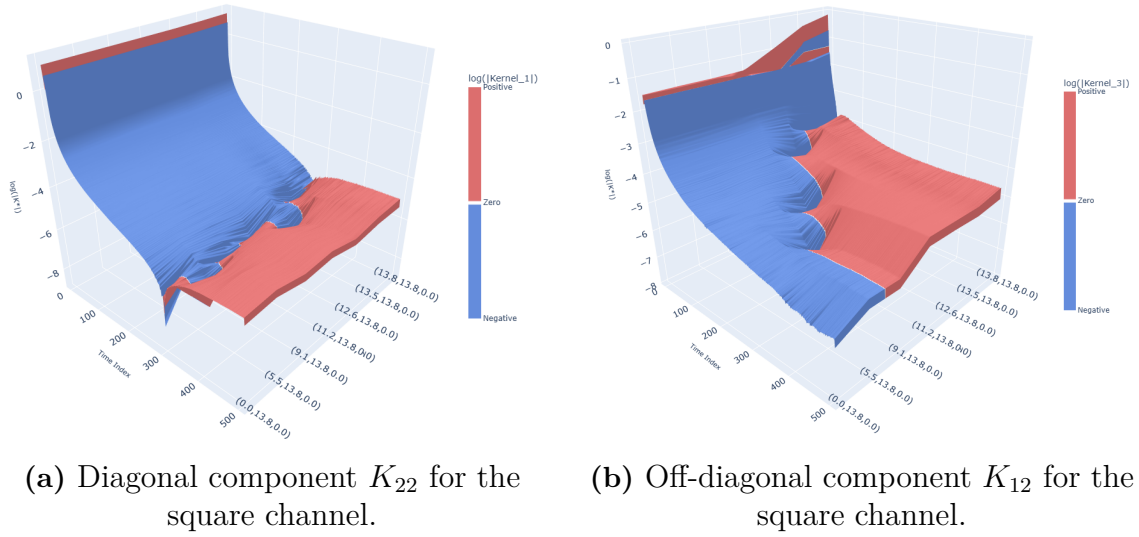


Figure 4.11: Sign-coloured surface plots of the square-channel memory kernel components. The height shows $\log_{10}(|K|)$ and the colour encodes the sign of K : red is positive, blue is negative, and the white band marks the zero crossing.

The sign of the kernel has a direct physical meaning that follows from the way it enters the hydrodynamic force in Equation (2.13), where the force is given by minus the convolution of the kernel with the velocity history. A positive kernel value at a given lag therefore means that the particle's velocity at that earlier instant contributes a force opposing its present motion, which is the familiar sense of viscous drag in which the fluid resists the particle. A negative value reverses this, so that the earlier motion instead contributes a force in the same direction as the present velocity, meaning the fluid returns momentum to the particle rather than resisting it. The sign change captured in Figure 4.11 therefore marks the lag at which the hydrodynamic memory switches from resisting the particle to reinforcing its earlier motion. Why confinement brings about these reversals, and why they strengthen as the particle nears an edge or corner, is not addressed here and remains an interesting question for further study.

4.2.6 Verification and Validation

To validate the reduced-order model, we compare the particle trajectory obtained from full DNS against that from an LPT simulation driven by the precomputed memory-kernel library, with interpolation for off-grid positions. Figure 4.12 shows the particle position over time for both methods in the $10r_p$ cube. The two trajectories follow one another closely, though the LPT particle drifts towards the wall slightly faster than the DNS particle and therefore reaches it, and ends, a little earlier.

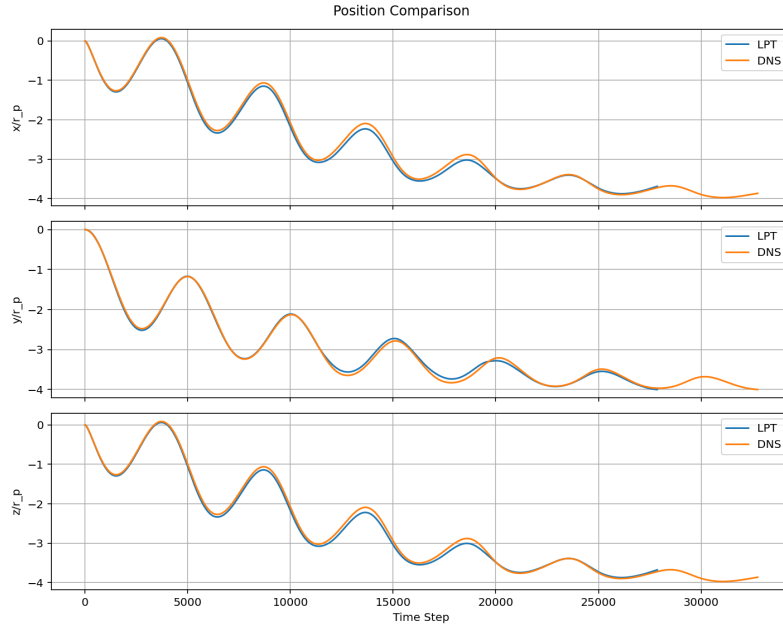


Figure 4.12: Particle position over time from DNS (orange, dashed) and LPT (blue, solid) for the $10r_p$ cube. The LPT trajectory ends earlier because the particle reaches the wall sooner.

The two trajectories agree closely over the course of the run, with a maximum position difference of $0.139r_p$, less than fourteen percent of a particle radius. The LPT particle reaches the wall slightly sooner than in the full DNS, and reducing this difference is one of the improvements identified for future work. Overall the memory-kernel library combined with interpolation reproduces the confined hydrodynamics to good accuracy, which supports its use as a reduced-order model for confined Langevin dynamics.

4.3 Machine Learning

4.3.1 Choice of Machine Learning model architecture

When selecting the best performing models for predicting the memory kernels, two key performance indicators are used. Firstly, the Mean Squared Error and logarithmic loss over the memory kernel are used as indicators of performance. Secondly, the models are compared directly with the test data for different spatial positions.

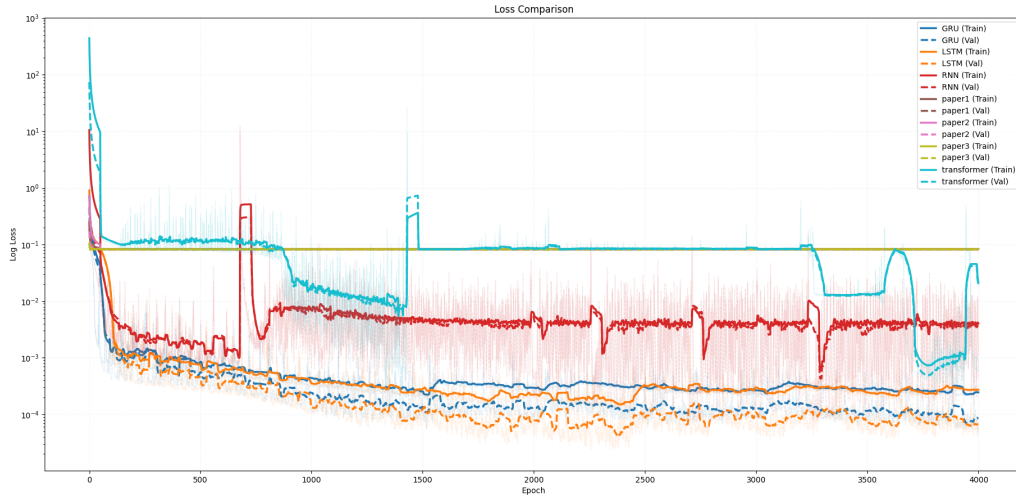


Figure 4.13: Comparison between different types of model architectures. The vertical axis shows the Mean Square Error for the during of training. The horizontal axis shows the number of epochs.

The MSE and logarithmic loss are both used to determine if the model is able to replicate the peaks and tails of the memory kernels. This is due to the fact that numerically the peaks are much larger, by a factor of 10^6 than the tails and always dominate the MSE terms. The logarithmic loss functions helps reduce this loss. In figure 4.13, a large set of machine learning models of similar parameter sizes are compared. Of the presented models, the GRU and LSTM perform the best, as they maintain the lowest MSE between the training/validation data kernels and the predicted kernels, throughout the training process.

4.3.2 Memory Kernels across geometry

To verify that the ML model is able to replicate the simulation data, the accuracy and the consistency of models was tested throughout development. In this project a good ML model replicates the Memory Kernels throughout different spatial positions for different geometry domains.

The best performing model thus far is the LSTM architecture. Through a hyper parameter search, the size of the network mentioned in the Method chapter was concluded to be best. This was done on a range from 64 to 258 neurons for the Dense layer and for the LSTM layers it was ranged from 64 to 516 neurons. This was done to balance the effects of overfitting and underfitting. Similarly, other parameters such as the dropout and learning rate were also testing in a range which ranged from a factor of 10 multiplied and divided by the found optima. Finally, to improve the models complete fit of the entire kernel time steps, the model was split into two models, which were trained on different parts of the kernel. This was used to tackle the numerical variety found in the data set, the memory kernel values reached from 10 to 10^{-6} . This helped the model fit the memory kernel more uniformly. The best so far determined split of the memory kernel training was found

at 50 time steps, where the two differently trained models are unified into a singular model.

In figures 4.14, 4.15, and 4.16, the machine learning model has generated memory kernels for a trajectory extending from the origin to the domain corner. The different graphs contain different elements of the memory kernels, where the height of the data displays the absolute values of the kernel on a logarithmic scale. Furthermore, the 500 time steps of the kernel are displayed for different positions moving along the diagonal.

In these results, the peaks and tails themselves are well-fitted when compared to the baseline data for approaching a corner. The sharp downward dips appearing in the data are not actual physical dips, but rather indicators of a sign change; values appearing after a dip are positive. This effect is consistent with the DNS data from previous chapters. Between the peak and the tail, at approximately time step 50, the models are unified, which accounts for the localized noise observed where the peak rises.

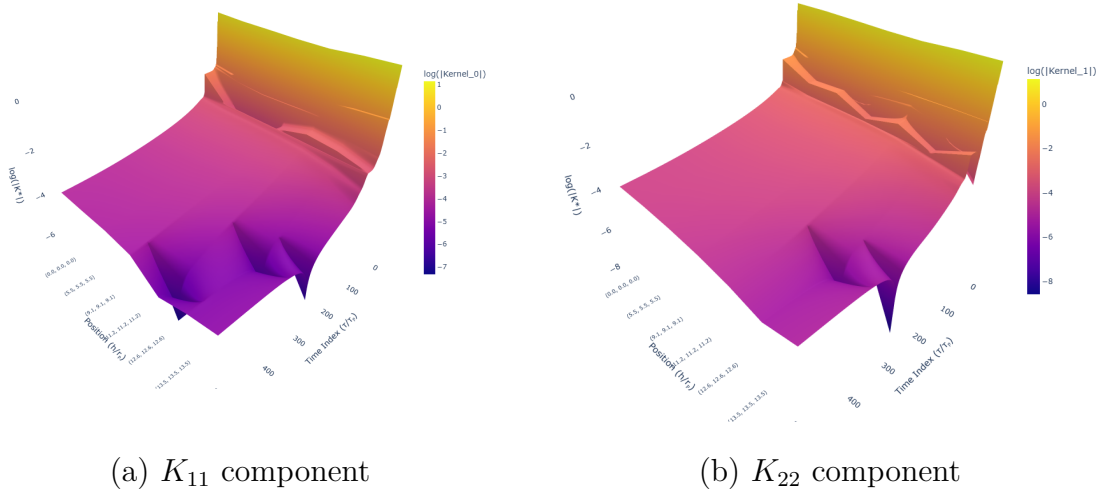


Figure 4.14: The principal diagonal components K_{11} (xx) and K_{22} (yy) of the memory kernel. The magnitudes are evaluated along a diagonal trajectory extending from the origin to the domain corner.

4.3.2.1 Physics-Informed Neural Network Results

While Physics-Informed Neural Networks have led to performance improvements in many PDE applications[49], their use in this project primarily resulted in faster convergence rather than a significantly lower final error. In figure 4.17, the influence of PINNs can be seen during training. The leftmost curve depicts an equal weight balance between the Residual loss term (Equation 2.20) $\mathcal{L}_{\text{Residual}}$ and the loss term arising from the hydrodynamic effects, $\mathcal{L}_{\text{History}}$.

The other two curves show an imbalance between the terms, where the hyperparameters λ_1 and λ_2 are separated by a factor of 10. The results show that the PINN model is capable of learning the system's behavior more quickly early on. However, the final trained ML model with Physics Informed learning showed a significantly

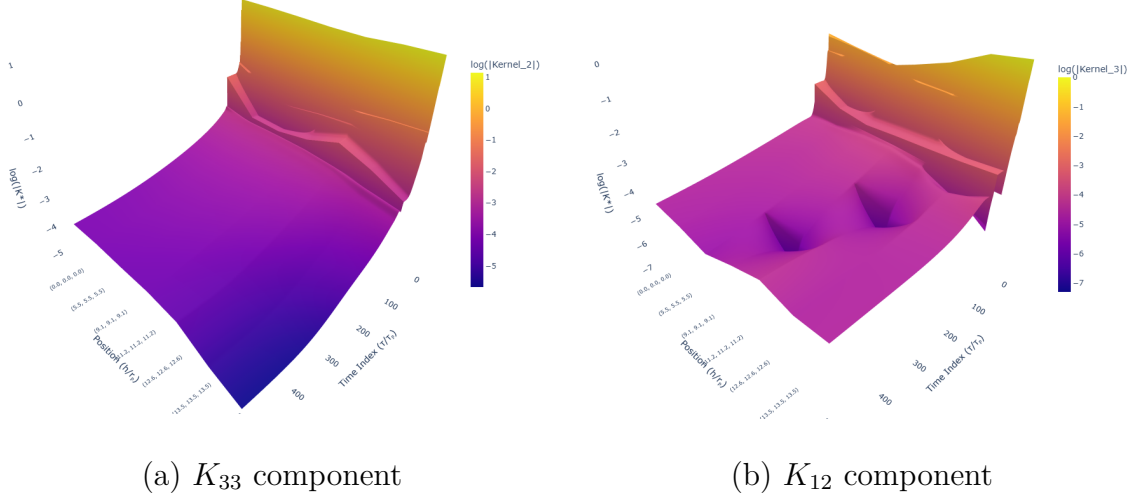


Figure 4.15: The longitudinal component K_{33} (zz) and the first transverse cross-coupling component K_{12} (xy). Both are plotted as a function of the time index and spatial position along the diagonal.

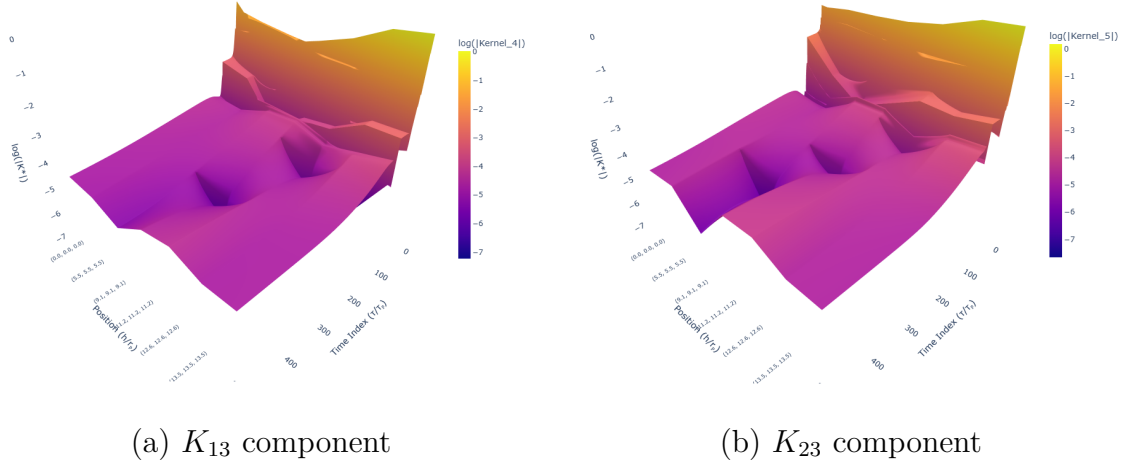


Figure 4.16: The remaining transverse cross-coupling memory kernel components, K_{13} (xz) and K_{23} (yz), representing the interactions between the z-axis and the orthogonal plane along the trajectory.

worse fit. The added complexity of PINNs did not improve the models ability to replicate DNS trajectories.

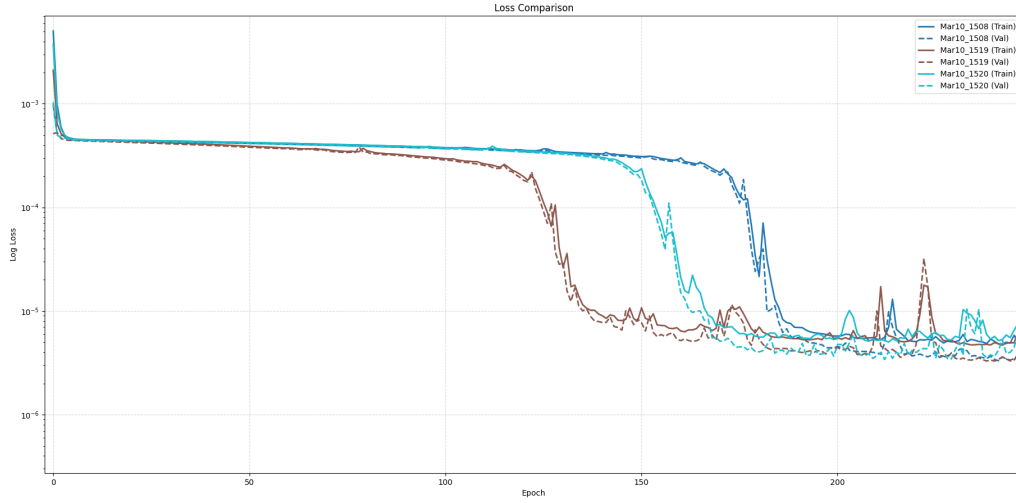


Figure 4.17: In this diagram the MSE loss is compared by the extent the PINN loss term $\mathcal{L}_{\text{History}}$ for a LSTM based Machine Learning model. The figure shows three lines, brown depicting $\lambda_2 = 1$, cyan depicting $\lambda_2 = 0.1$ and navy showing $\lambda_2 = 10$, differently weighted $\mathcal{L}_{\text{History}}$ terms. For all training sets, the Residual loss term hyperparameter $\lambda_1 = 1$ remained constant.

4.3.3 Validation of particle trajectory

The primary goal of optimizing the ML model is to reduce the error between the predicted kernel and the ground truth. This section compares a Direct Numerical Simulation (DNS) with Lagrangian Particle Tracking (LPT) utilizing the ML-estimated kernels. To test the validity of the memory kernels, the particle is driven by a steady force into a spiral pattern, which introduces more complexity and forces contributions from the full memory kernel.

The DNS is performed using the theory and methods explained in preceding chapters. The LPT simulation uses memory kernels generated by the LSTM model at each time step. For the sake of a clear comparison, the Brownian autocorrelation force was omitted here, as it introduces randomness that makes direct trajectory comparison more difficult.

In figure 4.18, the DNS is compared to the ML-assisted LPT for a particle traveling toward a corner in a spiral-like trajectory. The positions are shown across three graphs for the X, Y, and Z axes. The paths of the two simulations match reasonably well, following each other closely for the majority of the trajectory. The Y position shows a slightly larger deviation between the two methods compared to X and Z, though the offset remains relatively small.

4. Results

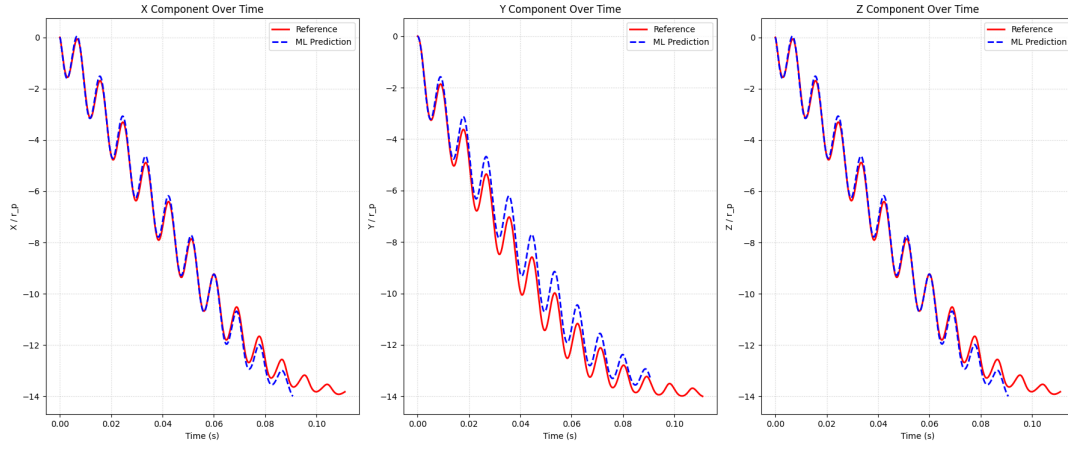


Figure 4.18: Comparison of particle position over time from DNS (solid red line) and LPT (dashed blue line). The diagram displayed the X, Y and Z positions over time. The LPT trajectory ends earlier because it reached the wall earlier, which ends the simulation.

5

Conclusion

This thesis built directly on the memory kernel extraction framework established by Michael et al. [25], [53] and systematically investigated how geometric confinement modifies hydrodynamic memory. The replacement of expensive direct numerical simulations (DNS) by a machine learning surrogate for predicting memory kernels was also explored.

5.1 Development of memory kernels

In this work, we first corrected the memory kernel optimization framework. The original method produced unphysical collapses of off-diagonal kernel components for highly confined positions under the $[1, 1, 1]$ forcing direction. Reversing the sign of all forcing directions eliminated this artifact entirely, yielding stable and physically consistent off-diagonal kernels while leaving the diagonal components unaffected. With a stable framework in place, the confinement study showed that hydrodynamic memory depends strongly on the number of orthogonal walls surrounding the particle. Approaching a single wall increased the diagonal kernel peaks, which is consistent with the expected growth of drag and added mass near a boundary. More strikingly, sign-change valleys in the kernel appeared only in the square-channel and cube geometries and were absent in the two-wall slab, showing that the reversal is a signature of multi-directional confinement rather than of proximity to a wall alone. For approaches to an edge the off-diagonal components changed sign at every sampled position and their peaks grew sharply near the edge, whereas for sampling parallel to a wall the sign-change location of the perpendicular component stayed fixed while that of the aligned component shifted to earlier lags as the confinement strengthened. Taken together these results indicate that the location of the sign change is set by the wall distance in the direction of the kernel component considered.

These sign changes are, to our knowledge, a previously uncharacterised feature of the confined memory kernel, and they give a kernel-level view of the wall-modified hydrodynamic memory that earlier studies inferred indirectly from the confined velocity autocorrelation function. Why confinement produces these reversals, and why they strengthen towards edges and corners, is not resolved here and remains an open question that the present kernels are well suited to explore.

5.2 Machine Learning Surrogate

The validity of the developed model is based on its ability to generate accurate, consistent memory kernels and, more importantly, to replicate direct numerical simulation trajectories. Among the tested configurations, the LSTM-based architecture paired with a logarithmic loss function gave the best performance. The results demonstrate that the machine learning model is able to closely replicate the DNS results, as seen in Figure 4.18. The similarity among the particle trajectories confirms that the predicted memory kernels capture the underlying hydrodynamics of the particle. While there is room for optimisation and further testing, this successful replication serves as a proof of concept.

An LSTM model split into two parts during training successfully reproduced the diagonal and off-diagonal kernel surfaces along a diagonal trajectory, capturing the peaks, tails, and sign changes. Physics-informed neural networks resulted in faster learning but did not improve the model’s final performance. A spiral trajectory test using LPT with the ML-predicted kernels matched the DNS well, showing that the surrogate model can successfully replicate complex particle dynamics.

5.3 Validation and Limitations

The reduced-order Lagrangian particle tracking (LPT) model, driven by the pre-computed kernel library with interpolation, reproduced the full-DNS trajectory in the $10r_p$ cube to within a maximum position difference of $0.139r_p$, less than fourteen percent of a particle radius. The LPT particle reached the wall slightly sooner than in the DNS, and closing this gap is one of the improvements identified below. Validation was carried out for a single geometry at a neutrally buoyant density ratio, so the main limitations of the present work are that the kernel library is tied to specific discrete geometries and to one density ratio, and that its accuracy is weakest close to the walls where the kernel varies most rapidly. On the machine learning side, there appears noise and fluctuations near corners and edges in the dataset due to increasing complexity and near-zero data, which leads to the network fitting the noise during training.

5.4 Future Outlook

On the kernel side, several extensions would build directly on the present work. The optimisation framework used to fit the memory kernels could be improved further, both to sharpen the accuracy of the extracted kernels and to reduce the instabilities that appeared for the most strongly confined positions. Denser sampling of positions near the walls would resolve the rapid near-wall variation of the kernel and reduce the wall-ward bias seen in the LPT trajectories, while extending the extraction across a range of particle-to-fluid density ratios would broaden the applicability of the library. The study could also be extended to a wider range of confining geometries beyond the cube and square channel considered here, including genuinely arbitrary geometries rather than the straight-line sampling paths used in this work, and to rotational

Brownian motion through the corresponding angular-velocity kernels. Finally, the sign-change behaviour uncovered in this work invites a dedicated study of its physical origin, which the extracted kernels are well placed to support.

On the machine learning side, future work should focus on generalising the model for different physical geometries. While we took initial steps in this direction, time constraints limited full training and analysis. The key is changing the model's inputs, shifting the coordinate reference system from the distance to the origin to the distance to the nearest wall. This doubles the input coordinates but decouples the model from any specific spatial setup. Once trained on this generalised approach, the model could act as a stronger predictive tool to estimate memory kernels and map hydrodynamics in new, complex geometries where regular DNS simulations are complex and expensive.

To achieve this, a few technical improvements could help enhance the model fitting. First, the network's activation functions should be re-evaluated. The current setup uses Leaky ReLU, which transforms weights linearly and zeroes out negative values, and testing alternative activation functions might capture more complex dynamics. Second, the loss function can be refined. The current method calculates the logarithm of the absolute difference, due to values spanning a large interval of positive and negative numbers, and using a more complete logarithmic loss, taking the difference of the logarithms directly, could improve training stability and better penalise errors. Beyond the network architecture, a larger, more diverse dataset will prevent the neural network from learning a bias towards a coordinate or geometry and will lessen the overfitting of near-corner noise, and with a larger dataset and an updated architecture, a thorough study of network depth and hyperparameter tuning will maximise the model's performance.

Summary

Overall, this work provided a robust framework for extracting memory kernels from DNS, quantified how confinement alters hydrodynamic memory across three geometries, and demonstrated that a machine learning surrogate can replace expensive DNS for predicting memory kernels and particle trajectories in confined microfluidic environments.

References

- [1] A. J. deMello, “Control and detection of chemical reactions in microfluidic systems,” *Nature*, vol. 442, pp. 394–402, Jul. 2006.
- [2] E. Frey and K. Kroy, “Brownian motion: A paradigm of soft matter and biological physics,” *Annalen der Physik*, vol. 517, no. 1-3, pp. 20–50, 2005.
- [3] A. Jeyasountharan, G. D’Avino, and F. Del Giudice, “Confinement effect on the viscoelastic particle ordering in microfluidic flows: Numerical simulations and experiments,” *Physics of Fluids*, vol. 34, no. 4, p. 042015, Apr. 2022. DOI: 10.1063/5.0090997.
- [4] S. Prakash and J. Yeom, *Nanofluidics and microfluidics: systems and applications*. William Andrew, 2014.
- [5] P. Hänggi and F. Marchesoni, “Artificial brownian motors: Controlling transport on the nanoscale,” *Rev. Mod. Phys.*, vol. 81, pp. 387–442, 1 Mar. 2009. DOI: 10.1103/RevModPhys.81.387.
- [6] A. B. Basset, *A treatise on hydrodynamics: with numerous examples*. Deighton, Bell and Company, 1888, vol. 2.
- [7] A. Einstein, “Zur theorie der brownischen bewegung,” *Annalen der physik*, vol. 324, no. 2, pp. 371–381, 1906.
- [8] A. Einstein, “Theoretische bemerkungen über die brownische bewegung,” *Zeitschrift für Elektrochemie und angewandte physikalische Chemie*, vol. 13, no. 6, pp. 41–42, 1907.
- [9] P. Langevin et al., “Sur la théorie du mouvement brownien,” *CR Acad. Sci. Paris*, vol. 146, no. 530-533, p. 530, 1908.
- [10] B. J. Alder and T. E. Wainwright, “Velocity autocorrelations for hard spheres,” *Phys. Rev. Lett.*, vol. 18, pp. 988–990, 23 Jun. 1967. DOI: 10.1103/PhysRevLett.18.988.
- [11] B. J. Alder and T. E. Wainwright, “Decay of the velocity autocorrelation function,” *Phys. Rev. A*, vol. 1, pp. 18–21, 1 Jan. 1970. DOI: 10.1103/PhysRevA.1.18.
- [12] E. H. Hauge and A. Martin-Löf, “Fluctuating hydrodynamics and brownian motion,” *Journal of Statistical Physics*, vol. 7, no. 3, pp. 259–281, Mar. 1973. DOI: 10.1007/BF01030307.
- [13] D. Bedeaux and P. Mazur, “Brownian motion and fluctuating hydrodynamics,” *Physica*, vol. 76, no. 2, pp. 247–258, 1974. DOI: [https://doi.org/10.1016/0031-8914\(74\)90198-0](https://doi.org/10.1016/0031-8914(74)90198-0).
- [14] R. Zwanzig and M. Bixon, “Hydrodynamic theory of the velocity correlation function,” *Phys. Rev. A*, vol. 2, pp. 2005–2012, 5 Nov. 1970. DOI: 10.1103/PhysRevA.2.2005.

- [15] M. H. Ernst, E. H. Hauge, and J. M. J. van Leeuwen, “Asymptotic time behavior of correlation functions,” *Phys. Rev. Lett.*, vol. 25, pp. 1254–1256, 18 Nov. 1970. DOI: 10.1103/PhysRevLett.25.1254.
- [16] M. Ernst, E. Hauge, and J. Van Leeuwen, “Asymptotic time behavior of correlation functions and mode-mode coupling theory,” *Physics Letters A*, vol. 34, no. 7, pp. 419–420, 1971.
- [17] J. R. Dorfman and E. G. D. Cohen, “Velocity correlation functions in two and three dimensions,” *Phys. Rev. Lett.*, vol. 25, pp. 1257–1260, 18 Nov. 1970. DOI: 10.1103/PhysRevLett.25.1257.
- [18] J. Boussinesq, *Théorie analytique de la chaleur: mise en harmonie avec la thermodynamique et avec la théorie mécanique de la lumière*. Gauthier-Villars, 1903, vol. 2.
- [19] S. Kim and S. J. Karrila, “Chapter 12 - particle-wall interactions,” in *Microhydrodynamics*, Butterworth-Heinemann, 1991, pp. 311–320. DOI: <https://doi.org/10.1016/B978-0-7506-9173-4.50018-9>.
- [20] H. A. Lorentz, *Abhandlungen über theoretische Physik*. BG Teubner, 1907, vol. 2.
- [21] H. Faxén, *Einwirkung der gefässwände auf den widerstand gegen die bewegung einer kleinen kugel in einer zähen flüssigkeit*, ger, Uppsala, 1921.
- [22] J. Happel and H. Brenner, “Wall effects on the motion of a single particle,” in *Low Reynolds number hydrodynamics: with special applications to particulate media*. Dordrecht: Springer Netherlands, 1983, pp. 286–357. DOI: 10.1007/978-94-009-8352-6_7.
- [23] N. Liron and S. Mochon, “Stokes flow for a stokeslet between two parallel flat plates,” *Journal of Engineering Mathematics*, vol. 10, no. 4, pp. 287–303, 1976.
- [24] W. W. Hackborn, “Asymmetric stokes flow between parallel planes due to a rotlet,” *Journal of Fluid Mechanics*, vol. 218, pp. 531–546, 1990. DOI: 10.1017/S0022112090001100.
- [25] A. J. Michael, A. Mark, S. Sasic, and H. Ström, “Generalized langevin dynamics in multiphase direct numerical simulations using hydrodynamically optimized memory kernels,” *Physics of Fluids*, vol. 37, no. 3, p. 033317, Mar. 2025. DOI: 10.1063/5.0254930.
- [26] A. J. Micheal, A. Mark, S. Sasic, and H. Ström, “A nonlocal multiscale model for brownian particles: Application to hindered deposition in microfluidic systems,” *International Journal of Multiphase Flow*, vol. 193, p. 105421, 2025.
- [27] X. Bian, C. Kim, and G. E. Karniadakis, “111 years of brownian motion,” *Soft Matter*, vol. 12, pp. 6331–6346, 30 2016. DOI: 10.1039/C6SM01153E.
- [28] R. Kubo, “The fluctuation-dissipation theorem,” *Reports on Progress in Physics*, vol. 29, no. 1, p. 255, Jan. 1966. DOI: 10.1088/0034-4885/29/1/306.
- [29] G. G. Stokes et al., “On the effect of the internal friction of fluids on the motion of pendulums,” 1851.
- [30] P. J. Atzberger, “Velocity correlations of a thermally fluctuating brownian particle: A novel model of the hydrodynamic coupling,” *Physics Letters A*, vol. 351, no. 4, pp. 225–230, 2006. DOI: <https://doi.org/10.1016/j.physleta.2005.10.107>.

-
- [31] H. K. Shin, C. Kim, P. Talkner, and E. K. Lee, “Brownian motion from molecular dynamics,” *Chemical Physics*, vol. 375, no. 2, pp. 316–326, 2010, Stochastic processes in Physics and Chemistry (in honor of Peter Hänggi). DOI: <https://doi.org/10.1016/j.chemphys.2010.05.019>.
 - [32] M. V. Díaz, “On the long-time persistence of hydrodynamic memory,” *The European Physical Journal E*, vol. 44, no. 11, p. 141, 2021.
 - [33] J. W. Dufty, “Gaussian model for fluctuation of a brownian particle,” *The Physics of Fluids*, vol. 17, no. 2, pp. 328–333, Feb. 1974. DOI: 10.1063/1.1694718.
 - [34] R. Mittal and G. Iaccarino, “Immersed boundary methods,” *Annual Review of Fluid Mechanics*, vol. 37, pp. 239–261, Jan. 2005. DOI: 10.1146/annurev.fluid.37.061903.175743.
 - [35] R. Glowinski, T.-W. Pan, and J. Periaux, “A fictitious domain method for dirichlet problem and applications,” *Computer Methods in Applied Mechanics and Engineering*, vol. 111, no. 3-4, pp. 283–303, 1994.
 - [36] N. Sharma and N. A. Patankar, “A fast computation technique for the direct numerical simulation of rigid particulate flows,” *Journal of Computational Physics*, vol. 205, no. 2, pp. 439–457, 2005.
 - [37] N. Sharma, Y. Chen, and N. A. Patankar, “A distributed lagrange multiplier based computational method for the simulation of particulate-stokes flow,” *Computer methods in applied mechanics and engineering*, vol. 194, no. 45-47, pp. 4716–4730, 2005.
 - [38] C. S. Peskin, “Numerical analysis of blood flow in the heart,” *Journal of computational physics*, vol. 25, no. 3, pp. 220–252, 1977.
 - [39] J. Mohd-Yusof, “For simulations of flow in complex geometries,” *Annual research briefs*, vol. 317, p. 35, 1997.
 - [40] J. Mohd-Yusof, “Methods for complex geometries,” *Annual Research Briefs-1998*, vol. 325, 1998.
 - [41] D. Kim and H. Choi, “A second-order time-accurate finite volume method for unsteady incompressible flow on hybrid unstructured grids,” *Journal of computational physics*, vol. 162, no. 2, pp. 411–428, 2000.
 - [42] S. Majumdar, G. Iaccarino, P. Durbin, et al., “Rans solvers with adaptive structured boundary non-conforming grids,” *Annual Research Briefs*, vol. 1, p. 179, 2001.
 - [43] A. Mark and B. G. van Wachem, “Derivation and validation of a novel implicit second-order accurate immersed boundary method,” *Journal of Computational Physics*, vol. 227, no. 13, pp. 6660–6680, 2008. DOI: <https://doi.org/10.1016/j.jcp.2008.03.031>.
 - [44] A. Mark, R. Rundqvist, and F. Edelvik, “Comparison between different immersed boundary conditions for simulation of complex fluid flows,” *Fluid Dynamics and Materials Processing*, vol. 7, no. 3, pp. 241–258, 2011, Cited by: 59. DOI: 10.3970/fdmp.2011.007.241.
 - [45] J. P. Van Doormaal and G. D. Raithby, “Enhancements of the simple method for predicting incompressible fluid flows,” *Numerical heat transfer*, vol. 7, no. 2, pp. 147–163, 1984.

- [46] C. M. Rhie and W.-L. Chow, “Numerical study of the turbulent flow past an airfoil with trailing edge separation,” *AIAA journal*, vol. 21, no. 11, pp. 1525–1532, 1983.
- [47] J. Crank and P. Nicolson, “A practical method for numerical evaluation of solutions of partial differential equations of the heat-conduction type,” in *Mathematical proceedings of the Cambridge philosophical society*, Cambridge University Press, vol. 43, 1947, pp. 50–67.
- [48] S. Hochreiter and J. Schmidhuber, “Long short-term memory,” *Neural Computation*, vol. 9, pp. 1735–1780, Nov. 1997. DOI: 10.1162/neco.1997.9.8.1735.
- [49] S. Cai, Z. Mao, Z. Wang, M. Yin, and G. E. Karniadakis, *Physics-informed neural networks (pinns) for fluid mechanics: A review*, 2021.
- [50] M. Yin, X. Zheng, J. D. Humphrey, and G. E. Karniadakis, “Non-invasive inference of thrombus material properties with physics-informed neural networks,” *Computer Methods in Applied Mechanics and Engineering*, vol. 375, p. 113 603, 2021. DOI: 10.1016/j.cma.2020.113603.
- [51] R. H. Byrd, P. Lu, J. Nocedal, and C. Zhu, “A limited memory algorithm for bound constrained optimization,” *SIAM Journal on Scientific Computing*, vol. 16, no. 5, pp. 1190–1208, 1995. DOI: 10.1137/0916069.
- [52] C. Zhu, R. H. Byrd, P. Lu, and J. Nocedal, “Algorithm 778: L-bfgs-b: Fortran subroutines for large-scale bound-constrained optimization,” *ACM Transactions on Mathematical Software*, vol. 23, no. 4, pp. 550–560, 1997, Cited by: 2779; All Open Access, Bronze Open Access. DOI: 10.1145/279232.279236.
- [53] A. J. Michael, A. Mark, S. Sasic, and H. Ström, “A nonlocal multiscale model for brownian particles: Application to hindered deposition in microfluidic systems,” *International Journal of Multiphase Flow*, vol. 193, p. 105 421, 2025. DOI: <https://doi.org/10.1016/j.ijmultiphaseflow.2025.105421>.



CHALMERS
UNIVERSITY OF TECHNOLOGY