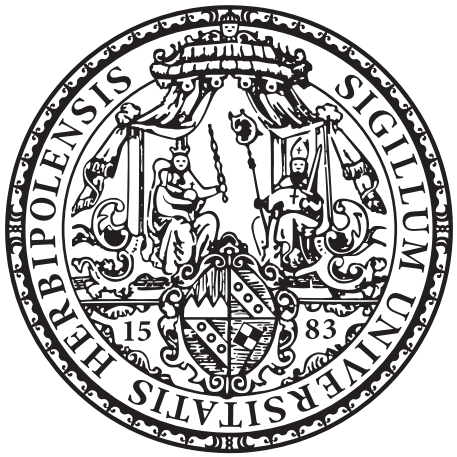
Neural-Network Approaches to Learning Solution Operators for Steady Stokes Flow

Working Group on Numerical Mathematics and Applied Analysis

Supervisor: Dr. Eloi Martinet



Institute of Mathematics
Julius-Maximilians-Universität Würzburg

submitted by

Noe Ennenbach - Matr. N°: 2860820

Stefan Klaus - Matr. N°: 2742233

Tim Plote - Matr. N°: 2739847

Würzburg, Summer Term 2026

Abstract

This report studies data-driven approximations of the force-to-velocity solution operator for stationary incompressible Stokes flow. Reference data are generated with a stable Taylor–Hood finite-element method, and two operator-learning strategies are investigated: direct neural approximation of the matrix-valued Green kernel and a linear graph neural operator. On the no-slip problem, the best direct Green-network model attains a held-out relative error of 0.6626%, while an extended single-precision experiment for counter-moving horizontal walls reaches 0.6200%. The graph neural operator obtains 2.980% error at its training resolution and can be evaluated without retraining on finer grids, although its error increases under this resolution transfer. A further Stokes–Brinkman experiment shows that conditioning a Green network only on local endpoint values of a spatial design field is insufficient for the globally coupled variable-coefficient problem. The results demonstrate the accuracy of direct Green-kernel learning, the flexibility and finite-grid limitations of graph-based operator learning, and the need for global coefficient information in design-dependent flow models.

Contents

1 Introduction **1**

2 Mathematical Background **1**
2.1 Stationary Stokes equations . . . 1
2.2 Existence and uniqueness . . . 2
2.3 The Stokes Green matrix . . . 2
2.4 Operator-learning objective . . . 3

3 Finite-Element Data Generation **3**
3.1 Taylor–Hood reference discretisation . . . 3
3.2 Smooth random body forces . . . 4
3.3 Restriction to the learning resolution . . . 4

4 Green-Kernel Learning with Neural Networks **5**
4.1 Neural representation and error measure . . . 5
4.2 Activation comparison and memory constraint . . . 6
4.3 Development of the learned Green matrix . . . 7
4.4 Corrected top–bottom lid-driven experiments . . . 10

5 Divergence-Free Green-Kernel Models **11**
5.1 Mathematical Background and Training Setup . . . 11
5.2 Results and Comparison to Velocity-Based Green-Kernel Models . . . 12

6 Graph Neural Operator **15**
6.1 Discrete GNO architecture . . . 15
6.2 Motivation for discretisation transfer . . . 15
6.3 Message passing . . . 16
6.4 Training setup . . . 16
6.5 GNO results . . . 17
6.5.1 No-slip operator test . . . 17
6.5.2 Top–bottom lid-driven operator test . . . 18

7 Stokes–Brinkman Bridge towards Shape Optimisation **19**
7.1 Variable-coefficient flow model . . . 19
7.2 Local-knowledge kernel ansatz . . . 20
7.3 Numerical assessment . . . 21

8 Summary and Conclusions **22**

Declaration of AI Assistance **23**

1 Introduction

Many scientific and engineering applications constrained by partial differential equations (PDEs) require multiple solutions for varying inputs rather than a single solution. Instead of repeatedly solving the PDE, it is often more efficient to learn its solution operator directly. This report studies data-driven approximations of the two-dimensional stationary incompressible Stokes equation with force-to-velocity map

$$\boldsymbol{f} \longmapsto \boldsymbol{u}.$$

The same map is approached in two complementary ways. First, a neural network directly parameterises the matrix-valued Green kernel, following the human-understandable operator-learning strategy of Boullé, Earls, and Townsend [3]. Second, a graph neural operator [10] learns the map through a discretised integral operator. Both methods are trained on (force, velocity) pairs obtained from a stable mixed finite-element discretisation.
Section 2 introduces the Stokes equations, their weak formulation, and discusses its well-posedness and the Green-matrix representation. Section 3 describes the finite-element discretisation, the generation of random body forces, and restriction to the learning grid. Section 4 presents the Green-kernel approach and its numerical experiments. These include a comparison of activation functions subject to the documented memory constraint and corrected top–bottom lid-driven data. A divergence-free approach is discussed as a subsequent model improvement in Section 5, while also hihglighting limitations of this approach due to computational limitations. Section 6 develops the graph neural operator and assesses its transfer to finer evaluation grids. The Stokes–Brinkman experiment in Section 7 then assesses whether conditioning a Green network on only local values of a design field provides a useful bridge towards flow optimisation.

2 Mathematical Background

2.1 Stationary Stokes equations

Let $\Omega = (0,1)^2$, let $\nu > 0$ be the kinematic viscosity, and let $\boldsymbol{f} : \Omega \to \mathbb{R}^2$ be a body force. The stationary incompressible Stokes equations are

$$-\nu\Delta\boldsymbol{u} + \nabla p = \boldsymbol{f}, \qquad \nabla\cdot\boldsymbol{u} = 0 \quad \text{in } \Omega, \qquad \boldsymbol{u} = \boldsymbol{g} \quad \text{on } \partial\Omega. \tag{1}$$

Here $\boldsymbol{u}$ is the velocity and p is the pressure. The pressure is determined only up to an additive constant and is therefore normalised to have zero mean.
For homogeneous no-slip boundary data, define

$$V = H_0^1(\Omega)^2, \qquad Q = L_0^2(\Omega) := \left\{ q \in L^2(\Omega) : \int_\Omega q \,\mathrm{d}x = 0 \right\}.$$

The notation

$$H^{-1}(\Omega) := (H_0^1(\Omega))'$$

denotes the topological dual of $H_0^1(\Omega)$. Its Cartesian product $H^{-1}(\Omega)^2$ consists of pairs of scalar dual functionals and is canonically identified with $V' = (H_0^1(\Omega)^2)'$; the duality pairing below is understood componentwise.

Multiplying the momentum balance by $\boldsymbol{v} \in V$, integrating by parts, and testing incompressibility with $q \in Q$ gives: find $(\boldsymbol{u}, p) \in V \times Q$ such that

$$\nu(\nabla\boldsymbol{u}, \nabla\boldsymbol{v})_{L^2(\Omega)} - (p, \nabla\cdot\boldsymbol{v})_{L^2(\Omega)} = \langle \boldsymbol{f}, \boldsymbol{v}\rangle \qquad \forall \boldsymbol{v} \in V, \tag{2a}$$

$$-(q, \nabla\cdot\boldsymbol{u})_{L^2(\Omega)} = 0 \qquad \forall q \in Q, \tag{2b}$$

where we denote the L^2 inner product by

$$(a,b)_{L^2(\Omega)} := \int_\Omega \langle a,b\rangle,$$

where $\langle\cdot,\cdot\rangle$ denotes the Euclidean product of vector valued functions. In particular,

$$\boldsymbol{u}\cdot\boldsymbol{v} = \sum_{i=1}^{2} u_i v_i, \qquad \langle\nabla\boldsymbol{u},\nabla\boldsymbol{v}\rangle = \sum_{i,j=1}^{2} \partial_j u_i\,\partial_j v_i.$$

This is the standard velocity–pressure saddle-point formulation [1, p. 462, Eqs. (8.2.1)–(8.2.5)]. Nonzero boundary data with zero net flux can be treated by a divergence-compatible lifting of the boundary values [1, p. 464, Remark 8.2.2].

2.2 Existence and uniqueness

By introducing the bilinear forms

$$a(\boldsymbol{u},\boldsymbol{v}) = \nu(\nabla\boldsymbol{u},\nabla\boldsymbol{v})_{L^2(\Omega)}, \qquad b(\boldsymbol{v},q) = -(q,\nabla\cdot\boldsymbol{v})_{L^2(\Omega)}$$

the weak formulation (2) can be expressed as an abstract saddle-point problem. The form a is coercive on the divergence-free kernel of b, and $V\times Q$ satisfies the continuous inf–sup condition. The classical saddle-point result therefore gives the following statement [1, p. 463, Theorem 8.2.1]:

Theorem 2.1 (Well-posedness of the Dirichlet Stokes problem)**.** *Let Ω be a bounded, connected Lipschitz domain and let $\nu>0$. For every $\boldsymbol{f}\in H^{-1}(\Omega)^2$, there is a unique pair $(\boldsymbol{u},p)\in H^1_0(\Omega)^2\times L^2_0(\Omega)$ satisfying* (2)*. Moreover,*

$$\|\boldsymbol{u}\|_{H^1(\Omega)} + \|p\|_{L^2(\Omega)} \le C_{\Omega,\nu}\|\boldsymbol{f}\|_{H^{-1}(\Omega)^2}.$$

Without the zero-mean normalisation, the velocity remains unique and the pressure is unique modulo constants.

Consequently, the force-to-velocity solution operator

$$\mathcal{S} : H^{-1}(\Omega)^2 \longrightarrow H^1_0(\Omega)^2, \qquad \mathcal{S}(\boldsymbol{f}) = \boldsymbol{u}, \tag{3}$$

is linear and continuous.

2.3 The Stokes Green matrix

For a source point $y\in\Omega$ and coordinate vector $\boldsymbol{e}_j$, the jth column of the Green pair $(\boldsymbol{G},\boldsymbol{\Pi})$ is defined distributionally by

$$\begin{aligned} -\nu\Delta_x \boldsymbol{G}_{\cdot j}(x,y) + \nabla_x \Pi_j(x,y) = \delta_y \boldsymbol{e}_j, \qquad \nabla_x\cdot\boldsymbol{G}_{\cdot j}(x,y) = 0 \quad &\text{in } \Omega,\\ \boldsymbol{G}_{\cdot j}(x,y) = 0 \quad &\text{on } \partial\Omega. \end{aligned} \tag{4}$$

The pressure Green function is normalised to have zero mean. Green functions for stationary Stokes systems in two-dimensional domains exist, with pointwise estimates, under assumptions that include the constant-coefficient Dirichlet problem on the square [5, p. 109, Theorem 3.2].

Here, "sufficiently regular forcing" will mean explicitly $\boldsymbol{f} \in L^\infty(\Omega)^2$; smooth compactly supported forcing is a special case. For such forcing, the velocity representation holds for almost every $x \in \Omega$ [5, pp. 107–108, Definition 2.3(iii)]:

$$u_i(x) = \sum_{j=1}^{2} \int_\Omega G_{ij}(x,y) f_j(y)\,\mathrm{d}y, \qquad i = 1, 2. \tag{5}$$

For prescribed nonzero boundary velocities, the force-dependent part is still given by this integral, while a homogeneous Stokes solution accounts for the boundary data:

$$\boldsymbol{u}(x) = \int_\Omega \boldsymbol{G}(x,y)\boldsymbol{f}(y)\,\mathrm{d}y + \boldsymbol{u}_{\mathrm{hom}}(x). \tag{6}$$

2.4 Operator-learning objective

Given solution pairs $\{(\boldsymbol{f}^{(n)}, \boldsymbol{u}^{(n)})\}_{n=1}^{N}$, the objective is to approximate the solution operator

$$\boldsymbol{f} \longmapsto \boldsymbol{u} = \mathcal{S}(\boldsymbol{f}).$$

The Green-network approach parameterises the four entries of $\boldsymbol{G}$ directly and fits them through their action on random forces, as proposed by Boullé, Earls, and Townsend [3, pp. 3–4]. The graph neural operator described by Kovachki et al. [10] and used in Section 6 instead parameterises the same solution map by a learned discrete integral operator.

3 Finite-Element Data Generation

The neural-network models are trained on reference solutions obtained with the finite-element method (FEM). Velocity fields are computed by solving the stationary Stokes equations with a stable Taylor–Hood finite-element discretisation. Random body forces are generated to sample the solution operator, and the resulting solutions are evaluated on a fixed Cartesian grid that serves as the learning discretisation for the neural-network models.

3.1 Taylor–Hood reference discretisation

The reference solutions are generated with a conforming mixed finite-element method (FEM). The unit square is divided into $l \times l$ square cells and each cell is triangulated. Velocity and pressure are approximated by the Taylor–Hood pair

$$V_h = \{\text{continuous, vector-valued } P_2 \text{ functions}\}, \qquad Q_h = \{\text{continuous } P_1 \text{ functions}\}.$$

Taylor and Hood introduced this mixed velocity–pressure construction in their finite-element treatment of incompressible flow [13, pp. 73–76]. On shape-regular triangulations, the (P_2, P_1) pair satisfies the discrete inf–sup condition and is therefore a stable choice for Stokes flow [1, pp. 496–498, Theorem 8.8.1].

With velocity basis functions $\{\boldsymbol{\varphi}_i\}$ and pressure basis functions $\{\psi_k\}$, the assembled system has the block structure

$$\begin{bmatrix} \nu A & B^{\mathsf{T}} \\ B & 0 \end{bmatrix} \begin{bmatrix} \mathbf{u} \\ \mathbf{p} \end{bmatrix} = \begin{bmatrix} \mathbf{F} \\ 0 \end{bmatrix}, \quad A_{ij} = \int_\Omega \nabla\boldsymbol{\varphi}_j : \nabla\boldsymbol{\varphi}_i \,\mathrm{d}x, \quad B_{ki} = -\int_\Omega \psi_k \nabla\cdot\boldsymbol{\varphi}_i \,\mathrm{d}x. \tag{7}$$

A quadrature order of four is used. Equation (7) displays the system before Dirichlet elimination. When prescribed nonzero velocity degrees of freedom are eliminated, their known contributions are transferred to the reduced right-hand side; these contributions are included in the implementation. One pressure degree of freedom is pinned to remove the discrete constant nullspace. Two boundary configurations are considered. In the no-slip case all walls are stationary. In the top–bottom lid-driven case,

$$\boldsymbol{g} = (1, 0)^{\mathsf{T}} \text{ on the top wall}, \qquad \boldsymbol{g} = (-1/3, 0)^{\mathsf{T}} \text{ on the bottom wall},$$

and the vertical walls are stationary. These values are assigned directly to the corresponding vector-valued P_2 boundary degrees of freedom. At the four corner degrees of freedom, the top- or bottom-wall value takes precedence over the stationary vertical-wall value. A zero-force sample is included to anchor the learned homogeneous component. The configuration is related to, but distinct from, the top-lid-only cavity experiment of Boullé, Earls, and Townsend [3, Supplementary Material Sec. 8.4].

All numerical experiments are implemented in Python. The finite-element implementation is based on the `scikit-fem` library [7]. The computational domain is discretised by a mesh of square cells with $l = 113$, each subdivided into two triangles. Unless stated otherwise, all experiments use a viscosity of $\nu = 10^{-2}$.

3.2 Smooth random body forces

Each component of the body force is an independent smooth Gaussian random field with target wavelength $\lambda = 0.1$, matching the role of `randnfun2(0.1)` in the reference Stokes experiment [3, Supplementary Material Sec. 8.4]. A Fourier–Wiener series is constructed on an extended periodic square of side $L = 1.4$ and then restricted to Ω:

$$f_\ell(x) = \mathrm{Re}\left[\frac{1}{\sqrt{|\mathcal{K}|}} \sum_{k \in \mathcal{K}} c_{\ell,k} \exp\left(2\pi \mathrm{i}\, \frac{k \cdot x}{L}\right)\right], \qquad \ell \in \{1, 2\}, \tag{8}$$

where the $c_{\ell,k}$ are independent complex Gaussian coefficients and

$$\mathcal{K} = \left\{(k_1, k_2) \in \{-\mu, \ldots, \mu\}^2 : (k_1/\mu)^2 + (k_2/\mu)^2 \le 1 \text{ with weight } \mu = L/\lambda = 14\right\}.$$

The circular spectral cutoff makes the fields approximately isotropic. Smooth random functions and their Gaussian-process interpretation are discussed by Filip, Javeed, and Trefethen [6, pp. 185–188].

3.3 Restriction to the learning resolution

For every random force, the load is evaluated at finite-element quadrature points and the Taylor–Hood system is solved on the fine mesh. The continuous P_2 velocity and the force are then evaluated at the Cartesian grid

$$X_h = \{(i/26, j/26) : i, j = 0, \ldots, 26\},$$

which contains $27 \times 27 = 729$ points. Thus the lower-resolution data are obtained by evaluating the fine-grid finite-element solution, rather than by averaging or re-solving the PDE on a coarse mesh. The principal Stokes data set contains $N = 200$ force–velocity pairs. An 80/20 split gives 160 training pairs and 40 held-out test pairs. The Green integral is discretised by the tensor-product trapezoidal rule on X_h; the weights account for interior, edge, and corner points and sum to $|\Omega| = 1$.

4 Green-Kernel Learning with Neural Networks

4.1 Neural representation and error measure

The representation follows the two-network construction of Boullé, Earls, and Townsend [3]. A Green network $N_G : \mathbb{R}^4 \to \mathbb{R}^4$ maps (x_1, x_2, y_1, y_2) to the four entries of $\boldsymbol{G}_\theta(x, y)$. For top–bottom lid-driven data, a second network $N_{\mathrm{hom}} : \mathbb{R}^2 \to \mathbb{R}^2$ represents the homogeneous velocity; for no-slip data this term is fixed to zero. Each multilayer perceptron (MLP) has four hidden layers. On the $M = 729$ point learning grid, the prediction is

$$\widehat{\boldsymbol{u}}^{(n)}(x_i) = \sum_{k=1}^{M} w_k \, \boldsymbol{G}_\theta(x_i, y_k) \boldsymbol{f}^{(n)}(y_k) + \boldsymbol{u}_{\mathrm{hom},\phi}(x_i), \tag{9}$$

where w_k are tensor-product trapezoidal weights.
The network is trained by minimising the mean sample-wise relative discretationed L^2 error, which is also used for reporting the final prediction accuracy:

$$\mathcal{E}(\theta, \phi) = \frac{1}{N} \sum_{n=1}^{N} \left(\frac{\sum_{i=1}^{M} \|\widehat{\boldsymbol{u}}^{(n)}(x_i) - \boldsymbol{u}^{(n)}(x_i)\|_2^2}{\sum_{i=1}^{M} \|\boldsymbol{u}^{(n)}(x_i)\|_2^2} \right)^{1/2}. \tag{10}$$

This differs from the loss used in the reference work [3, Supplementary Material, Sec. 2]. That loss averages squared sample-relative continuous L^2 residuals and discretises the integrals by quadrature, whereas (10) averages unsquared relative Euclidean norms of the sampled values. The trapezoidal weights enter (9), but not the reported error norm.
The models were implemented using the PyTorch deep learning framework [11]. The baseline training schedule uses 500 Adam epochs with learning rate 1.5×10^{-3}, followed by 2500 limited-memory Broyden–Fletcher–Goldfarb–Shanno (L-BFGS) epochs with learning rate one, at most five inner iterations per epoch, and history size 400. These baseline calculations use 64-bit floating-point arithmetic (FP64). The extended top–bottom lid-driven run retains 500 Adam epochs, increases the L-BFGS phase to 9500 epochs, and uses 32-bit floating-point arithmetic (FP32). The activation abbreviations are sigmoid linear unit (SiLU) and rectified linear unit (ReLU); rational $(3, 2)$ denotes numerator and denominator degrees three and two. Table 1 reports the final errors from the activation study and both corrected lid-driven experiments. All calculations in this section were performed on an NVIDIA GeForce RTX 3070; the reported runtimes are elapsed neural-network training times.

Table 1: Final Green-network solution errors measured by (10). All models have four hidden layers. The extended FP32 run uses 500 Adam plus 9500 L-BFGS epochs; all other runs use 500 Adam plus 2500 L-BFGS epochs. Runtime is measured in seconds.

Boundary data	Precision	Activation	Width	Epochs	Runtime [s]	Train error	Test error
No slip	FP64	SiLU	128	3000	10 507.16	0.4551%	0.6626%
No slip	FP64	tanh	128	3000	10 196.93	0.5044%	0.8140%
No slip	FP64	rational $(3, 2)$	41	3000	4439.97	1.7443%	2.1554%
No slip	FP64	ReLU	128	3000	9335.44	1.5537%	2.6439%
Top–bottom lid-driven	FP64	SiLU	128	3000	10 622.56	1.5425%	1.7654%
Top–bottom lid-driven	FP32	SiLU	128	10000	3252.84	0.5118%	0.6200%

Zero-shot resolution transfer. The best no-slip model (FP64 SiLU, width 128) was evaluated without retraining on 200 independently generated force–solution pairs constructed by the same

procedure as the training data. Table 2 reports the average quadrature-weighted relative L^2 error. The 27×27 value differs from the test error in Table 1 because this experiment uses a new 200-pair test set and a quadrature-weighted norm rather than the unweighted norm in (10).

Table 2: Zero-shot resolution test of the no-slip FP64 SiLU model.

Testing grid	Average relative error	Error in percent
27×27	0.005789	0.5789%
50×50	0.025611	2.5611%
53×53	0.026277	2.6277%

The error increases on the finer grids, which may indicate overfitting to the 27×27 learning grid. Better resolution interpolation may be obtained by training the same model across several learning grids.

4.2 Activation comparison and memory constraint

The no-slip comparison in Figure 1 shows that SiLU gives the smallest final held-out error. The tanh activation is the closest alternative, while ReLU converges to a substantially larger error. The rational model also remains above SiLU and tanh, although its smaller width prevents a pure activation-only comparison.

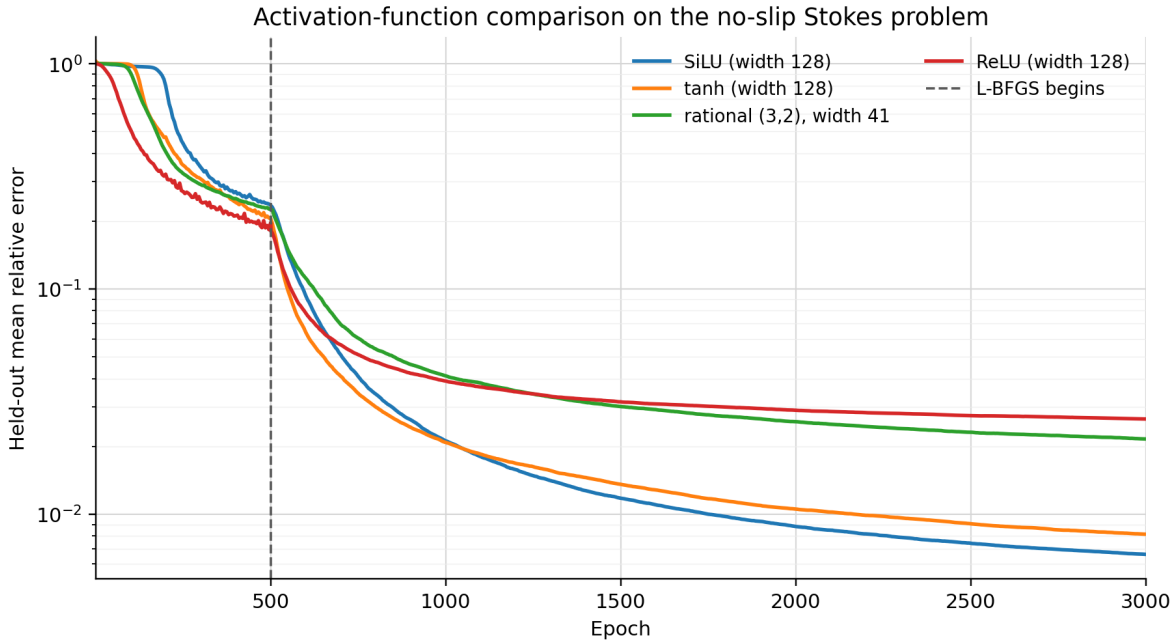


Figure 1: Held-out relative error for the four activation choices. The change in convergence rate at epoch 500 corresponds to the switch from Adam to L-BFGS. The SiLU, tanh, and ReLU networks have width 128; the rational network has width 41.

Rational activations are trainable quotient functions designed to provide high approximation accuracy with comparatively few layers [4, pp. 14243–14245]. In the present implementation, retaining width 128 for the rational model was not feasible within the available GPU-memory

budget. The completed rational run therefore uses width 41 and nevertheless records 5708 MiB of video random-access memory (VRAM). Evaluating all 729^2 source–target pairs in double precision creates large intermediate tensors, and the L-BFGS history adds further storage. This forced reduction in width must be kept in mind when interpreting the rational result: it combines an activation change with a capacity change. Within the feasible controlled runs, SiLU is the preferred activation.

4.3 Development of the learned Green matrix

Figures 2–5 show the no-slip SiLU kernel at epochs 100, 200, 250, and 3000 for the fixed source $y_0 = (0.5, 0.5)$. At epoch 100 the response is still broad and dominated by large-scale boundary structure. By epoch 200, localised and coupled source-response patterns have emerged; the epoch-250 kernel shows their continued development during the Adam phase. At epoch 3000, the diagonal entries are sharply localised near $x = y_0$, while the off-diagonal entries show the expected component coupling and reflection structure. Each panel uses its own colour scale; the figures therefore document structural development rather than a directly comparable amplitude scale.

These images are descriptive rather than a pointwise validation of the exact Green matrix. The loss constrains the integrated action of the kernel on the training forces, so a low held-out solution error does not by itself prove pointwise identification of every entry.

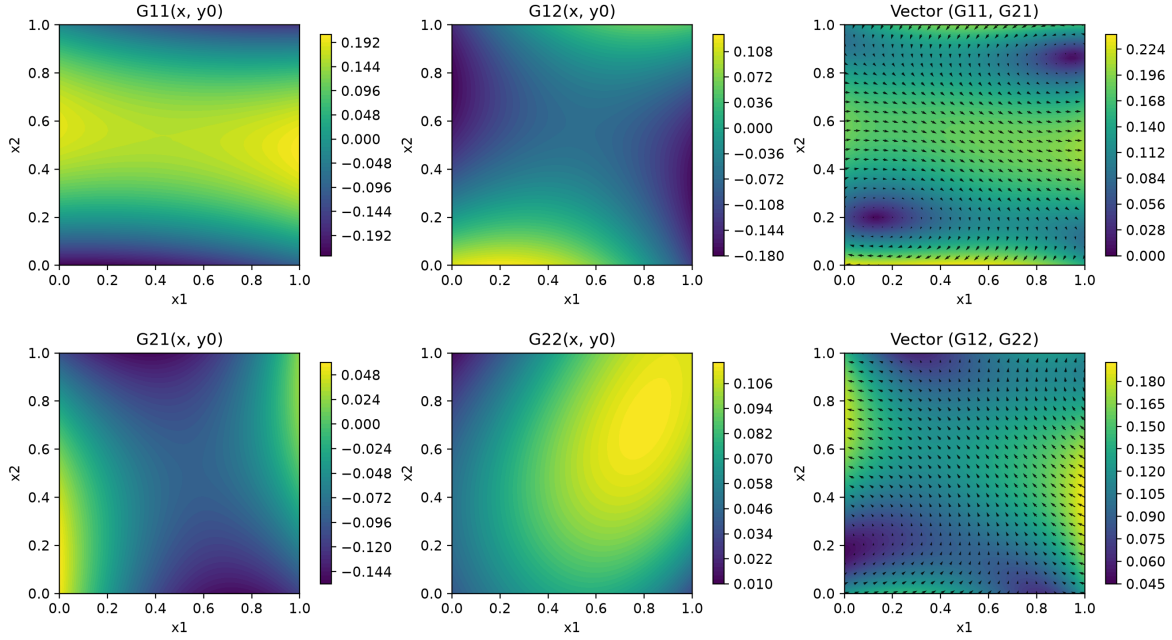


Figure 2: Learned no-slip SiLU Green matrix at epoch 100 and source $y_0 = (0.5, 0.5)$.

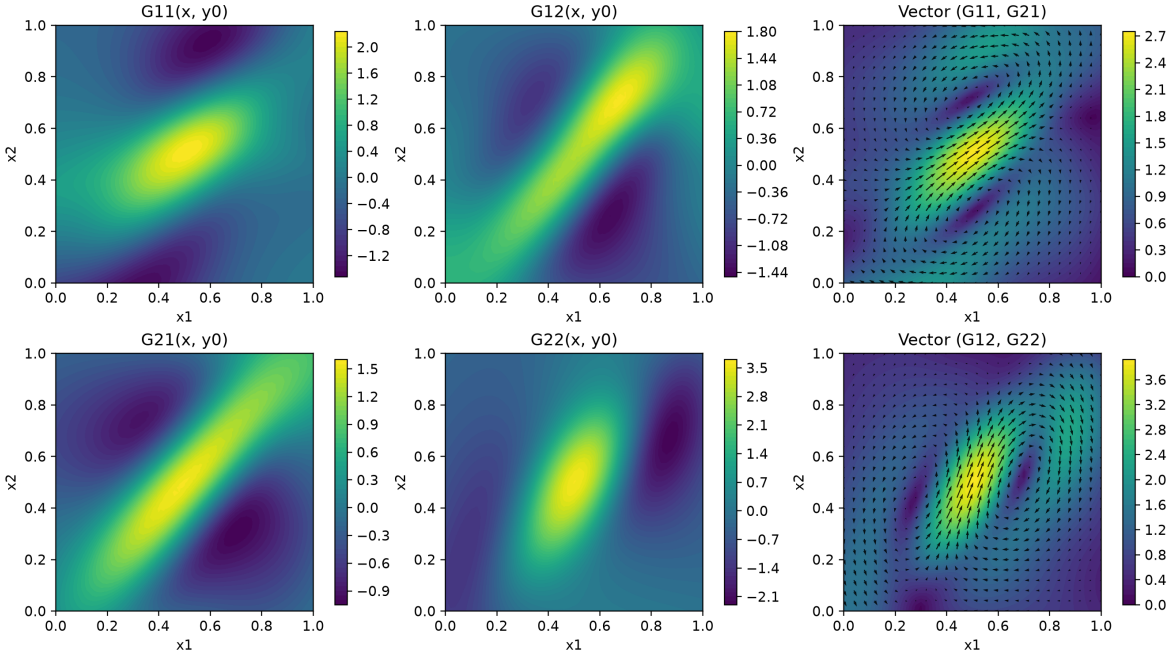


Figure 3: Learned no-slip SiLU Green matrix at epoch 200 and source $y_0 = (0.5, 0.5)$.

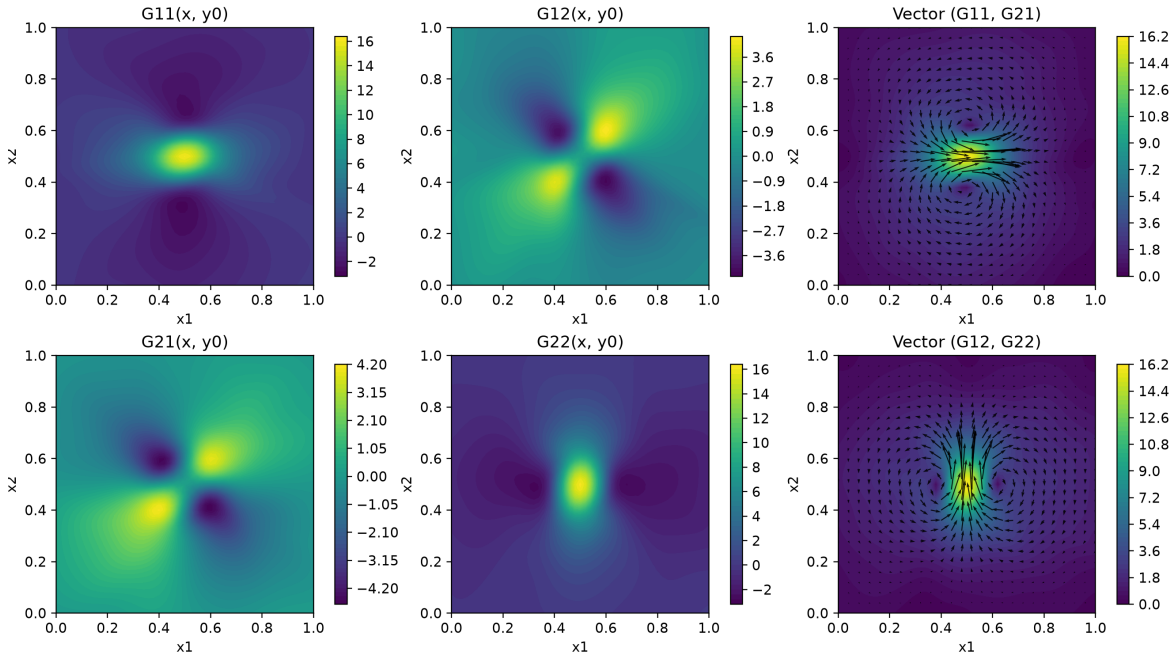


Figure 4: Learned no-slip SiLU Green matrix at epoch 250 and source $y_0 = (0.5, 0.5)$.

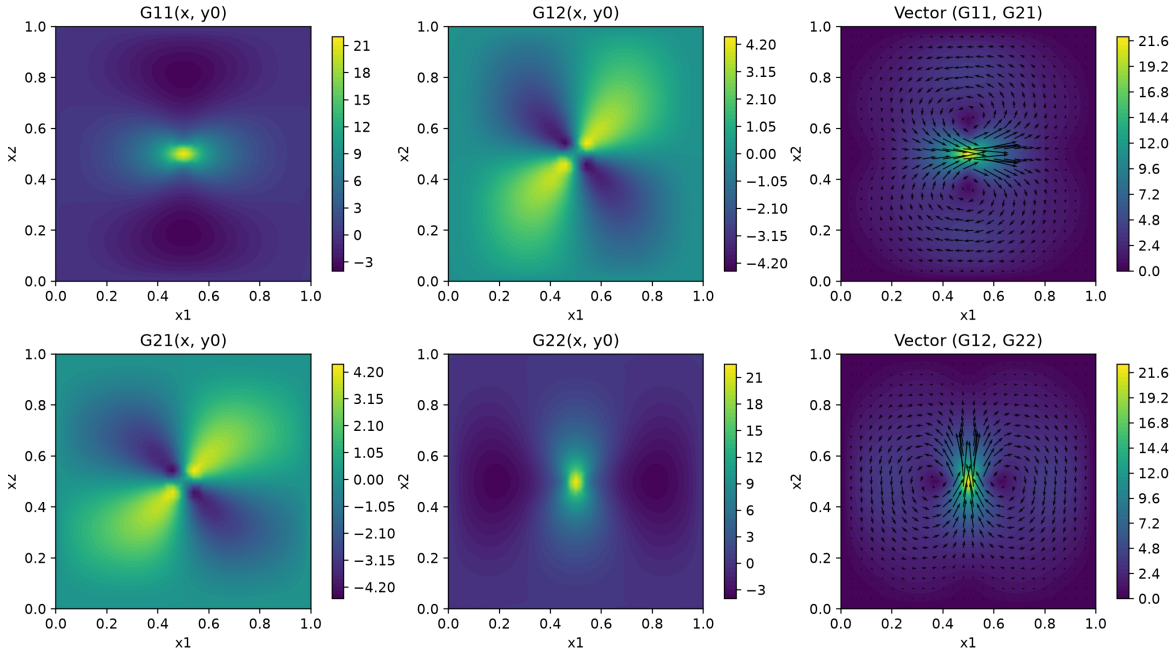


Figure 5: Learned no-slip SiLU Green matrix after training, at epoch 3000 and source $y_0 = (0.5, 0.5)$.

4.4 Corrected top–bottom lid-driven experiments

For nonhomogeneous data, (9) includes the learned homogeneous component. The corrected experiment imposes the top velocity $(1, 0)^\mathsf{T}$ and bottom velocity $(-1/3, 0)^\mathsf{T}$ directly on the boundary degrees of freedom and includes a zero-force anchor. In FP64, the training and held-out errors decrease steadily to 1.5425% and 1.7654% at epoch 3000 (Figure 6). Their small gap provides no evidence of severe overfitting; the remaining error may instead arise from approximation or optimisation.



Figure 6: Convergence of the corrected width-128 SiLU model for the top–bottom lid-driven problem. The dashed line marks the switch from Adam to L-BFGS.

Extended single-precision run. This run retains the same 200 samples, fixed 160/40 split, 27×27 trapezoidal grid, width-128 four-layer SiLU networks, other optimiser settings, and relative-error objective; it has no additional boundary penalty. The data differ only by the expected FP32 rounding. Extending L-BFGS from 2500 to 9500 epochs gives 10 000 epochs in total. Figure 7 shows continued improvement through the final epoch: the minimum held-out error is 0.6200% and the training error is 0.5118%. This lowers the FP64 held-out error by 1.1454 percentage points, or 64.9%, in 3252.84 seconds (54 min 12.84 s). Because precision and epoch count change together, this comparison does not isolate the effect of precision. It nevertheless shows that the extended FP32 configuration is a practical training option, achieving a better fit in substantially less time.

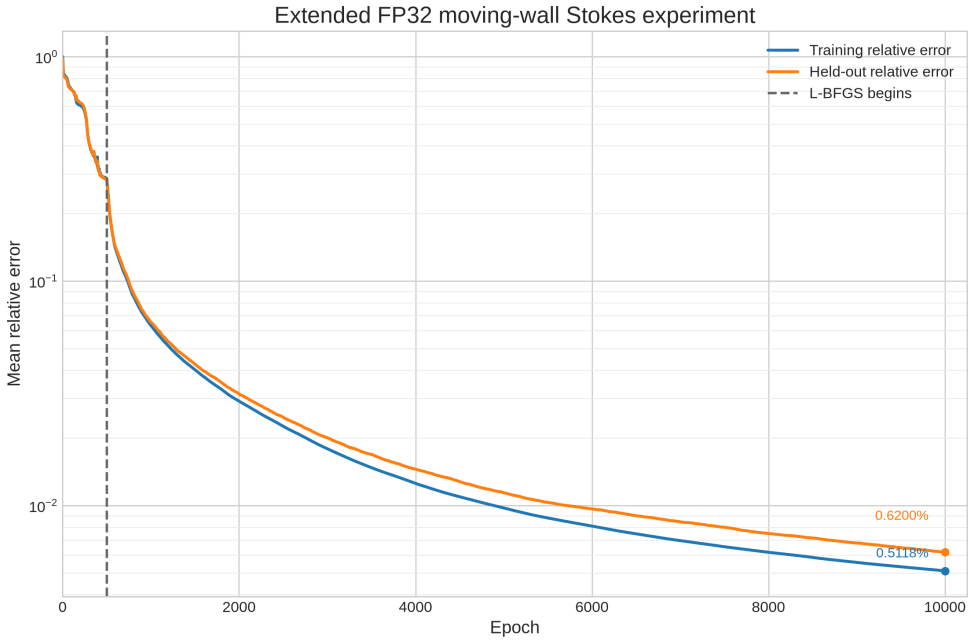


Figure 7: Training and held-out relative solution errors for the extended width-128 SiLU run in FP32. Both curves use (10); the dashed line marks the switch from Adam to L-BFGS at epoch 500.

5 Divergence-Free Green-Kernel Models

5.1 Mathematical Background and Training Setup

The Green-function formulation introduced in the previous section predicts the velocity field directly. Although the incompressibility constraint is satisfied by the reference finite-element solutions, it is only enforced implicitly during learning. This section presents an alternative formulation in which incompressibility is built directly into the representation by predicting a scalar stream-function Green kernel. Such structure-preserving parameterizations have recently received considerable attention in scientific machine learning, since they enforce the divergence-free constraint exactly at the architectural level rather than only through the loss function [12, 9]. The velocity field is then recovered through differentiation, guaranteeing a divergence-free prediction satisfying $\nabla_x \cdot \mathbf{G}_{\cdot j}(x, y) = 0$ by construction. An incompressible vector field can be represented in two dimensions by a scalar stream function ψ such that

$$\boldsymbol{u} = \nabla^{\perp}\psi = \begin{bmatrix} \partial_{x_2}\psi \\ -\partial_{x_1}\psi \end{bmatrix}, \tag{11}$$

where $\nabla^{\perp}$ denotes the perpendicular gradient operator. This implies that the velocity field is divergence-free by construction, $\nabla \cdot \boldsymbol{u} = \partial_{x_1}\partial_{x_2}\psi - \partial_{x_2}\partial_{x_1}\psi = 0$, provided that ψ is sufficiently smooth.

The velocity representation for homogeneous problems can be rewritten in terms of the stream-function Green kernel $\boldsymbol{K} = (K_1, K_2)^{\top}$ as

$$\boldsymbol{u}(x) = \int_{\Omega} \nabla_x^{\perp} \boldsymbol{K}(x, y) \boldsymbol{f}(y)\, dy, \tag{12}$$

where $\nabla_x^{\perp}$ acts componentwise on $\boldsymbol{K}$. The neural network predicts the stream-function Green kernel and outputs two scalar kernels for any pair of target and source coordinates (x, y). The velocity field is then recovered by differentiating $\boldsymbol{K}$ with respect to the target coordinates x. The differentiation is implemented using automatic differentiation in PyTorch, which is a standard feature of the framework [11, Sec. 4.3].
The training data are generated analogously to the velocity-based Green-kernel model. First, the Stokes problem is solved using the Taylor–Hood finite-element discretization to obtain the reference velocity field. Subsequently, the corresponding stream function is reconstructed by solving the auxiliary finite-element problem

$$\int_\Omega \nabla\psi_h \cdot \nabla v_h \, dx = \int_\Omega (-u_{2,h}, u_{1,h}) \cdot \nabla v_h \, dx, \qquad \forall v_h \in V_h, \tag{13}$$

in a scalar quadratic finite-element space. Homogeneous Dirichlet boundary conditions are imposed. The reconstructed stream function is then evaluated at the neural-network sampling points and serves as the training target.
The loss function is formulated in terms of the reconstructed stream function rather than the velocity field analogous to (10):

$$\epsilon(\theta, \phi) = \left(\frac{\sum_{i=1}^{N} w_i \left(\hat{\psi}^{(n)}(x_i) - \psi^{(n)}(x_i) \right)^2}{\sum_{i=1}^{N} w_i \psi^{(n)}(x_i)^2 + 0^+} \right)^{1/2}$$

During training, the predicted stream function is obtained by integrating the learned Green kernels against the forcing, and the network parameters are optimized by minimizing the relative discrete L^2-error between the predicted and reconstructed stream functions. The velocity field is recovered only during inference and evaluation by differentiating the learned kernels, while the divergence-free property is satisfied identically by construction.

5.2 Results and Comparison to Velocity-Based Green-Kernel Models

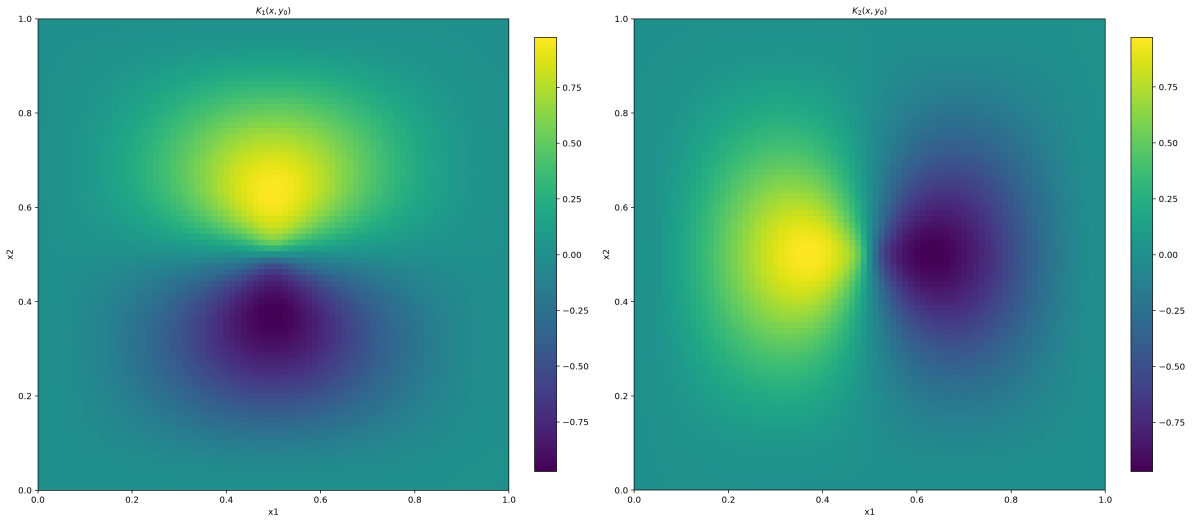


Figure 8: Learned scalar kernels K_1 and K_2 for fixed source point $y = (0.5, 0.5)$.

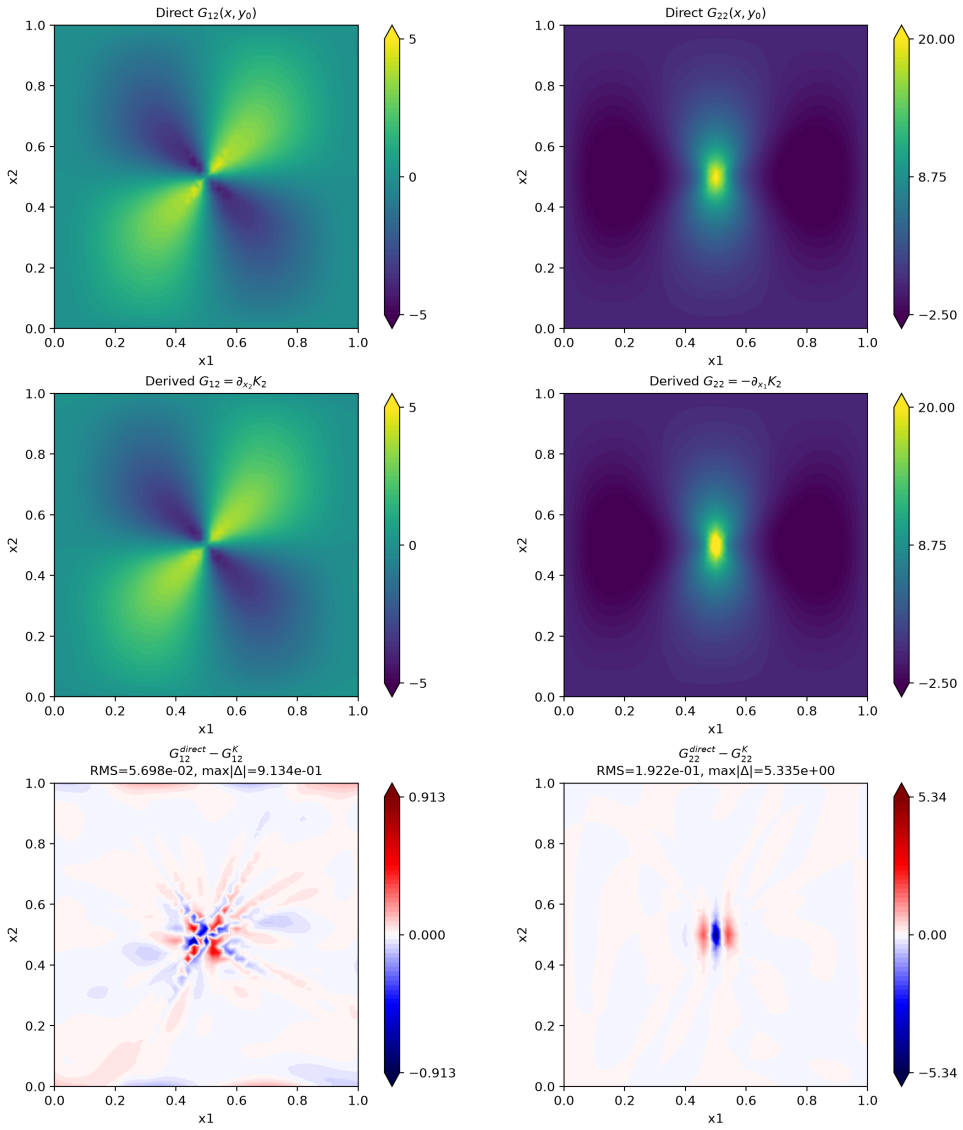


Figure 9: Comparison of G_{12} and G_{22} obtained from the velocity-based model and reconstructed from K_2, including their point wise differences, which are the largest at the singularities in the centre and at the boundary.

The following results were obtained using the same network architecture and training procedure as for the velocity-based Green-kernel model, employing the `tanh` activation function. Both models were trained for 2570 epochs to enable a direct comparison on the same FEM-data set. The divergence-free model learns the scalar kernels K_1 and K_2, from which the Green tensor is reconstructed as

$$G_K(x, y) = \begin{pmatrix} \partial_{x_2} K_1 & \partial_{x_2} K_2 \\ -\partial_{x_1} K_1 & -\partial_{x_1} K_2 \end{pmatrix}.$$

Figure 8 shows the learned kernels for a fixed source location, displaying the expected 180-degree rotational antisymmetry around the source point, as well as a 90-degree rotational symmetry between the two kernels. Figure 9 compares the reconstructed components G_{12}^K, G_{22}^K with the

corresponding components obtained from the directly trained velocity-based model. The pointwise differences indicate where the two formulations deviate. These differences are largest near the domain boundary and near the singularity of the Green kernel. This behavior is expected because differentiating the learned kernels amplifies approximation errors, particularly near the singularity and close to the domain boundary. The training and test histories are shown in Figure 10. Since the velocity-based model is trained on velocity data and the divergence-free model on stream-function data, the training loss values are not directly comparable. However, the curves allow a comparison of convergence and generalization. Recovering the velocity field by automatic differentiation increases the computational cost and memory requirements during evaluation. If velocity-based quantities are also included in the training loss, this differentiation would additionally increase the training cost. This additional overhead arises from the repeated evaluation of spatial derivatives and has also been reported for other divergence-free neural-network parameterizations based on automatic differentiation [12].

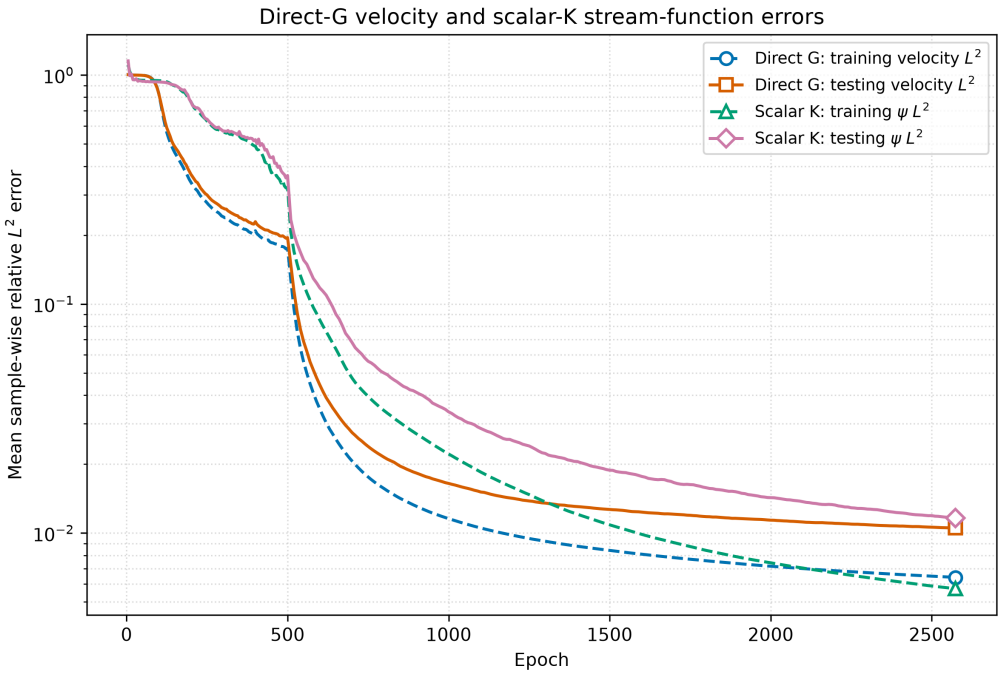


Figure 10: Training and held-out error histories for the velocity-based and stream-function-based models. At later epochs, the predictive accuracy of the stream-function model approaches that of the directly learned velocity model.

The main advantage of the scalar-kernel formulation is that incompressibility is satisfied exactly by construction. In contrast, the directly learned Green tensor does not explicitly enforce the divergence-free condition. Point wise evaluation of the divergence of the directly trained model yields a mean divergence of the order of 10^{-2}, whereas the divergence-free formulation produces a residual divergence at machine precision, of the order of 10^{-16}.
Although the divergence-free formulation guarantees incompressibility exactly, it requires significantly more computational time and memory than the direct velocity-based formulation. Consequently, the velocity-based approach provides a more computationally efficient alternative when exact point wise incompressibility is not required.

6 Graph Neural Operator

The preceding sections already define the force-to-velocity operator $\mathcal{S}$ in (3), the finite-element reference solver, and the learning grid X_h. This section therefore focuses only on the graph neural operator (GNO) used as an alternative approximation of the same map. The construction follows the coordinate-kernel GNO of Kovachki et al. [10], but is adapted to the linear Stokes problem and to the notation established above.

6.1 Discrete GNO architecture

The graph nodes are the $M = 729$ points $x_i \in X_h$ introduced in Section 3. For each target node, define the physical-radius neighbourhood

$$\mathcal{N}_r(x_i) = \{x_j \in X_h : \|x_i - x_j\|_2 \le r\}, \qquad r = 0.25.$$

The sampled force $\boldsymbol{f}(x_i) \in \mathbb{R}^2$ is lifted to a 32-dimensional feature vector. Four message-passing layers are then applied, followed by a projection back to the two velocity components:

$$\begin{aligned} \boldsymbol{v}_0(x_i) &= P\boldsymbol{f}(x_i), \\ \boldsymbol{v}_{t+1}(x_i) &= W_t\boldsymbol{v}_t(x_i) + \frac{1}{|\mathcal{N}_r(x_i)|}\sum_{x_j \in \mathcal{N}_r(x_i)} \boldsymbol{\kappa}_t(x_j, x_i) \odot \boldsymbol{v}_t(x_j), \qquad t = 0, \dots, 3, \\ \widehat{\boldsymbol{u}}_\theta(x_i) &= Q\boldsymbol{v}_4(x_i). \end{aligned} \tag{14}$$

Here $P \in \mathbb{R}^{32\times 2}$, $W_t \in \mathbb{R}^{32\times 32}$, and $Q \in \mathbb{R}^{2\times 32}$ are learned linear maps. The symbol $\odot$ denotes componentwise multiplication. Each $\boldsymbol{\kappa}_t(x_j, x_i) \in \mathbb{R}^{32}$ is produced from the source and target coordinates by a three-hidden-layer ReLU network of width 256. The neighbourhood sum in (14) is a mean aggregation in the sense of message-passing neural networks [8]. Its graph interpretation is described in Section 6.3.

The maps P, W_t, and Q have no biases, and no activation is applied to the force-dependent features $\boldsymbol{v}_t$. Although the coordinate networks $\boldsymbol{\kappa}_t$ are nonlinear in (x_j, x_i), their output only multiplies $\boldsymbol{v}_t(x_j)$. Consequently, the complete map remains exactly linear in $\boldsymbol{f}$. Finally, a fixed mask sets the predicted velocity to zero on $\partial\Omega$, consistently with the no-slip data used for this experiment.

6.2 Motivation for discretisation transfer

Neural operators are motivated by the fact that their learned maps act on fields rather than on one fixed-length vector. The coordinate kernels and pointwise maps in (14) therefore do not depend on the number of nodes. The same parameters can be evaluated on the training grid X_h and on a refined grid containing more samples of the same force field. This is the relevant distinction from a conventional network whose input and output sizes would change with the grid. The general discretisation-invariance result is formalised by Kovachki et al. [10, Definition 4 and Theorem 8]; here it is used only as motivation for the zero-shot test.

For a force $\boldsymbol{f}$, write $\mathcal{S}(\boldsymbol{f})$ for the FEM force-to-velocity solution and $\mathcal{S}_\theta$ for the learned operator. On a grid X_h, the observed error can be viewed schematically as

$$\left\|\widehat{\mathcal{S}}_{\theta,h}(\boldsymbol{f}|_{X_h}) - \mathcal{S}(\boldsymbol{f})|_{X_h}\right\|_h \le \underbrace{\left\|\widehat{\mathcal{S}}_{\theta,h}(\boldsymbol{f}|_{X_h}) - \mathcal{S}_\theta(\boldsymbol{f})|_{X_h}\right\|_h}_{\text{grid/discretisation effect}} + \underbrace{\|\mathcal{S}_\theta(\boldsymbol{f})|_{X_h} - \mathcal{S}(\boldsymbol{f})|_{X_h}\|_h}_{\text{learned-operator error}}. \tag{15}$$

The first term is what changes when unchanged parameters are evaluated on a new point set; the second is the approximation error of the trained model.

The discretisation-invariance and approximation results of Kovachki et al. [10] provide theoretical control of these two contributions: the former concerns the change induced by admissible discretisations, while the latter concerns how accurately the neural operator can represent the target solution map. They do not make either term vanish for a finite graph, finite training set, or fixed neighbourhood radius, but they motivate the resolution transfer tests in Section 6.5.1. There, the same trained parameters are evaluated on finer grids; the observed errors combine the finite-grid effect with the learned-model error described by (15).

6.3 Message passing

A graph discretisation consists of nodes, edges, and a kernel shared across those edges. If all M points in X_h interacted, the kernel part of layer t would have the dense form

$$(\widehat{\mathcal{K}}_t \boldsymbol{v}_t)(x_i) = \frac{1}{M} \sum_{j=1}^{M} \boldsymbol{\kappa}_t(x_j, x_i) \odot \boldsymbol{v}_t(x_j). \tag{16}$$

In message-passing terminology [8], source node x_j sends the coordinate-conditioned message

$$\boldsymbol{m}_{ji} = \boldsymbol{\kappa}_t(x_j, x_i) \odot \boldsymbol{v}_t(x_j)$$

to target node x_i. The target averages its incoming messages and combines them with the local transformation $W_t \boldsymbol{v}_t(x_i)$, as shown in (14). Repeating the update lets information pass through intermediate nodes, so four local layers have a larger effective receptive field than a single layer.
The radius r retains every node but only creates the edge $x_j \to x_i$ when $\|x_i - x_j\|_2 \le r$. It therefore sparsifies the set of point pairs, giving the neighbourhood average in (14). The implemented Stokes GNO retains all $M = 729$ nodes but is not fully connected. With $r = 0.25$, an interior point on the 27×27 grid has approximately 137 neighbours, and four message-passing rounds propagate information beyond one neighbourhood.
The GNO composes four local kernels in a 32-dimensional latent space with pointwise channel-mixing maps. No individual $\boldsymbol{\kappa}_t$ is a physical Stokes Green matrix; only the complete composition defines the learned force-to-velocity operator.

6.4 Training setup

The Stokes equations, viscosity, finite-element discretisation, and random-force construction are exactly those described in Sections 2 and 3; they are not restated here. The GNO-specific data set contains 1000 training, 100 validation, and 200 test pairs on X_h. Each force and velocity component is divided by its training-set standard deviation without subtracting a mean, which preserves linearity. Training minimises the sample-relative discrete L^2 error defined in (10).

For the fixed nonhomogeneous boundary condition used in the top–bottom lid-driven experiment of Section 4.4, a separate affine model is trained, using the notation of the Green-kernel formulation:

$$\widehat{\boldsymbol{u}}(x_i; \boldsymbol{f}) = \widehat{\mathcal{S}}_{\theta,h}(\boldsymbol{f})(x_i) + \boldsymbol{u}_{\mathrm{hom},\phi}(x_i).$$

The response branch is the bias-free GNO and remains linear in the body force, whereas the four-hidden-layer coordinate MLP $\boldsymbol{u}_{\mathrm{hom},\phi}(x)$ learns the force-independent homogeneous velocity associated with the fixed wall data. The training data use top-wall velocity $(1, 0)^{\mathsf{T}}$, bottom-wall

Table 3: Configuration of the full-node linear Stokes GNO.

Ingredient	Value
Training/validation/test pairs	1000/100/200
Training grid	27×27 ($M = 729$)
Message-passing layers / latent width	4/32
Coordinate-kernel network	3 hidden layers, width 256, ReLU
Neighbourhood aggregation	mean, $r = 0.25$
Feature activation / bias	identity / none
Precision	FP32
Optimiser	Adam, 500 epochs
Learning rate	10^{-3}, halved every 100 epochs

velocity $(-1/3, 0)^{\mathsf{T}}$, stationary vertical walls, and a zero-force anchor. The wall velocities are fixed and are not model inputs; the resulting top–bottom lid-driven flow and error are reported in the results subsection below.

6.5 GNO results

6.5.1 No-slip operator test

The 27×27 test set contains 200 force–velocity pairs that were not used for training or validation. Two zero-shot tests evaluate the same trained parameters, without fine-tuning, on the corresponding 50×50 and 53×53 grids. The 53×53 grid is nested and contains all 27×27 training-grid points, whereas the non-nested 50×50 grid shares only the four corner points with the training grid.

Table 4: Mean sample-relative L^2 errors of the complete GNO velocity prediction. Each value is averaged over 200 held-out random-force samples.

Boundary condition	Evaluation	Mean relative error
No slip	27×27 training resolution	2.980%
No slip	50×50 non-nested zero-shot	4.635%
No slip	53×53 nested zero-shot	5.661%
Top–bottom lid-driven	27×27 training resolution	1.432%

At the training resolution, the no-slip model reaches a mean sample-relative velocity error of 2.980% on 200 held-out force–velocity pairs. The retained checkpoint was selected using the validation set, so this number is an independent test error rather than the training objective. Figure 11 shows a representative refined-grid sample: the FEM target and GNO prediction retain the same large-scale velocity pattern, while the pointwise error remains small over most of the domain and is more pronounced near parts of the boundary and in finer-scale structures.

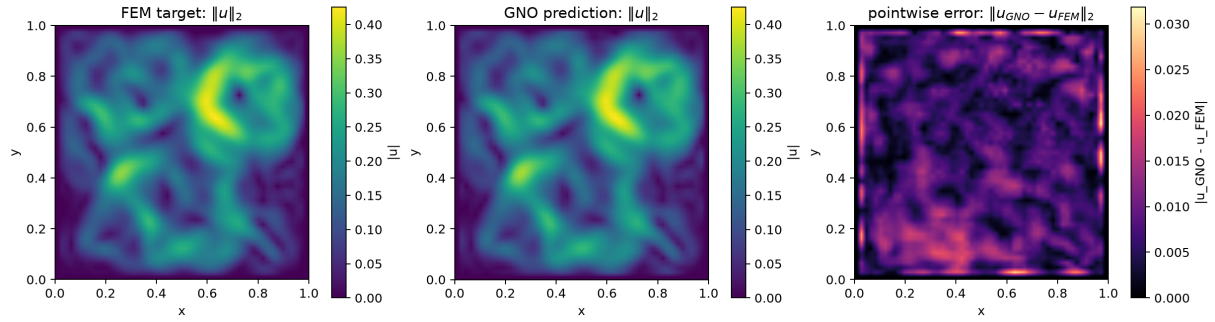


Figure 11: Nested $27 \to 53$ zero-shot evaluation for the no-slip problem: finite-element reference, GNO prediction, and pointwise error. The model is evaluated without fine-tuning.

The mean zero-shot errors are 4.635% on the non-nested 50×50 grid and 5.661% on the nested 53×53 grid. At first sight, one might expect the nested grid to perform better because it contains every training-grid point. The GNO, however, does not interpolate a velocity prediction already computed on the 27×27 grid. It constructs a new graph on the complete evaluation grid and recomputes all four message-passing layers at every node. Consequently, even the nodes shared by the 27×27 and 53×53 grids receive different aggregated messages.
For the fixed radius $r = 0.25$, an interior node has approximately 137 neighbours on the 27×27 graph, 481 on the 50×50 graph, and 529 on the 53×53 graph. The equal-weight neighbourhood mean therefore represents a different finite sum on each grid; node degrees, boundary neighbourhoods, and multi-layer message-passing paths change as well. These finite-grid effects need not vary monotonically with the number of nodes. In this experiment, their combined effect happens to produce the lower empirical error on the 50×50 graph. The aggregate error alone does not isolate one of these graph changes as the unique cause. Nesting preserves coordinates, but it does not preserve graph connectivity or the discrete message operator. This contributes to the first term in (15), while the second term remains the approximation error of the already-trained model.

This increase does not contradict the discretisation-invariance and approximation results of Kovachki et al. [10]. Those results establish convergence and approximation guarantees under their stated assumptions and in the appropriate limiting or controlled-discretisation setting; they do not assert identical errors for one finite, trained graph network on every finite grid, nor do they imply that errors on two finite grids must be ordered by whether one grid is nested. The experiment in Figure 11 is consistent with (15): discretisation transfer is possible, but it is not exactly invariant or necessarily monotone for this finite-radius, mean-aggregation implementation.

6.5.2 Top–bottom lid-driven operator test

The top–bottom lid-driven experiment uses the fixed boundary data from Section 4.4. On 200 held-out random forces, the complete affine model attains a mean sample-relative L^2 error of 1.432%, as reported in Table 4. This value is computed in exactly the same way as the no-slip errors: for every sample, both velocity components at all grid points are flattened, the L^2 error is divided by the L^2 norm of the corresponding FEM velocity, and the resulting relative errors are averaged over the test set. The two 27×27 values are therefore compatible summary statistics for comparing the complete predictions under the two boundary conditions.
The smaller total error for the top–bottom case should nevertheless be interpreted in the context

of its affine decomposition. Denote the force-dependent response by

$$\widehat{\boldsymbol{u}}_{\mathrm{resp},\theta}(x_i;\boldsymbol{f}) := \widehat{\mathcal{S}}_{\theta,h}(\boldsymbol{f})(x_i).$$

The complete top–bottom prediction is then

$$\widehat{\boldsymbol{u}}(x_i;\boldsymbol{f}) = \underbrace{\widehat{\boldsymbol{u}}_{\mathrm{resp},\theta}(x_i;\boldsymbol{f})}_{\text{force-dependent response}} + \underbrace{\boldsymbol{u}_{\mathrm{hom},\phi}(x_i)}_{\text{fixed wall-driven field}}.$$

The response branch has a hard zero-boundary mask, while the coordinate MLP supplies the fixed nonzero wall velocities and learns the associated zero-force flow throughout the domain. In the no-slip experiment the second term is absent, so the hard mask is applied to the complete prediction and enforces the no-slip condition exactly at every boundary node.

The fixed wall-driven component is present in every top–bottom target and is learned accurately by the coordinate MLP, while the relative-error denominator contains the complete velocity, including this component. Indeed, the force-dependent response alone has a mean relative error of 2.867%, which is much closer to the 2.980% no-slip result. Thus, the 1.432% total error does not indicate that the GNO response branch is approximately twice as accurate; it reflects the accuracy and scale of both terms in the complete affine prediction.

As a separate diagnostic, we plot the zero-force streamplot. The GNO model reproduces the recirculation and boundary values closely, as shown in Figure 12, with relative error 0.504%.

The largest discrepancies in the error plot are localised rather than a global loss of the cavity structure.

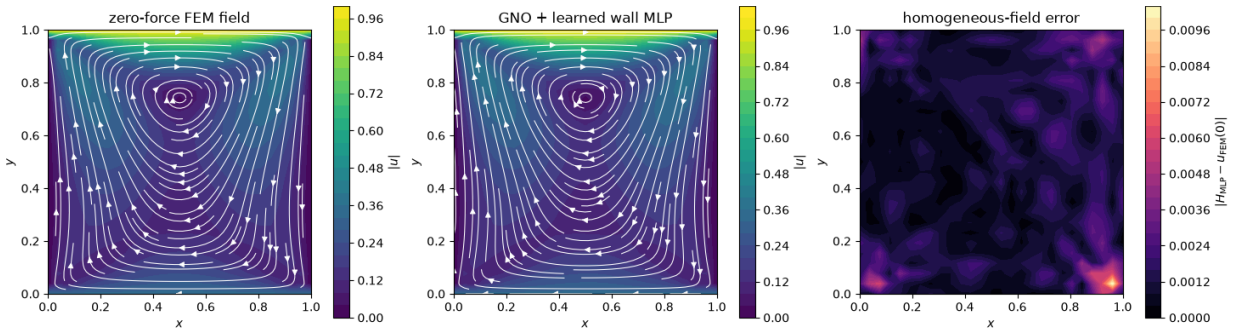


Figure 12: Zero-force top–bottom lid-driven diagnostic. The reported 0.504% is the relative error of the isolated boundary-driven field; the complete model has a mean error of 1.432% over 200 held-out random-force samples.

7 Stokes–Brinkman Bridge towards Shape Optimisation

7.1 Variable-coefficient flow model

A common route from Stokes flow to topology optimisation is to introduce an inverse-permeability term controlled by a spatial design field. Borrvall and Petersson use a generalised Stokes equation for this purpose and interpret the coefficient as a velocity penalty in regions with low permeability [2, p. 78]. The experiment considered here uses the Stokes–Brinkman system

$$\begin{aligned} -\mu\Delta\boldsymbol{u} + \alpha(q)\boldsymbol{u} + \nabla p &= \boldsymbol{f} \quad \text{in } \Omega, \\ \nabla\cdot\boldsymbol{u} &= 0 \quad \text{in } \Omega, \\ \boldsymbol{u} &= 0 \quad \text{on } \partial\Omega, \end{aligned} \qquad \alpha(q) = \frac{5\mu}{2q^2}. \tag{17}$$

The coefficient in (17) matches the two-dimensional generalised Stokes term obtained from a thin-layer reduction [2, p. 83, Eqs. (12)–(13)]. Allowing the field to vary in space provides a design variable whose permeability changes locally, which motivates its use as a bridge towards flow optimisation [2, p. 84].

In the experiment, $\Omega = (0,1)^2$, $\mu = 10^{-2}$, and $q(x) \in [q_{\min}, q_{\max}] = [0.05, 1]$. Each design field is generated as

$$q(x) = q_{\min} + (q_{\max} - q_{\min})\,\sigma(2z(x))\,, \qquad \sigma(t) = \frac{1}{1+\mathrm{e}^{-t}}, \tag{18}$$

where z is a smooth Gaussian random field with target wavelength 0.2. The body-force components are drawn independently with target wavelength 0.1, as in Section 3. Reference solutions use a Taylor–Hood discretisation on 57×57 square cells and are evaluated on a 20×20 Cartesian learning grid. There are 400 force/design/velocity samples, split into 320 training and 80 held-out samples.

7.2 Local-knowledge kernel ansatz

The purpose of the experiment is to test whether the variable-coefficient solution operator can be learned from pointwise information about the design field. The Green network therefore uses only the source and target coordinates and the two corresponding local design values:

$$(x_1, x_2, y_1, y_2, q(x), q(y)) \longmapsto \big(G_{11}, G_{12}, G_{21}, G_{22}\big).$$

It therefore predicts

$$\widehat{\boldsymbol{u}}(x) = \int_\Omega \boldsymbol{G}_\theta(x, y, q(x), q(y))\, \boldsymbol{f}(y)\, \mathrm{d}y. \tag{19}$$

On the learning grid, the integral in (19) is evaluated with the same tensor-product trapezoidal quadrature as in (9). The network parameters are fitted by minimising the mean sample-wise relative discrete velocity error

$$\mathcal{E}_{\mathrm{B}}(\theta) = \frac{1}{N}\sum_{n=1}^{N} \frac{\Big(\sum_{i=1}^{M} \|\widehat{\boldsymbol{u}}^{(n)}(x_i) - \boldsymbol{u}^{(n)}(x_i)\|_2^2\Big)^{1/2}}{\Big(\sum_{i=1}^{M} \|\boldsymbol{u}^{(n)}(x_i)\|_2^2\Big)^{1/2}}, \tag{20}$$

which is the error definition in (10), specialised to the Stokes–Brinkman data. Table 5 summarises the data and neural-network configuration.

Table 5: Configuration of the locally conditioned Stokes–Brinkman Green network.

Quantity	Configuration
Domain and viscosity	$\Omega = (0,1)^2$, $\mu = 10^{-2}$
Design-field range	$q(x) \in [0.05, 1]$
Learning data	400 samples: 320 training, 80 held out
Learning grid	20×20 points
Network input	$(x_1, x_2, y_1, y_2, q(x), q(y)) \in \mathbb{R}^6$
Network output	$(G_{11}, G_{12}, G_{21}, G_{22}) \in \mathbb{R}^4$
Hidden architecture	four layers, width 128, SiLU activation
Optimiser	Adam, 2200 epochs, batch size 50
Learning rate	1.5×10^{-3}
Precision schedule	1600 FP32 epochs, then 600 FP64 epochs
Recorded runtime	64 427.48 s (17 h 53 min 47 s) on RTX 3070

This input contract is deliberately local. In contrast, the Green matrix of (17) depends on the full coefficient field q because changes in permeability anywhere in the domain can alter the global incompressible flow and pressure. Thus two design fields can agree at $q(x)$ and $q(y)$ while inducing different responses between the same source–target pair.

7.3 Numerical assessment

Figure 13 confirms the limitation of the local ansatz. The held-out error reaches its minimum of 15.47% at epoch 1165, where the training error is 13.00% and the generalisation gap is therefore 2.47 percentage points. Training continues to reduce the error on the training set, but the held-out error subsequently rises. At epoch 2200 the training and held-out errors are 11.36% and 16.35%, respectively, so the gap has widened to 4.99 percentage points. Switching to double precision at epoch 1600 does not reverse this trend. The complete run takes 64 427.48 seconds, or approximately 17.90 hours.

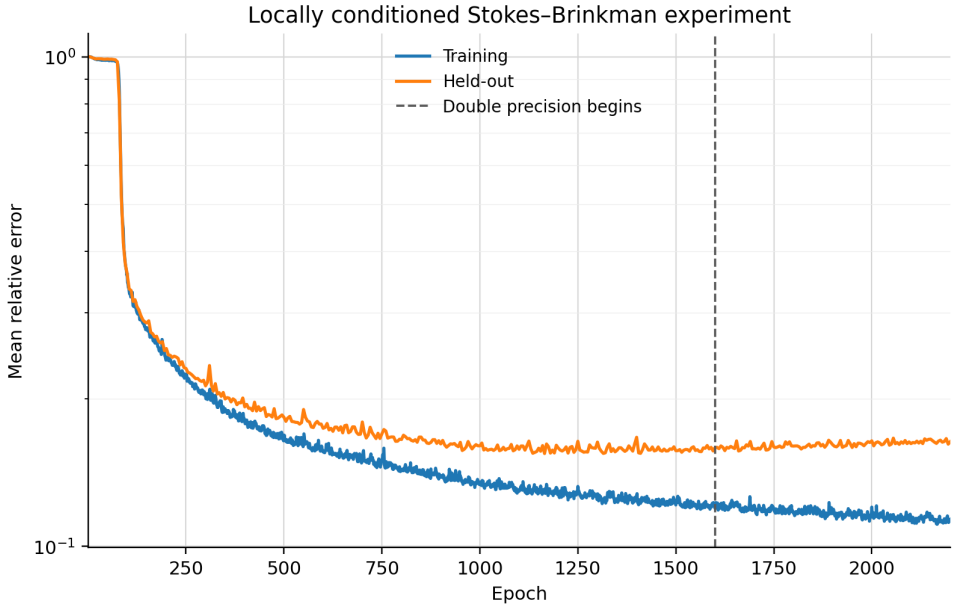


Figure 13: Training and held-out error for the locally conditioned Stokes–Brinkman Green network. The held-out curve reaches its minimum at epoch 1165 while the training curve continues to decrease, and the train–test gap widens thereafter. The dashed line marks the switch from single to double precision.

The resulting model is not sufficiently accurate for a subsequent optimisation loop and is therefore not pursued further in its present form. The widening generalisation gap shows increasing overfitting, while the magnitude of the held-out error exposes a more fundamental limitation of the model class. We attribute this limitation to the exclusive use of pointwise coefficient information: Stokes–Brinkman flow is globally coupled through viscosity, incompressibility, pressure, and the spatially varying resistance. Knowledge of $q(x)$ and $q(y)$ cannot in general determine the response between x and y because it omits the coefficient field along alternative flow paths and throughout the rest of the domain. A viable design-field-dependent surrogate should therefore receive global information about q, for example through a graph operator or a learned global encoding, rather than only the two endpoint values.

The final kernel visualisation in Figure 14 exhibits a localised source response, but its visually plausible structure is not sufficient to compensate for the large held-out solution error.

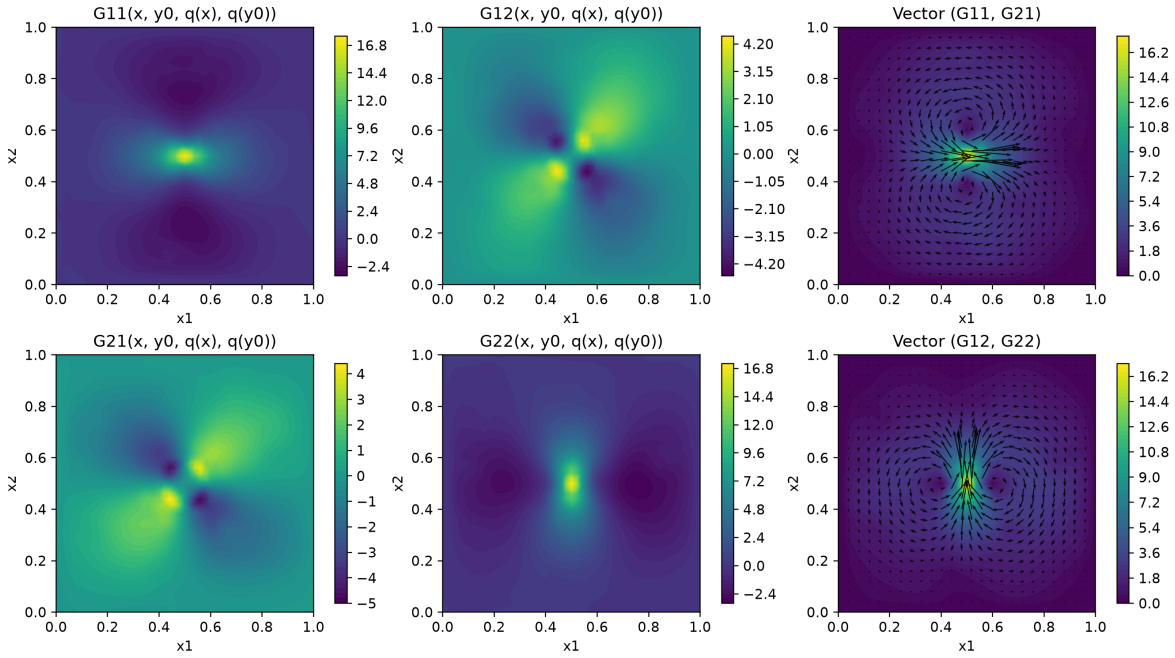


Figure 14: Locally conditioned Stokes–Brinkman kernel at epoch 2200 for $y_0 = (0.5, 0.5)$ and the displayed design-field realisation.

8 Summary and Conclusions

This project compared two data-driven approximations of the stationary Stokes force-to-velocity operator and examined a possible extension to a variable-coefficient flow problem. All models were trained on velocity fields generated with stable Taylor–Hood finite-element discretisations. The experiments assessed predictive accuracy, activation choice, resolution transfer, nonhomogeneous boundary data, and the information required to represent a spatially varying coefficient.

For direct Green-kernel learning on the no-slip Stokes problem, the width-128 SiLU network attained the smallest held-out error, 0.6626%. The corresponding tanh, rational, and ReLU results were 0.8140%, 2.1554%, and 2.6439%, respectively. Because the rational model was restricted to width 41 by the available VRAM, it was not capacity matched to the other networks. When the best no-slip model was evaluated on 200 new samples, its quadrature-weighted error increased from 0.5789% on the 27×27 learning resolution to 2.5611% on a 50×50 grid and 2.6277% on a 53×53 grid. This behaviour suggests sensitivity to the training grid and motivates training across several resolutions.
For the corrected top–bottom lid-driven problem, the FP64 SiLU model reached a held-out error of 1.7654% after 3000 epochs. Extending the same training procedure to 10 000 epochs in FP32, without an additional boundary penalty, reduced the held-out error to 0.6200%. Since both the precision and the training duration changed, this comparison does not isolate a precision effect, but it demonstrates that the extended FP32 configuration is both accurate and computationally practical on the available hardware.

The divergence-free Green-kernel formulation guarantees incompressibility exactly by construction and achieves predictive accuracy comparable to the direct velocity-based model. However, this increased physical consistency comes at the cost of higher computational and memory requirements due to the repeated evaluation of spatial derivatives through automatic differentiation.

The graph neural operator provided a complementary, explicitly linear approximation. Its no-slip test error was 2.980% at the 27×27 training resolution and increased to 4.635% and 5.661% on the 50×50 and 53×53 zero-shot grids. The fixed-radius graph can therefore be evaluated on new discretisations without retraining, but its finite message-passing operator is not exactly resolution invariant. For the top–bottom lid-driven problem, the complete affine model attained 1.432%, while the force-dependent response alone had an error of 2.867%. The lower complete error partly reflects the accurately learned wall-driven component and should not be interpreted as a twofold improvement of the response branch.
The Stokes–Brinkman experiment produced a different conclusion. Conditioning the kernel only on $q(x)$ and $q(y)$ gave a best held-out error of 15.47% and a final error of 16.35% after a complete runtime of 64 427.48 seconds (17 h 53 min 47 s). The train–test gap widened from 2.47 percentage points at the best held-out epoch to 4.99 percentage points at the end of training. Local endpoint values are therefore insufficient for this globally coupled variable-coefficient operator. A future surrogate should encode the complete design field, for example with a graph operator, while an incompressibility-preserving parameterisation remains a natural improvement for the Green-network models.

Taken together, the results show that direct Green-kernel learning gives the best accuracy in the present fixed-grid experiments, whereas the graph neural operator offers a natural mechanism for evaluation on different point sets. The numerical values should not be treated as a strictly controlled ranking of the two approaches because their training-set sizes and optimization schedules differ. The principal next steps are multi-resolution training, global conditioning for variable coefficients, incorporation of velocity based errors for the divergence free formulation and direct enforcement of symmetries of the homogeneous Green's function.

Declaration of AI Assistance

AI-assisted tools were used in this project to support software development, including code generation, debugging, and documentation, and to assist with drafting, revising, and proofreading the report. All suggestions and outputs were critically reviewed, verified, and adapted by the authors. The authors take full responsibility for the correctness, integrity, and final content of the code, analyses, and report.

References

[1] Daniele Boffi, Franco Brezzi, and Michel Fortin. *Mixed Finite Element Methods and Applications*, volume 44 of *Springer Series in Computational Mathematics*. Springer, Berlin, Heidelberg, 2013.

[2] Thomas Borrvall and Joakim Petersson. Topology optimization of fluids in Stokes flow. *International Journal for Numerical Methods in Fluids*, 41(1):77–107, 2003.

[3] Nicolas Boullé, Christopher J. Earls, and Alex Townsend. Data-driven discovery of Green's functions with human-understandable deep learning. *Scientific Reports*, 12(1):4824, 2022.

[4] Nicolas Boullé, Yuji Nakatsukasa, and Alex Townsend. Rational neural networks. In *Advances in Neural Information Processing Systems*, volume 33, pages 14243–14253, 2020.

[5] Jongkeun Choi and Doyoon Kim. Estimates for Green functions of Stokes systems in two-dimensional domains. *Journal of Mathematical Analysis and Applications*, 471(1–2):102–125, 2019.

[6] S.-I. Filip, A. Javeed, and L. N. Trefethen. Smooth random functions, random ODEs, and gaussian processes. *SIAM Review*, 61(1):185–205, 2019.

[7] Henrik Finsberg and Jørgen Schartum Dokken. scientificcomputing/scifem, December 2024.

[8] Justin Gilmer, Samuel S. Schoenholz, Patrick F. Riley, Oriol Vinyals, and George E. Dahl. Neural message passing for quantum chemistry. In *Proceedings of the 34th International Conference on Machine Learning*, volume 70 of *Proceedings of Machine Learning Research*, pages 1263–1272. PMLR, 2017.

[9] Juncai He, Xinliang Liu, and Zitong Tian. Divergence-free linearized neural networks: Integral representation and optimal approximation rates. *arXiv preprint*, 2026.

[10] Nikola Kovachki, Zongyi Li, Burigede Liu, Kamyar Azizzadenesheli, Kaushik Bhattacharya, Andrew Stuart, and Anima Anandkumar. Neural operator: Learning maps between function spaces with applications to PDEs. *Journal of Machine Learning Research*, 24(89):1–97, 2023.

[11] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, et al. Pytorch: An imperative style, high-performance deep learning library. In *Advances in Neural Information Processing Systems*, volume 32, 2019.

[12] Jack Richter-Powell, Yaron Lipman, and Ricky T. Q. Chen. Neural conservation laws: a divergence-free perspective. In *Proceedings of the 36th International Conference on Neural Information Processing Systems*, NIPS '22, Red Hook, NY, USA, 2022. Curran Associates Inc.

[13] Cedric Taylor and Peter Hood. A numerical solution of the Navier–Stokes equations using the finite element technique. *Computers & Fluids*, 1(1):73–100, 1973.