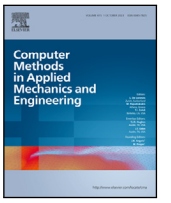
Comput. Methods Appl. Mech. Engrg. 417 (2023) 116351

Contents lists available at ScienceDirect

Comput. Methods Appl. Mech. Engrg.

journal homepage: www.elsevier.com/locate/cma



Physics-informed graph neural network emulation of soft-tissue mechanics

David Dalton *, Dirk Husmeier, Hao Gao

School of Mathematics and Statistics, University of Glasgow, University Avenue, Glasgow, G12 8QQ, United Kingdom

ARTICLE INFO

Keywords:
Soft-tissue mechanics
Graph neural networks
Physics-informed machine learning

ABSTRACT

Modern computational soft-tissue mechanics models have the potential to offer unique, patient-specific diagnostic insights. The deployment of such models in clinical settings has been limited however, due to the excessive computational costs incurred when performing mechanical simulations using conventional numerical solvers. An alternative approach to obtaining results in clinically relevant time frames is to make use of a computationally efficient surrogate model, called an emulator, in place of the numerical simulator. In this work, we propose an emulation framework for soft-tissue mechanics which builds on traditional approaches in two ways. Firstly, we use a Graph Neural Network (GNN) to perform emulation. GNNs can naturally handle the unique soft-tissue geometry of a given patient, without requiring any low-order approximations to be made. Secondly, the emulator is trained in a physics-informed manner to minimise a potential energy functional, meaning that no costly numerical simulations are required for training. We present results showing that our framework allows for highly accurate emulation for a range of soft-tissue mechanical models, while making predictions several orders of magnitude more quickly than the simulator.

1. Introduction

Major developments have been made recently in computational soft-tissue mechanics [1]. Models of soft-tissue systems and organs, such as the liver [2], heart [3], brain [4], cartilages [5] and blood vessels [6] have grown from simplified idealisations to sophisticated models that allow for highly realistic, patient specific simulations to be made in silico. The mechanical functioning of soft-tissue bodies is critical to the optimal functioning of the organism itself [7]. For example, the loss of myocytes in the myocardium after myocardial infarction can result in an imbalanced stress/strain micro-environment, impeding the efficiency of the heart's pump cycle [8]. Computational models have the potential for transformative clinical advances in this area, by allowing for patient-specific, *synthetic* simulation data to be generated. This enables experiments and analyses to be performed beyond what is possible with purely *observational* data [9,10], including for example sensitivity analyses [11,12], inverse problems [13–15] and treatment design optimisation [16–18].

Simulation of soft-tissue mechanics involves numerically solving for the dynamics which satisfy the underlying physiological partial differential equations (PDEs), which are in general non-linear. The finite element method (FEM) [19] has become the standard numerical method to solve complex systems described by PDEs, due to its capability of handling realistic geometries, material nonlinearity at multiscales, time-varying boundary conditions and multiphysics coupling, with well-developed commercial and open-source software available. However, the FEM is too expensive to provide real-time results when hundreds or thousands of high fidelity simulations are required, for instance for solving inverse problems, a computational bottleneck which has limited the

* Corresponding author.
E-mail address: david.dalton@glasgow.ac.uk (D. Dalton).

https://doi.org/10.1016/j.cma.2023.116351
Received 25 April 2023; Received in revised form 2 August 2023; Accepted 4 August 2023
Available online 16 September 2023

Nomenclature

Ω_0	Reference configuration
Ω	Current configuration
$\boldsymbol{X}$	Coordinates in reference configuration
$\boldsymbol{x}$	Coordinates in current configuration
$\boldsymbol{u}$	Displacement
$\boldsymbol{\sigma}$	Cauchy stress tensor
$\boldsymbol{b}$	Body force in current configuration
Ω^d	Dirichlet boundary in current configuration
$\boldsymbol{u}_d$	Prescribed displacement on Dirichlet boundary
Ω^σ	Neumann boundary in current configuration
$\boldsymbol{n}$	Surface normal vector in current configuration
$\boldsymbol{t}$	Traction force on Neumann boundary
$\boldsymbol{F}$	Deformation gradient
$\boldsymbol{C}$	Right Cauchy-Green tensor
J	Determinant of $\boldsymbol{F}$
I_1	First invariant of $\boldsymbol{F}$
Ψ	Strain energy density function
Π	Total potential energy
$\mathcal{G}/\tilde{\mathcal{G}}$	Graph / augmented graph
$\mathcal{V}/\tilde{\mathcal{V}}$	Graph nodes / augmented graph nodes
$\boldsymbol{v}_i$	Node feature vector
$\mathcal{E}/\tilde{\mathcal{E}}$	Graph edges / augmented edges
$\boldsymbol{e}_{i\to j}$	Edge feature vector
$\boldsymbol{\theta}$	Global graph parameters / material parameters
$\boldsymbol{\omega}$	Trainable emulator parameters
$\boldsymbol{m}^k_{j\to i}$	Message from node j to node i at processor step k
$\hat{\boldsymbol{u}}$	Emulator predicted displacement
$\boldsymbol{U}$	Array of displacements from simulator
$\hat{\boldsymbol{U}}$	Array of displacements from emulator

deployment of soft-tissue mechanic models in clinical decision support [9]. Clearly then, there is need for more efficient simulation methods to be developed.

The bottleneck imposed by traditional numerical simulation methods is common to high fidelity models across computational physics, which has lead in turn to the development of the field of surrogate modelling, or emulation [20,21]. An *emulator* is a machine learning model that approximates the numerical solver, or *simulator*, while incurring significantly lower expense at prediction time. Commonly used emulation approaches include Gaussian processes [22], polynomial chaos [23] and deep neural networks, both fully connected and convolutional [24, Chapters 13–14]. These approaches have been successfully deployed for emulation of a wide range of physical systems, with recent examples including computational fluid dynamics [25], aerodynamics [26], climate models [27] and nonlinear dynamical systems [28].

Emulators are typically trained in a data-driven manner against a large set of simulation results from the numerical solver [21]. There are disadvantages to the data-driven approach, however. Firstly, a large dataset may be required to train an accurate surrogate model, which will be expensive to obtain. Also, with a purely data-driven training, any *a-priori* known underlying physical properties of the system under consideration (which could include momentum conservation, for instance) are not explicitly incorporated into the emulator. Instead, such properties are only implicitly incorporated via the simulation data. Physics-informed machine learning approaches constitute an alternative approach to emulation, whereby properties or constraints of the underlying physical system are explicitly accounted for in the structure of the emulator, and/or into the training procedure. Early work in the field includes the use of both neural networks [29–31] and Gaussian processes [32,33] for solving forward and inverse problems involving differential equations. A major development was the seminal work of Raissi et al. (2019) [34], which introduced physics-informed neural networks (PINNs). PINNs explicitly incorporate the underlying equations of the model (typically PDEs) into the training of the neural network surrogate model. PINNs can be trained on a loss function solely derived from the underlying PDEs, or in a multi-task learning framework where observational data is also included. PINNs were originally implemented using the strong form of the underlying PDEs, with all required partial derivatives computed using automatic differentiation, but have subsequently been extended to model PDE systems using variational [35] and energy minimisation [36,37] approaches. PINNs have become one of the most highly researched areas of scientific machine learning, for purposes including forward, inverse, control and model

discovery problems across a range of disciplines. To give a handful of examples, this includes computational fluid dynamics [38], hyperelasticity [36], electromagnetics [39] and molecular dynamics [40]. Several detailed survey papers on PINNs are also available, providing a comprehensive overview of the field [41,42].

PINNs have also been applied in soft-tissue mechanics, for emulation of the dynamics of the left ventricle of the heart for the entire cardiac cycle [43], and during the passive diastolic phase [44], with both approaches making use of fully connected neural networks (FCNNs). One issue with applying FCNNs in soft-tissue mechanics is that the geometries of individual organs or vessel networks vary across the population, meaning that a different computational mesh representation must be generated for each subject. This makes it difficult to construct an accurate emulator using a traditional approach, especially one that can generalise to geometries not seen in the training phase, due to the so-called curse of dimensionality. In Buoso et al. (2021) [43], a low rank approximation to both the left ventricle (LV) geometry and displacement field is required to overcome this issue and allow the FCNN to generalise to new LV geometries, while in Zhang et al. (2022) [44], only a simple cuboidal geometry is considered, not a real heart geometry.

A new generation of surrogate models has emerged in recent years based on Graph Neural Networks (GNNs), which overcome many of the challenges faced by traditional approaches in modelling high fidelity simulation data. GNNs form an extremely active area of research in modern machine learning, with applications to graph structured data in physics [45,46], chemistry [47], biology [48], among other areas [49–51]. Various approaches to GNN architecture design have been proposed in the literature, including spectral methods based on the graph Laplacian and spatial graph convolution approaches [24, Chapter 20]. For the specific case of surrogate modelling, message passing GNNs have proven particularly successful [45,52,53]. The key step required to apply a GNN for surrogate modelling is to represent the physical system under consideration in the form of a graph. For a mesh based simulation for instance, the vertices of the graph representation can be taken as simply the nodes of the underlying mesh, while the graph topology can be extracted from the mesh topology. A message passing GNN then learns the underlying physical dynamics using a series of message passing steps between neighbouring nodes of the graph representation. This allows a learned representation of each node to be found, from which system dynamics are decoded. There are a number of advantages to using GNNs for surrogate modelling. For instance, the architecture of the GNN can be designed to exactly satisfy *a-priori* known invariances or equivariances of the system under consideration [54]. In addition, GNNs have been demonstrated to operate well on very large datasets [45] and generalise to systems not seen in the training phase [55]. This is particularly useful for soft-tissue mechanics, as the specific geometry will be unique from subject to subject. Despite the potential for application of GNN surrogate models in soft-tissue mechanics, there has been relatively little development in this area. One exception is a recent publication by the present authors on data-driven GNN emulation of passive cardiac mechanics [56], the results of which demonstrated a clear gain in accuracy for a GNN over the results obtained using an FCNN.

1.1. Contributions

In this manuscript we present an emulation framework for soft-tissue mechanics based on GNNs. As we have demonstrated in previous work [56], GNN emulation is particularly suited to soft-tissue mechanics, where the geometries of the bodies under consideration can have highly irregular shapes. Here we build upon our existing work by making use of physics-informed training through application of the principle of minimum total potential energy, in place of a data-driven approach based on numerical simulation data. This is illustrated schematically in Fig. 1. Training on a potential energy objective function is enabled by the use of transformation barrier functions which explicitly incorporate known physical constraints and stabilise the objective. A range of realistic soft-tissue mechanics models are considered, including highly non-linear, fibre reinforced materials. Experimental results demonstrate that strong out-of-sample accuracy can be achieved, i.e. the emulator can generalise to input points not seen during the training phase. Additional experimental comparisons indicate that physics-informed training allows for a more physically realistic deformation to be consistently captured in comparison to data-driven training. Finally, significant computational savings at prediction time are made over the finite-element based simulator.

The manuscript is laid out as follows; Section 2 first describes the methods used, including the underlying mechanics framework considered and the proposed physics-informed emulation approach. Section 3 then presents emulation results for a number of mechanics models. Section 4 discusses these results, before Section 5 concludes.

2. Methods

2.1. Mechanics framework

Consider a continuum body made of a hyperelastic material. Let $\Omega_0 \subset \mathbb{R}^3$ be the reference configuration of the body, comprised of material points $\boldsymbol{X} = (X_1, X_2, X_3)^\top \in \Omega_0$. Under external loading, the body can deform into current configuration Ω, comprised of material points $\boldsymbol{x} = (x_1, x_2, x_3)^\top$. The material points in the current and reference configurations are related as $\boldsymbol{x} = \chi(\boldsymbol{X}, t) = \boldsymbol{X} + \boldsymbol{u}$, with χ the (invertible) motion map and $\boldsymbol{u} = \boldsymbol{u}(\boldsymbol{X}, t)$ the displacement. The present work considers quasi-static, nonlinear boundary value problems, in which case the displacement field is found as that which satisfies momentum balance subject to prescribed boundary conditions, i.e.,

$$\begin{cases} \nabla \cdot \boldsymbol{\sigma} + \boldsymbol{b} = \boldsymbol{0} & \text{in } \Omega, \\ \boldsymbol{u} = \boldsymbol{u}_d & \text{on } \partial\Omega^d, \\ \boldsymbol{\sigma} \cdot \boldsymbol{n} = \boldsymbol{t} & \text{on } \partial\Omega^\sigma. \end{cases} \tag{1}$$

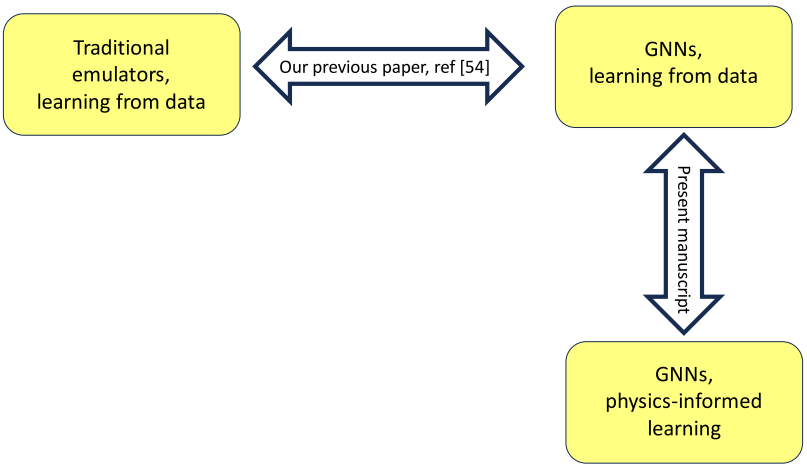


Fig. 1. Schematic illustration of the distinction between previous work [56] and the present manuscript. In [56], different emulation methods were considered (both traditional methods and GNNs) using the same data-driven learning paradigm. In the present work, GNN emulation making use of physics-informed training is considered, with experimental comparisons to data-driven training.

Here $\boldsymbol{\sigma} \in \mathbb{R}^{3\times3}$ is the Cauchy stress tensor, $\boldsymbol{b} \in \mathbb{R}^{3\times1}$ is the body force, $\boldsymbol{u}_d \in \mathbb{R}^{3\times1}$ is the prescribed displacement on the Dirichlet boundary $\partial\Omega^d$, $\boldsymbol{n} \in \mathbb{R}^{3\times1}$ is the surface normal vector and $\boldsymbol{t} \in \mathbb{R}^{3\times1}$ the applied traction density to the Neumann boundary $\partial\Omega^\sigma$. An important quantity for quantifying the forward map from the reference to the current configuration is the deformation gradient tensor, which is defined as $\boldsymbol{F} \in \mathbb{R}^{3\times3} = \frac{\partial \boldsymbol{x}}{\partial \boldsymbol{X}} = \nabla \boldsymbol{u} + \boldsymbol{I}$, with $\boldsymbol{I}$ the identity tensor. For a hyperelastic material, the stress $\boldsymbol{\sigma}$ and strain (or $\boldsymbol{F}$) tensors can be related by a constitutive law, Ψ, i.e. $\boldsymbol{\sigma} = J^{-1}\left(\frac{\partial \Psi}{\partial \boldsymbol{F}}\right)^T$, where $J = \det(\boldsymbol{F})$. Three constitutive laws are considered in this work: Neo-Hookean [57], Holzapfel–Ogden [3] and Guccione [58].[1]

The Neo-Hookean model is derived from first principles on the properties of cross-linked polymer chains, and is suitable for isotropic plastic and rubber-like materials [57]. The strain energy density for a compressible Neo-Hookean material is given by

$$\textbf{Neo-Hookean}: \quad \Psi(\boldsymbol{F}, \boldsymbol{\theta}) = \frac{1}{2}\lambda[\log(J)]^2 - \mu\log(J) + \frac{1}{2}\mu\left(I_1 - 3\right). \tag{2}$$

Here, $I_1 = \mathrm{tr}(\boldsymbol{C})$ is the first invariant with $\boldsymbol{C} = \boldsymbol{F}^\top\boldsymbol{F}$ the right-Cauchy–Green deformation tensor. The parameters λ and μ are the Lame material parameters, which are denoted collectively as $\boldsymbol{\theta}$. The Neo-Hookean model can be equivalently parameterised in terms of Young's modulus E and Poisson ratio ν, which are related to the Lame parameters as:

$$\lambda = \frac{E\nu}{(1+\nu)(1-2\nu)}, \quad \mu = \frac{E}{2(1+\nu)}. \tag{3}$$

The Holzapfel–Ogden (H-O) model is a phenomenologically derived anisotropic hyperelastic material model for describing the passive response of the myocardium [3]. The fully incompressible strain energy density function of a transversely isotropic H-O material is given by

$$\textbf{H-O}: \quad \Psi(\boldsymbol{F}, \boldsymbol{\theta}) = \frac{a}{2b}\left(e^{b(I_1-3)} - 1\right) + \frac{a_\mathrm{f}}{2b_\mathrm{f}}\left(e^{b_\mathrm{f}(\max(I_{4\mathrm{f}},1)-1)^2} - 1\right). \tag{4}$$

Here I_1 is defined as above, while the transversely isotropic invariant $I_{4\mathrm{f}}$ is equal to $\boldsymbol{f}_0 \cdot (\boldsymbol{C}\boldsymbol{f}_0)$, where $\boldsymbol{f}_0 \in \mathbb{R}^{3\times1}$ is the unit vector in the myofibre direction in the reference configuration. The material parameters are $\boldsymbol{\theta} = (a, b, a_\mathrm{f}, b_\mathrm{f})$. The max() term in Eq. (4) ensures that the myofibres only support extension but not compression. The use of the transversely isotropic version of the H-O material is motivated by the results of an earlier sensitivity study of the full H-O law [12]. In our numerical implementation, we consider the myocardium nearly incompressible, which is a widely used practice in cardiac models, using the F-bar method to Eq. (4) to ensure numerical stability [59]. By decomposing $\boldsymbol{F}$ into a deviatoric ($\bar{\boldsymbol{F}} = J^{-1/3}\boldsymbol{F}$) and volumetric component ($J^{1/3}\boldsymbol{I}$), the F-bar method will relax the near-incompressibility constraint. Accordingly, we have

$$\bar{\boldsymbol{C}} = \bar{\boldsymbol{F}}^T\bar{\boldsymbol{F}}, \quad \bar{I}_1 = \mathrm{tr}(\bar{\boldsymbol{C}}), \quad \bar{I}_4 = \bar{\boldsymbol{F}}\boldsymbol{f}_0 \cdot \bar{\boldsymbol{F}}\boldsymbol{f}_0, \tag{5}$$

and the H-O model is modified as

$$\Psi(\boldsymbol{F}, \boldsymbol{\theta}) = \frac{a}{2b}\left(e^{b(\bar{I}_1-3)} - 1\right) + \frac{a_\mathrm{f}}{2b_\mathrm{f}}\left(e^{b_\mathrm{f}(\max(\bar{I}_{4\mathrm{f}},1)-1)^2} - 1\right) + \frac{\mathcal{P}}{2}(J-1)^2, \tag{6}$$

[1] Experiments involving the Guccione law are performed in Appendix A.

where the final term in the equation, known as the penalty function, enforces the near-incompressibility constraint of the myocardium via the Lagrange multiplier $\mathcal{P} \in \mathbb{R}^+$, which may also be referred to as the bulk modulus. In this study, $\mathcal{P}$ is pre-set to be a fixed value, however $\mathcal{P}$ can also be treated as an unknown field variable based on the multi-field variable principles [60, Chapter 8].

The Guccione model [58] also describes the mechanical response of the passive myocardium, with the strain energy density function given by

$$\begin{aligned} \textbf{Guccione}: \quad & \Psi(\boldsymbol{F},\boldsymbol{\theta}) = \frac{C}{2}\left(e^{Q}-1\right) + \frac{\mathcal{P}}{2}(J-1)^2, \text{ with} \\ & Q = b_{\mathrm{f}}\bar{E}_{11}^2 + b_{\mathrm{t}}\left(\bar{E}_{22}^2+\bar{E}_{33}^2+\bar{E}_{23}^2+\bar{E}_{32}^2\right) + b_{\mathrm{fs}}\left(\bar{E}_{12}^2+\bar{E}_{21}^2+\bar{E}_{13}^2+\bar{E}_{31}^2\right). \end{aligned} \tag{7}$$

Here $\bar{\boldsymbol{E}} = \frac{1}{2}(\bar{\boldsymbol{C}} - \boldsymbol{I})$ is the Green–Lagrange strain tensor, and we have material parameter vector $\boldsymbol{\theta} = (C,\ b_{\mathrm{f}},\ b_{\mathrm{t}},\ b_{\mathrm{fs}})$.

2.2. Numerical methods

In this work, we propose a GNN emulation approach for boundary value problems of the form of Eq. (1), with particular emphasis on applications in soft-tissue mechanics. Emulation results are benchmarked with numerical simulations obtained using the nonlinear finite-element method. Before a numerical solution for the displacement field can be found using either the FEM simulator or GNN emulator, a number of processing steps are required.

Firstly, the nonlinear boundary-value problem (BVP) of Eq. (1) is very challenging to be solved analytically, in particular for complex geometries. In this study, we consider the quasi-static BVP to be a conservative mechanical system, which requires the existence of an energy functional Π for both the stresses of a deformable body and the loads. Such an assumption is commonly used in solid mechanics. Based on the stationary energy principle, and further treating $\boldsymbol{u}$ as the only unknown, then of all the admissible $\boldsymbol{u}$, we seek the solution that minimises the total potential energy [60, Chapter 8]. In other words, the first variation of the total potential energy $\delta\Pi$ needs to vanish in static equilibrium. In the case of a hyper-elastic continuum, the total potential energy Π of Eq. (1) is given by:

$$\Pi = \int_{\Omega_0} \Psi dV - \int_{\Omega_0} \boldsymbol{b}_0 \cdot \boldsymbol{u} dV - \int_{\partial\Omega_0^{\sigma}} \boldsymbol{t}_0 \cdot \boldsymbol{u} dA, \tag{8}$$

in which $\boldsymbol{b}_0(\boldsymbol{X}) = \boldsymbol{b}(\chi^{-1}(\boldsymbol{x},t),t=0)$ is the body force density with respect to the reference configuration, and $\boldsymbol{t}_0(\boldsymbol{X}) = \boldsymbol{t}(\chi^{-1}(\boldsymbol{x},t),t=0)$ is the traction force density with respect to the reference configuration. Mathematically, the desired solution can be obtained by requiring the directional derivative (or *Gateaux* derivative) of Π with respect to $\boldsymbol{u}$ to vanish in all directions $\delta\boldsymbol{u}$, that is

$$\delta\Pi(\boldsymbol{u},\delta\boldsymbol{u}) = \frac{d}{d\epsilon}\Pi(\boldsymbol{u}+\epsilon\delta\boldsymbol{u})|_{\epsilon=0} = 0. \tag{9}$$

Further details of the principle of stationary potential energy can be found in [60, Chapter 8].

The FEM is the most commonly used approach to numerically solve this type of BVP. Before a solution can be found using FEM, the reference configuration of the body under consideration needs to be discretised with n elements, denoted as $\{\mathcal{T}_h\}_{h>0}$. Specifically,

$$\Omega_0 \approx \{\mathcal{T}_h\}_{h>0} = \bigcup_{e_l=1}^{n} T_{e_l}, \tag{10}$$

where in this work we consider each T_{e_l} to be a tetrahedron. This means that in practice a finite-dimensional projection of the full displacement field is solved for when minimising Eq. (8). The continuous displacement field within each element is represented as a weighted sum of a finite set of basis functions

$$\boldsymbol{u}(\boldsymbol{X}) = \sum_{\alpha=1}^{n_e} \boldsymbol{u}_{\alpha i}^h N_\alpha(\boldsymbol{X}), \tag{11}$$

in which $\boldsymbol{u}_{\alpha i}^h$ is the displacement component along the ith direction at the αth node, n_e is the number of nodes in one finite element, and N_α is the finite element basis function at the αth node.

2.3. Graph neural network surrogate model

There has been considerable research into the application of GNNs to surrogate modelling in recent years [45,61–63]. A GNN can be seen as a generalisation of a Convolutional Neural Network (CNN) which allows for data without a regular grid-like neighbourhood structure to be modelled. GNNs are particularly well suited to emulation of systems represented by a computational mesh, because the mesh can be easily converted to a graph representation. We define a graph to be the following three-tuple

$$\mathcal{G} \triangleq (\mathcal{V},\mathcal{E},\boldsymbol{\theta}), \tag{12}$$

where $\mathcal{V}$ is the set of nodes of the graph, $\mathcal{E}$ is the set of directed edges which define the graph topology, and $\boldsymbol{\theta} \in \mathbb{R}^{D_\theta}$ is a vector of global parameters of the graph. The nodes $n_1, n_2, \ldots, n_{|\mathcal{V}|} \in \mathcal{V}$ are simply taken to be the nodes from the FE mesh, while the directed edges $\{n_i \rightarrow n_j\} \in \mathcal{E}$ are found by converting each undirected edge in the FE mesh into two directional edges, which point in opposite

Algorithm 1 `PrimalGraphEmulator`

Input: $\tilde{\mathcal{G}} = (\mathcal{V} \cup \tilde{\mathcal{V}}, \mathcal{E} \cup \tilde{\mathcal{E}}, \boldsymbol{\theta})$; trainable parameters = $\boldsymbol{\omega}$
Output: $\hat{\boldsymbol{U}} = \{\hat{\boldsymbol{u}}_i \text{ for all } n_i \in \mathcal{V}\}$

Encoder:
1: $\boldsymbol{v}_i^0 = f_V\left(\boldsymbol{v}_i\right)$ for all $n_i \in \mathcal{V} \cup \tilde{\mathcal{V}}$
2: $\boldsymbol{e}_{i\to j}^0 = f_E\left(\boldsymbol{e}_{i\to j}\right)$ for all $\{n_i \to n_j\} \in \mathcal{E} \cup \tilde{\mathcal{E}}$ where $i > j$

Processor:
3: **for** $k = 1 : K$
4: $\quad \boldsymbol{m}_{i\to j}^k = g_E^k\left(\boldsymbol{e}_{i\to j}^{k-1}, \boldsymbol{v}_i^{k-1}, \boldsymbol{v}_j^{k-1}\right)$ for all $\{n_i \to n_j\} \in \mathcal{E} \cup \tilde{\mathcal{E}}$ where $i > j$
5: $\quad \boldsymbol{m}_{i\to j}^k = -\boldsymbol{m}_{j\to i}^k$ for all $\{n_i \to n_j\} \in \mathcal{E} \cup \tilde{\mathcal{E}}$ where $i < j$
6: $\quad \boldsymbol{v}_i^k = \boldsymbol{v}_i^{k-1} + g_V^k\left(\boldsymbol{v}_i^{k-1}, \sum_{j\in\mathcal{N}_i^{\tilde{\mathcal{G}}}} \boldsymbol{m}_{j\to i}^k\right)$ for all $n_i \in \mathcal{V} \cup \tilde{\mathcal{V}}$
7: $\quad \boldsymbol{e}_{i\to j}^k = \boldsymbol{e}_{i\to j}^{k-1} + \boldsymbol{m}_{i\to j}^k$ for all $\{n_i \to n_j\} \in \mathcal{E} \cup \tilde{\mathcal{E}}$ where $i > j$
8: **end for**
9: $\boldsymbol{z}_i^{\text{local}} = (\boldsymbol{v}_i^K, \sum_{j\in\mathcal{N}_i^{\tilde{\mathcal{G}}}} \boldsymbol{e}_{j\to i}^K)$ for all $n_i \in \mathcal{V}$

Decoder:
10: $\boldsymbol{z}^{\boldsymbol{\theta}} = f_P(\boldsymbol{\theta})$
11: $\hat{u}_i^d = h^d(\boldsymbol{z}^{\boldsymbol{\theta}}, \boldsymbol{z}_i^{\text{local}})$ for all $n_i \in \mathcal{V}$ for $d = 1, ..., D$
12: $\hat{\boldsymbol{u}}_i = (\hat{u}_i^1, ..., \hat{u}_i^D)$ for all $n_i \in \mathcal{V}$

directions. Finally, the global parameters $\boldsymbol{\theta}$ are taken in this work to be the material properties of the underlying soft-tissue body.[2] Before the graph can be passed to the GNN, each node n_i is assigned a feature vector $\boldsymbol{v}_i \in \mathbb{R}^{D_n}$, while edges are assigned a feature vector $\boldsymbol{e}_{i\to j} \in \mathbb{R}^{D_e}$. The form of the node and edge features will depend on the application context — see Section 2.4.5 for further details.

When working with the nodes and edges directed extracted from the FE mesh, it may take a prohibitive amount of processor stages for information to be propagated around the graph representation. For this reason, we augment the original graphs using *virtual* nodes and edges, which offer a more coarse representation of the geometry, improving the efficiency of the processor stage of the GNN with respect to the number of message passing steps performed. For a comprehensive overview of this, see Section 2.3 of Dalton et al. (2022) [56]. In brief, the virtual nodes $\tilde{\mathcal{V}}$ and virtual edges $\tilde{\mathcal{E}}$ are generated in a recursive manner, making use of the original FE nodes $\mathcal{V}$. The first layer of virtual nodes is generated by clustering the FE nodes into a set of points of lower cardinality. Inter-layer virtual edges are then created by assigning edge connections between each real node and its cluster centre node. Intra-layer virtual edges between the virtual nodes are found using κ nearest neighbours. This process is then repeated for the desired number of steps, where at each step, the clustering is performed on the virtual nodes from the previous step. The entire augmented graph on which the emulator operates is then

$$\tilde{\mathcal{G}} = (\mathcal{V} \cup \tilde{\mathcal{V}}, \mathcal{E} \cup \tilde{\mathcal{E}}, \boldsymbol{\theta}). \tag{13}$$

Algorithm 1 presents the GNN architecture used in this work, which defines a forward map of the form `PrimalGraphEmulator` : $(\tilde{\mathcal{G}}, \boldsymbol{\omega}) \to \hat{\boldsymbol{U}}$, with $\boldsymbol{\omega}$ used to denote the trainable parameters of the network (see Section 2.3.1 below for further details), and where $\hat{\boldsymbol{U}}$ is the predicted displacement field over the FE nodes. The algorithm applies an encode–process–decode approach to perform the forward map, which is illustrated schematically in Fig. 2. This GNN architecture is almost identical to that introduced in previous work [56], with some minor adjustments as here we concentrate on emulation of systems with a fixed geometry rather than considering varying geometry data. Extending the PI-GNN framework to handle varying geometries will be the remit of future work.

The first stage of the algorithm is the **Encoder**. One line 1, the node feature vectors $\boldsymbol{v}_i$ are encoded into a high dimensional embedding $\boldsymbol{v}_i^0$ using a fully-connected neural network (FCNN), denoted $f_V : \mathbb{R}^{D_n} \to \mathbb{R}^M$, where $M \gg D_n$. One line 2, edge features $\boldsymbol{e}_{i\to j}$ are similarly encoded into embedding vectors denoted $\boldsymbol{e}_{i\to j}^0$, using the edge-encode FCNN $f_E : \mathbb{R}^{D_e} \to \mathbb{R}^M$. This feature encoding increases the expressive power of the final learned representations.

The second stage of the algorithm is the **Processor**, during which the node and edge embeddings from stage one are iteratively updated over $K \in \mathbb{N}$ message passing steps. At each step $k = 1, \ldots, K$, a message vector $\boldsymbol{m}_{i\to j}^k$ is first computed over all edges in the graph. For those edges $\{n_i \to n_j\}$ where $i > j$, the corresponding message vectors are explicitly computed on line 4 using the edge-process FCNN $g_E^k : \mathbb{R}^{3M} \to \mathbb{R}^M$. For those edges $\{n_i \to n_j\}$ where $i < j$, the message vectors are simply found as the negative of the message from its twin edge $\{n_j \to n_i\}$ on line 5. Computing the messages in this manner enforces the symmetry $\boldsymbol{m}_{i\to j}^k = -\boldsymbol{m}_{j\to i}^k$ for all twin pairs of directed edges, which is inspired by interpreting the messages as forces and then applying Newton's third law

[2] This is why here we re-use the $\boldsymbol{\theta}$ notation from Section 2.1.

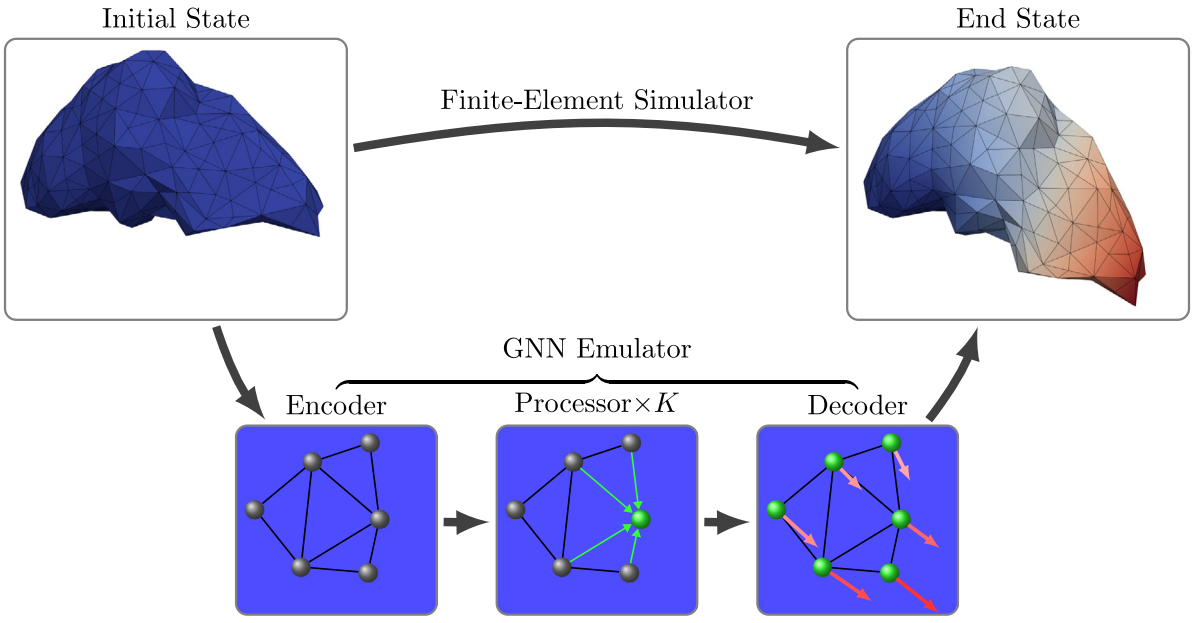


Fig. 2. The GNN emulator maps the initial state of the soft-tissue body (in this case a liver geometry) to its end state using a three-stage, encode–process–decode approach.

of motion [55,56]. The updated node-embeddings $\boldsymbol{v}_i^k$ are then found on line 6 using the node-update FCNN $g_V^k : \mathbb{R}^{2M} \to \mathbb{R}^M$ for all $n_i \in \mathcal{V} \cup \tilde{\mathcal{V}}$. Neighbourhood information is accounted for in the update by summing over the incoming messages to each node. The set of all nodes $n_j \in \mathcal{V} \cup \tilde{\mathcal{V}}$ which send a directed edge to node n_i is called its *graph neighbourhood* $\mathcal{N}_i^{\tilde{\mathcal{G}}}$, defined as

$$\mathcal{N}_i^{\tilde{\mathcal{G}}} \triangleq \{n_j \in \mathcal{V} \cup \tilde{\mathcal{V}} \mid \{n_j \to n_i\} \in \mathcal{E} \cup \tilde{\mathcal{E}}\}. \tag{14}$$

The updated edge-embeddings $\boldsymbol{e}_{i\to j}^k$ are found on line 7 by simply adding the associated message vector $\boldsymbol{m}_{i\to j}^k$ to the embedding from the previous step, $\boldsymbol{e}_{i\to j}^{k-1}$. After K steps of message passing are performed, a local learned representation $\boldsymbol{z}_i^{\text{local}}$ is returned for all real nodes $n_i \in \mathcal{V}$ on line 9.

The third and final stage of the algorithm is the **Decoder**. On line 10, the global parameters $\boldsymbol{\theta}$ are encoded to the embedding vector $\boldsymbol{z}^{\boldsymbol{\theta}}$ using the parameter-encode FCNN $f_P : \mathbb{R}^{D_\theta} \to \mathbb{R}^M$. The global parameter embedding $\boldsymbol{z}^{\boldsymbol{\theta}}$ and local node embedding $\boldsymbol{z}_i^{\text{local}}$ are then used by the node-decode FCNN $h^d : \mathbb{R}^{3M} \to \mathbb{R}$ on line 11 to output predicted displacement values $\hat{u}_i^d$ for all real nodes $n_i \in \mathcal{V}$, over each spatial dimension d.

One feature of this architecture to note is that the material parameter information $\boldsymbol{\theta}$ is only included after the message-passing steps have been completed [56]. When considering repeated forward evaluations for a fixed input geometry under different material parameter configurations, as required for example in an inverse problem, the processor must only be computed *once* initially. Subsequent evaluations then just require the decoder to be performed. Since the processor stage constitutes the bulk of the computational requirements of the GNN, forward evaluations of the surrogate can be made highly efficiently in this case — this is explored further in Section 3.2 (see Table 2).

2.3.1. GNN surrogate training

The GNN surrogate is comprised of $(3+2K+D)$ internal FCNNs: three encoders $\{f_V, f_E, f_P\}$, two processors for each of K rounds of message passing, $\{g_V^k, g_E^k\}_{k=1}^K$, and D decoders $\{h^d\}_{d=1}^D$, where D is the dimensionality of the system. One node-decode FCNN with D outputs could have been used instead here, however we found during experimentation in previous work that this lead to less accurate results [56]. A layer normalisation operation (LayerNorm) is applied after each FCNN (with the exception of the decoder FCNNs) and in the creation of $\boldsymbol{z}_i^{\text{local}}$ to normalise the intermediate values, which increases numerical stability during training [64]. The trainable parameters of the GNN, which we denote collectively as $\boldsymbol{\omega}$, consistent then of all the weights and biases of the internal FCNNs along with the LayerNorm parameters.

Neural network surrogate models are typically trained in a data-driven approach, using a loss function derived from a dataset of numerical forward simulations. Denote a simulation dataset as $\{(\mathcal{G}_j, \boldsymbol{U}_j)_{j=1}^N\}$, the data-driven approach to learning a point estimate of $\boldsymbol{\omega}$ is then

$$\textbf{Data-Driven :} \quad \boldsymbol{\omega}^* = \underset{\boldsymbol{\omega}}{\operatorname{argmin}} \sum_{j=1}^{N} L\left(\boldsymbol{U}_j(\boldsymbol{\theta}_j), \hat{\boldsymbol{U}}_j(\mathcal{G}_j; \boldsymbol{\omega})\right) \tag{15}$$

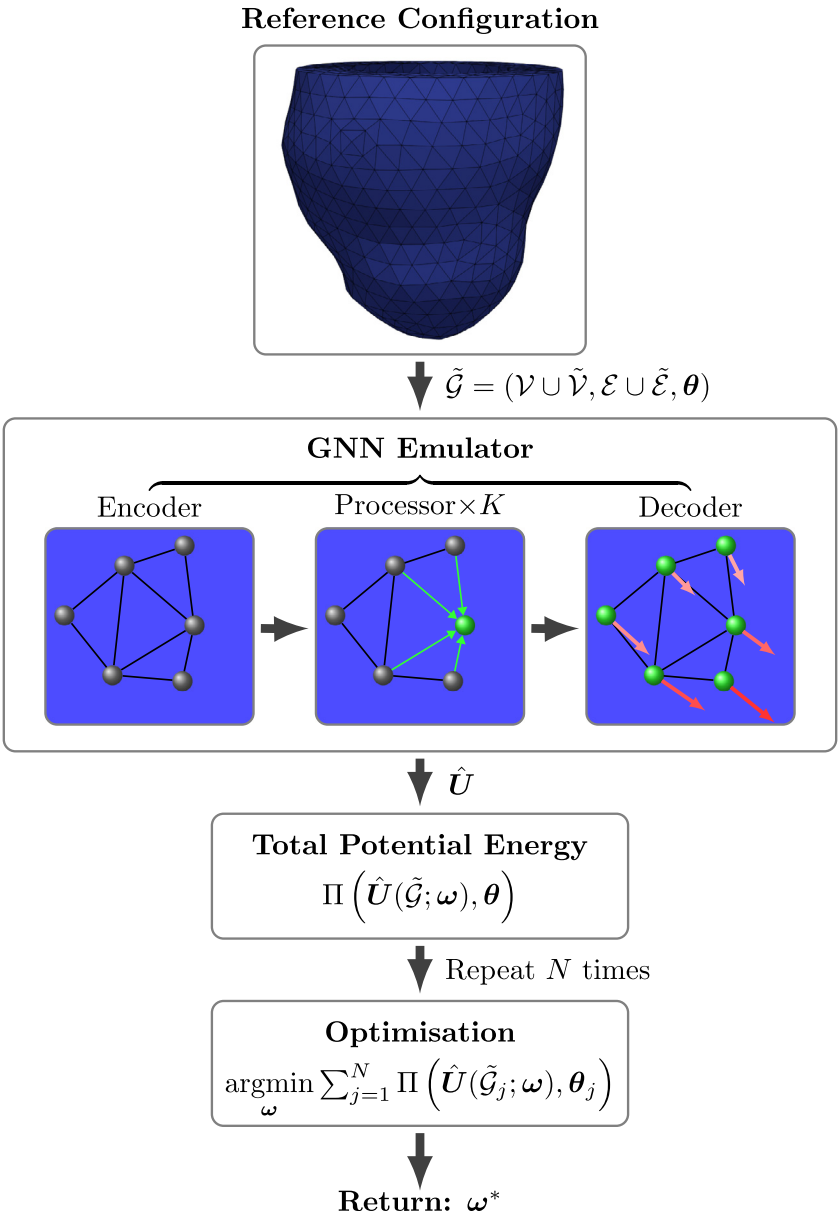


Fig. 3. Schematic illustration of physics-informed training of $\boldsymbol{\omega}$, the tunable parameters of a GNN emulator.

where $L : \mathbb{R}^{|\mathcal{V}|\times 3} \times \mathbb{R}^{|\mathcal{V}|\times 3} \to \mathbb{R}$ is a user-specified loss function, for example root mean-squared-error. Eq. (15) is generally minimised using an iterative, gradient based updating scheme. Instead of a batch approach, where the gradients are computed from all N data points, we have used the concept of mini-batches. This is equivalent to a stochastic gradient descent scheme, which tends to be faster, less susceptible to entrapment in local optima, and more robust with respect to overfitting [65].

Physics-Informed training is an alternative approach, whereby instead of considering simulation data, training is performed against known properties of the physical system being modelled, which can be formulated using balance equations, PDE residuals, or conservation principles [41]. In this work, we make use of a loss function based on the total potential energy functional Π (Eq. (8)), which is illustrated schematically in Fig. 3. Specifically, a point-estimate of the network parameters is learned as:

$$\textbf{Physics-Informed :} \quad \boldsymbol{\omega}^* = \underset{\boldsymbol{\omega}}{\operatorname{argmin}} \sum_{j=1}^{N} \Pi\left(\hat{\boldsymbol{U}}_j(\mathcal{G}_j; \boldsymbol{\omega}), \boldsymbol{\theta}_j\right) \tag{16}$$

A clear difference between the two approaches is that physics-informed training is not dependent on simulator outputs $\boldsymbol{U}_j$, in contrast to data-driven training — further differences are explored in Section 3.1. Note that a surrogate model can also be trained in a *multi-task* manner, using a loss function which incorporates both a data fit term and a physics-informed term [34].

2.4. Implementation details

2.4.1. FEM simulation details

All FEM simulations were performed using first-order tetrahedral elements. For any simulations involving a traction force, an iterative solution approach was taken whereby the pressure was linearly increased from zero to the final prescribed value.

2.4.2. Computation of total potential energy

To calculate the total potential energy in Eq. (8), the deformation gradient $\boldsymbol{F}$ must first be found. Because first-order tetrahedral elements were used, $\boldsymbol{F}$ only needs to be computed at the centroid of each element. Denoting the coordinates of the four vertices of a tetrahedron in the current (reference) configuration as $\boldsymbol{x}_i$, $\boldsymbol{x}_j$, $\boldsymbol{x}_k$, and $\boldsymbol{x}_l$ ($\boldsymbol{X}_i$, $\boldsymbol{X}_j$, $\boldsymbol{X}_k$ and $\boldsymbol{X}_l$), then $\boldsymbol{F}$ is found as

$$\boldsymbol{F} = \left[\boldsymbol{x}_j - \boldsymbol{x}_i,\ \boldsymbol{x}_k - \boldsymbol{x}_i,\ \boldsymbol{x}_l - \boldsymbol{x}_i\right]\left[\boldsymbol{X}_j - \boldsymbol{X}_i,\ \boldsymbol{X}_k - \boldsymbol{X}_i,\ \boldsymbol{X}_l - \boldsymbol{X}_i\right]^{-1}, \tag{17}$$

where the position vector for each node is treated as a column vector. Eq. (17) is the same as the method used in [43]. The integrals required for the calculation of Π are computed in the reference configuration Ω_0. As a result, we will need to express both the body force and the traction force in the reference configuration, $\boldsymbol{b}_0$ and $\boldsymbol{t}_0$, respectively. Here, the body force is taken to be gravity which is independent of the deformation, thus it remains the same. If the traction force is not a dead load but resulted from a pressure load P, then according to Nanson's formula, the traction force in the reference configuration can be expressed as

$$\boldsymbol{t}_0 = -J\, P\boldsymbol{F}^{-T}\boldsymbol{N}, \tag{18}$$

in which $\boldsymbol{N}$ is the normal of the boundary faces of the elements where a pressure load is applied. Finally the total potential energy (Eq. (8)) can be approximated as

$$\Pi \approx \sum_{el=1}^{n}\int_{\Omega_e^{el}} \Psi(\boldsymbol{F})dV - \sum_{el=1}^{n}\int_{\Omega_e^{el}} \boldsymbol{b}_0\cdot\boldsymbol{u}dV - \sum_{fl=1}^{n^f}\int_{\partial\Omega_e^{fl}} \boldsymbol{t}_0\cdot\boldsymbol{u}dA, \tag{19}$$

where Ω_e^{el} is the domain occupied by the elth tetrahedral element in the reference configuration, $\partial\Omega_e^{fl}$ is the flth face on the Neumann boundary with n^f the number of faces on the Neumann boundary. The differential volume dV and the differential area dA are computed in the reference configuration. The total potential energy is approximated using the first-order Gauss quadrature rule, that is

$$\Pi \approx \sum_{el=1}^{n}\Psi(\boldsymbol{F}^{el})V_e^{el} - \sum_{el=1}^{n}\left(\boldsymbol{b}_0^{el}\cdot\boldsymbol{u}_e^{el}\right)V_e^{el} - \sum_{fl=1}^{n^f}\left(\boldsymbol{t}_0^{fl}\cdot\boldsymbol{u}_A^{fl}\right)A_e^{fl}, \tag{20}$$

in which $\boldsymbol{u}_e^{el}$ is the displacement vector at the centroid of the elth element, and $\boldsymbol{u}_A^{fl}$ is the displacement vector at the centroid of the flth face, both of which are linearly interpolated from the nodal displacements $\boldsymbol{u}_i$. V_e and A_e are the volume and area of the associated element and face in the reference configuration, respectively, which can be precomputed in advance of emulator training.

2.4.3. Stabilising potential energy computations

Training an emulator directly on the discretised potential energy functional Π from Eq. (20) is numerically difficult. During the early stages of training, the surrogate may produce physically unrealistic predictions, leading to extreme behaviour (such as divergences) in the computation of Π. In [44], the authors overcome this issue by initially training on a small subset of parameter space, before gradually expanding the training domain. In the present work, we make use of the entire material parameter domain throughout training, where numerical stability in Π is ensured by the use of certain barrier transformation functions, which we now detail.

Firstly, due to the invertibility of the forward map χ and because volumes cannot be negative, we know $a-priori$ that any volumetric change ratios from the reference to current configuration must be strictly positive. Mathematically, this means that $J > 0$. An untrained emulator could however make a prediction to the displacement field which violates this condition. For this reason, we apply the following transformation to $\hat{J}$ before evaluating Π,

$$\hat{J}^{trans} = \begin{cases} \exp(\beta_1\hat{J} + \beta_2) + \beta_3, & \text{if } \hat{J} \le J^{ch} \\ \hat{J}, & \text{otherwise.} \end{cases} \tag{21}$$

Here J^{ch} is the point after which the transformation changes to the identity map, and $\{\beta_1, \beta_2, \beta_3\}$ are parameters of the transformation whose values must be specified. Elementary algebraic manipulations and the asymptotic behaviour of the exponential function can be combined to show that setting

$$\beta_1 = \frac{1}{J^{ch} - J^{min}}, \quad \beta_2 = \log(J^{ch} - J^{min}) - \frac{J^{ch}}{J^{ch} - J^{min}} \quad \text{and } \beta_3 = J^{min} \tag{22}$$

ensures $\hat{J}^{trans} > J^{min}$, and that the transformation is both continuous and differentiable at the change point J^{ch}. For all experiments, we set $J^{min} = 0.001$ and $J^{ch} = 0.05$, whereas for quasi-incompressible materials $J \approx 1$.

For the H-O and Guccione constitutive laws, a similar issue arises due to their use of the exponential function. Physically unrealistic emulator predictions during the initial phase of training can lead to large predicted values for I_1, I_{4f} or Q, which when exponentiated lead to overflow in the computation of Π. We prevent this behaviour through use of the following transformation

$$\hat{Z}_{trans} = \begin{cases} \tanh(\hat{Z} - Z^{ch}) + Z^{ch}, & \text{if } \hat{Z} \geq Z^{ch} \\ \hat{Z}, & \text{otherwise.} \end{cases} \tag{23}$$

where $\hat{Z} \in \{\hat{I}_1, \hat{I}_{4f}, \hat{Q}\}$ is the relevant value computed from the emulator's prediction and $Z^{ch} \in \{I_1^{ch}, I_{4f}^{ch}, Q^{ch}\}$ is the change point after which the non-linear transformation is applied. This transformation is continuous and differentiable at Z^{ch}, and ensures $\hat{Z}_{trans} < Z^{ch} + 1$. We do not have rigorous finite upper bounds for I_1, I_{4f} or Q. Instead, we set the bounds to be higher than is expected for the simulation results, but low enough to prevent overflow from becoming an issue. These specific values were $I_1^{ch} = 10$, $I_{4f}^{ch} = 8$ or $Q^{ch} = 15$.

Finally, we initialise the weights and biases in the final layer of the decoder FCNNs to zero. This ensures that the untrained emulator predicts zero displacement for all material parameter values, give further numerical stability in the initial stages of training.

2.4.4. Enforcing Dirichlet boundary conditions

For each model considered, we explicitly enforce the outputs of the GNN emulator along the Dirichlet boundary Ω^d to satisfy the prescribed displacement values. This is achieved through an additional post-processing step after evaluation of the emulator, where the prescribed values are substituted in place of the predicted values on Ω^d. In principle, these boundary conditions could be softly enforced using a penalty method, however we found training was significantly easier when using explicit enforcement — similar results were reported in [63].

2.4.5. GNN emulation details

Performing emulation with the `PrimalGraphEmulator` architecture (see Algorithm 1) requires the number of message-passing steps K, the dimensionality M of the internal embedding vectors, and the structure of the internal FCNNs to be specified. For all experiments, we fixed $K = 5$, $M = 40$ and used FCNNs with CELU activation function [66] and two hidden layers, each of width 128. The choice of these hyper-parameters is based on extensive numerical experiments conducted in previous work [56]. With these values, the emulator had approximately 510,000 trainable parameters for each model considered. Training is performed using Adam [67], with a batch size of one on a single GPU (Dual NVIDIA RTX A6000). The effect of changes in learning rate on emulator training is explored in Appendix A - we find that a learning rate of 1×10^{-4} is optimal in the early stages of training. In data-driven training of neural networks, the use of techniques such as early stopping can be critical to prevent overfitting and ensure good generalisation performance. However we found during experimentation that this was not required for physics-informed training, where overfitting was not observed during longer training runs.

The node and edge features for each model considered were specified to have the following form

$$\boldsymbol{v}_i = (d_i) \text{ or } (d_i, \boldsymbol{f}_i) \tag{24}$$

$$\boldsymbol{e}_{i \to j} = (\boldsymbol{x}_j - \boldsymbol{x}_i, \|\boldsymbol{x}_j - \boldsymbol{x}_i\|_2), \tag{25}$$

where d_i is a Boolean variable indicating if the node lies on the Dirichlet boundary, and $\boldsymbol{f}_i$ is the unit vector in the fibre direction. Fibre information is only included for models with non-isotropic constitutive law and spatially varying fibre field in the reference configuration. Node and edge features are normalised element-wise to mean zero and unit variance before being inputted to the emulator. Note how absolute positional information is not included in the node features, instead only relative positions are incorporated via the edge features. This ensures that the emulator is invariant under translations of the underlying geometry in space.

2.4.6. Data and code availability

All experiments were implemented in Python. FEM simulations were performed using FEniCS [68], and PI-GNN computations were performed using the JAX [69], Flax [70] and Optax [71] libraries. All data and code is available on GitHub[3].

All data and code will be made available if accepted for publication.

3. Numerical experiments

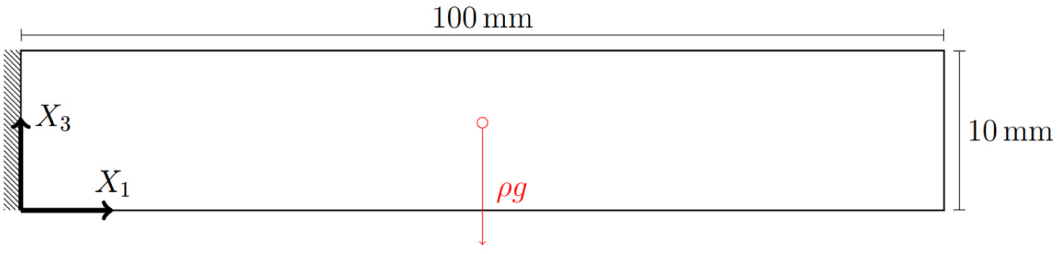
We conducted numerical experiments to evaluate the performance of the PI-GNN emulation framework involving a total of five different mechanics models.[4] In each case, a fixed geometry is considered, and an emulator is trained over a specified material parameter domain using the GNN architecture detailed in Section 2.4.5. Complete details of each model are given in Sections 3.1–3.4, while Table 1 presents a summary. The column for $\boldsymbol{\eta}$ refers to the cardinalities of the layers of virtual nodes created from the base FE nodes — see Algorithm 1 of Dalton et al. (2022) [56] for details. Benchmarking of emulation results is performed primarily in terms of error in prediction of the displacement $\boldsymbol{u}$, the deformation gradient $\boldsymbol{F}$, the first invariant I_1 and the total potential energy Π. Errors on these quantities are quantified using the metrics from Eq. (26), where the hat symbol denotes results from the emulator

[3] https://github.com/dodaltuin/soft-tissue-pignn

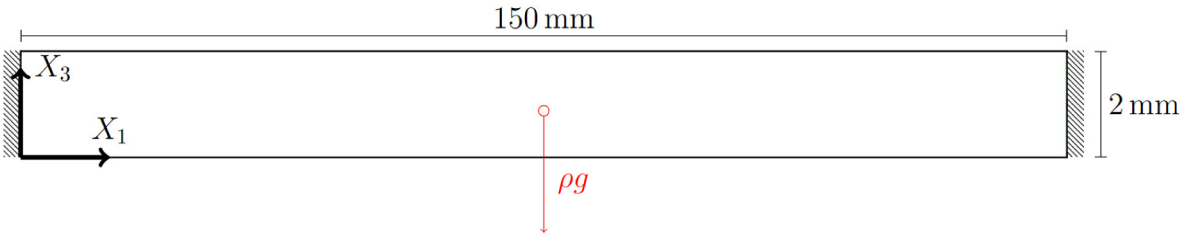
[4] Four additional models are considered in the supplementary material.

Table 1
Summary of models considered for emulation.

Model	Material	$\boldsymbol{\theta}$	$\boldsymbol{\eta}$	N_{node}	N_{elem}
`OnceClampedBeam`	Neo-Hookean	(λ, μ)	[97, 24]	390	1440
`TwiceClampedBeam`	Neo-Hookean	(λ, μ)	[97, 24]	390	1440
`TwistingCube`	Neo-Hookean	(E, ν)	Varying	Varying	Varying
`Liver`	Neo-Hookean	(λ, μ)	[233, 58, 14]	935	4408
`LeftVentricle`	Holzapfel–Ogden	(a, b, a_f, b_f)	[392, 98, 24]	1570	6176



(a) `OnceClampedBeam`



(b) `TwiceClampedBeam`

Fig. 4. Illustration of the rectangular beam models considered as 2-D slices in the (X_1, X_3) plane (not to scale), where the dashed lines indicate a clamped Dirichlet boundary, ρ is the density of the material and g the acceleration due to gravity.

and $\|\cdot\|_F$ is the Frobenius norm. The error metric on $\boldsymbol{u}$ is an absolute value measured in (mm) while $err_{\boldsymbol{F}}$, err_{I_1} and err_{Π} are relative errors, measured in (%).

$$\textbf{Error Metrics}: \quad err_{\boldsymbol{u}} = \|\boldsymbol{u} - \hat{\boldsymbol{u}}\|_2 \text{ (mm)}, \quad err_{\boldsymbol{F}} = \frac{\|\boldsymbol{F} - \hat{\boldsymbol{F}}\|_F}{\|\boldsymbol{F}\|_F} \times 100 \text{ (\%)},$$
$$err_{I_1} = \left|\frac{I_1 - \hat{I}_1}{I_1}\right| \times 100 \text{ (\%)}, \quad err_{\Pi} = \left|\frac{\Pi - \hat{\Pi}}{\Pi}\right| \times 100 \text{ (\%)}. \tag{26}$$

3.1. Data-driven and physics-informed training comparison

The first model considered, `OnceClampedBeam` is illustrated in Fig. 4(a). This model involves a beam with $\Omega_0 = [0, 100] \times [0, 10]^2$ (mm), discretised using 390 nodes and 1440 elements. The beam is clamped at the left end $\partial\Omega_0^d = \{\boldsymbol{X} \in \Omega_0 : X_1 = 0\}$, and displaced under gravitation from its own weight. Specifically, $\boldsymbol{b} = (0, 0, -\rho g)^\top$, with ρ the density of the beam and g the acceleration due to gravity. No traction force is applied. A Neo-Hookean material model is assumed (see Eq. (2)), with material parameters $\lambda, \mu \in [5, 10]$ kPa. The second model considered, `TwiceClampedBeam`, is illustrated in Fig. 4(b). This is similar to the first model, except we have $\Omega_0 = [0, 150] \times [0, 10] \times [0, 2]$ (mm), and both ends of the beam are clamped, that is $\partial\Omega_0^d = \{\boldsymbol{X} \in \Omega_0 : X_1 = 0 \text{ or } X_1 = 150\}$.

We use these models to compare the performance of a GNN emulator trained in a data-driven manner as in Eq. (15) (DD-GNN) with physics-informed training as in Eq. (16) (PI-GNN). For training of the DD-GNN, 200 simulations were first run from randomly sampled material parameter values. Training was then performed for 5000 epochs using a fixed learning rate of 1×10^{-4}. For consistency, the PI-GNN was trained at the exact same material parameter inputs, with the same number of training epochs and learning rate. For evaluation, an independent set of 100 simulations were performed for each model. The average magnitude of the nodal displacements $\|\boldsymbol{u}\|_2$ for `OnceClampedBeam` was 15.7 mm, with maximum value of 50.1 mm. For `TwiceClampedBeam`, the corresponding values were 2.3 mm and 6.0 mm respectively, where the disparity in displacement magnitudes is due to the additional clamped boundary constraints at both ends of the `TwiceClampedBeam` model.

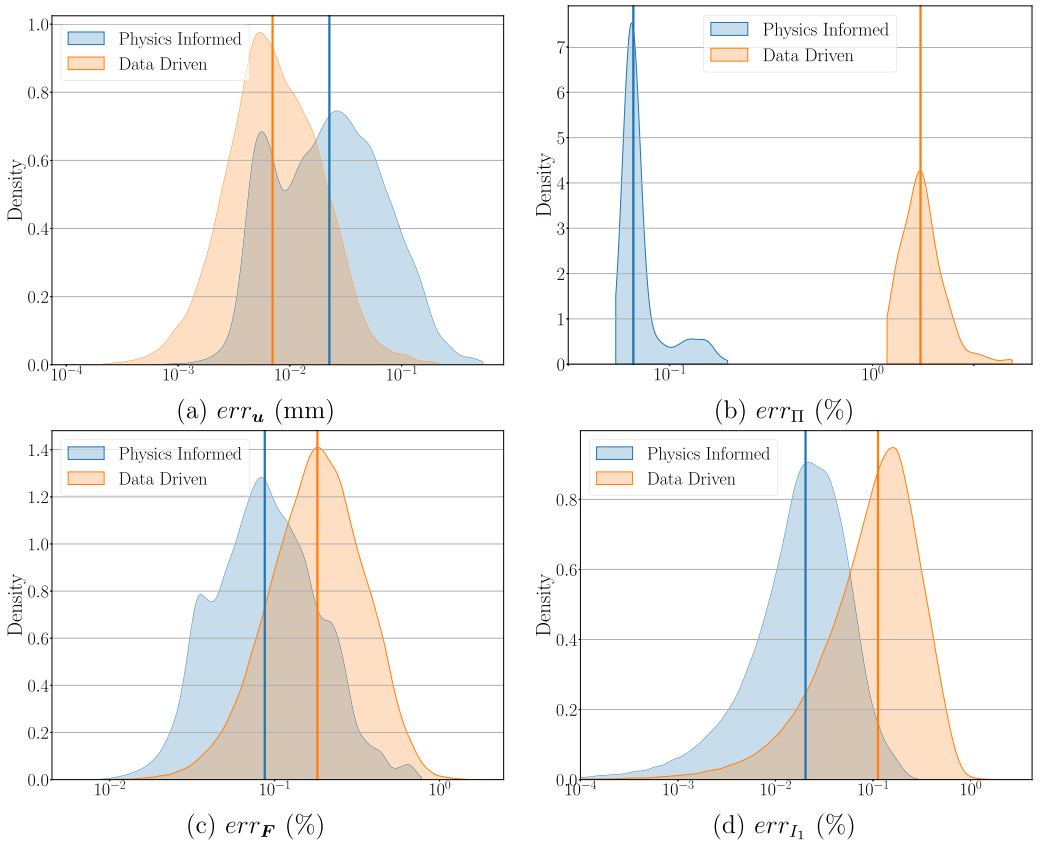


Fig. 5. Comparison of data-driven and physics-informed emulation results on `OnceClampedBeam` model.

Fig. 5 shows emulation results for the `OnceClampedBeam` model, using density plots which compare the test-set accuracy of the two training approaches in terms of the four error metrics from Eq. (26). Note that we present density plots on the log scale, as we believe this makes them more easy to interpret. Panel (a) indicates that the DD-GNN, which as been trained on displacement data, achieves lower error in prediction of the displacements, with median $err_{\boldsymbol{u}}$ value of 6.5×10^{-3} mm compared to 2.0×10^{-2} mm for the PI-GNN. Nevertheless there is a high degree of overlap between the errors. By contrast, there is a sharp divergence between the distributions of err_{Π} for the two emulators in panel (b), as the median err_{Π} value incurred by the DD-GNN of 1.7% percent is two orders of magnitude higher that the corresponding median value of 6.6×10^{-2}% seen with the PI-GNN. Furthermore, there is no overlap between the distributions, i.e. the lowest err_{Π} value for the DD-GNN is larger than the highest err_{Π} value with the PI-GNN. This shows that the predictions of the PI-GNN are consistently more physically realistic in terms of the potential energy state. The more realistic deformation captured by the PI-GNN is reflected in the prediction errors for $\boldsymbol{F}$ and I_1 displayed in panels (c) and (d), where we see lower errors obtained for the PI-GNN. Specifically, the DD-GNN incurs median error of 1.8×10^{-1}% for $err_{\boldsymbol{F}}$ against 8.7×10^{-2}% with the PI-GNN, while the median err_{I_1} value is 1.1×10^{-1}% for the data-driven approach is approximately one order of magnitude greater than for the PI-GNN, which had median err_{I_1} value of 2.0×10^{-2}%.

Fig. 6 shows emulation results for the `TwiceClampedBeam` model. The pattern of results is similar to that seen in Fig. 5, but here the data-driven method performs less well relative to the physics-informed approach. For errors in displacement space in panel (a), the PI-GNN achieves a slightly lower median error value of 9.6×10^{-3} mm versus 1.1×10^{-2} mm for the DD-GNN, albeit with large overlap between the two error distributions. For err_{Π} again we see no overlap between the errors, with the median value of err_{Π} for the DD-GNN of 5.3% three orders of magnitude higher than the corresponding median value for the PI-GNN of 1.6×10^{-3}%. Finally, for the difference between the accuracy for the predictions of $\boldsymbol{F}$ and I_1 was even more pronounced in this model, with PI-GNN errors typically one or two orders of magnitude lower. Specifically, the DD-GNN incurs median error of 8.0×10^{-1}% for $err_{\boldsymbol{F}}$ against 3.8×10^{-3}% with the PI-GNN, while median err_{I_1} value is 1.0×10^{-1}% for the data-driven approach, compared to 7.9×10^{-4}% for the PI-GNN.

One possible reason why the DD-GNN performs better relative to the PI-GNN on the `TwiceClampedBeam` model is the fact that the displacements $\boldsymbol{u}$ for this model had greater magnitude and variation across the test data compared to the `OnceClampedBeam` model. The DD-GNN has the advantage of operating in normalised space by making use of summary statistics for the displacement field found on the training data, whereas the PI-GNN is trained in the original space, meaning the network weights have to be trained to values with larger magnitude to capture the variation in the data.

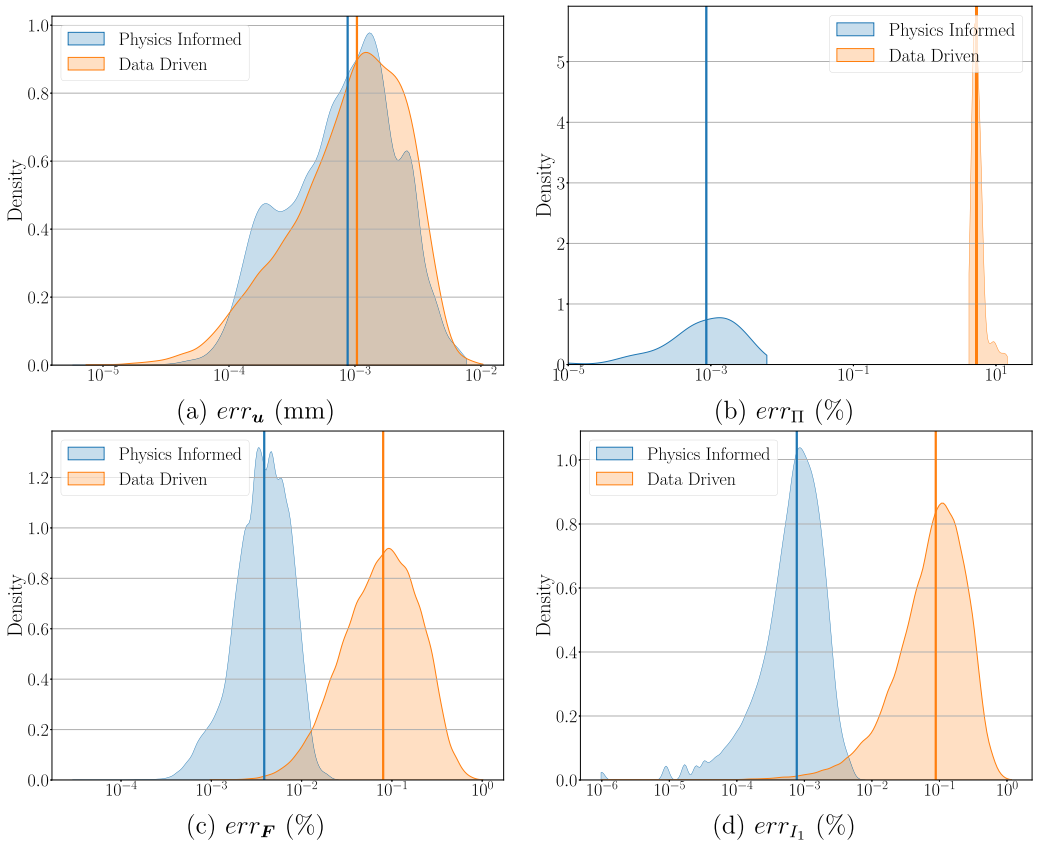


Fig. 6. Comparison of data-driven and physics-informed emulation results on `TwiceClampedBeam` model.

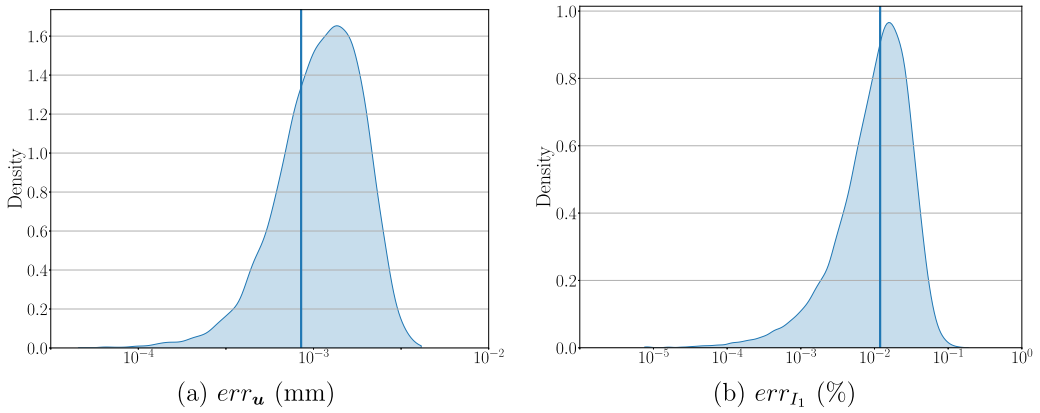
3.2. `TwistingCube`

We next consider emulation of a model `TwistingCube` involving a cuboidal geometry ($\Omega_0 = [0, 10]^3$ mm) discretised with 343 nodes and 1296 elements, based on a similar example from the FEniCS documentation [68]. Here, no external forces are explicitly applied. Instead the cube is clamped at the left most end $\partial\Omega_0^{d_0} = \{\boldsymbol{X} \in \Omega_0 : X_1 = 0\}$, and the following rotational displacements are prescribed at the right end $\partial\Omega_0^{d_1} = \{\boldsymbol{X} \in \Omega_0 : X_1 = 1\}$:

$$\begin{aligned}\boldsymbol{u} =&(0,\\ &(0.5 + (x_2 - 0.5)\cos(\pi/3) - (x_3 - 0.5)\sin(\pi/3) - x_2)/2,\\ &(0.5 + (x_2 - 0.5)\sin(\pi/3) + (x_3 - 0.5)\cos(\pi/3) - x_3)/2).\end{aligned} \tag{27}$$

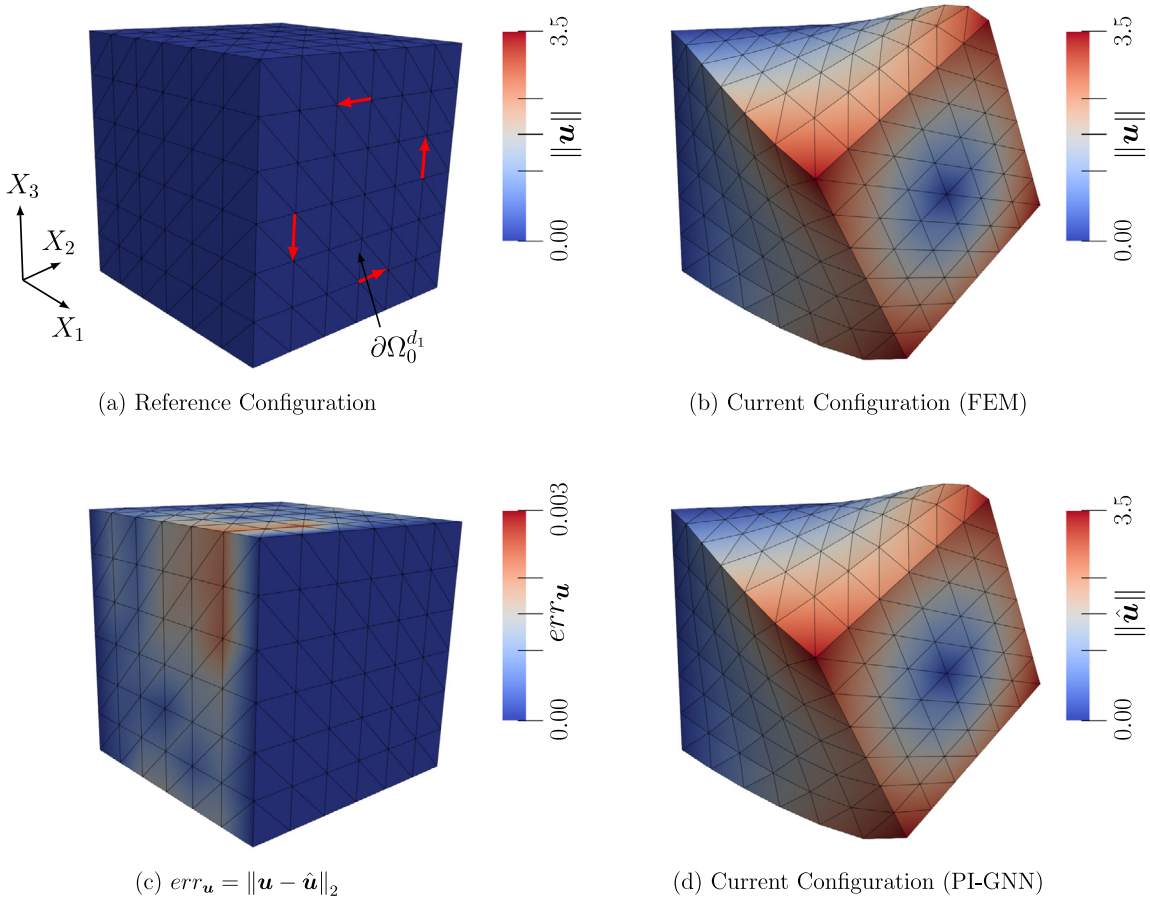
The Neo-Hookean material model is used (see Eq. (2)), parameterised in terms of Young's modulus E and Poisson ratio ν, with ranges $E \in [1, 25]$ and $\nu \in [0.1, 0.4]$. A PI-GNN emulator was trained for 1000 epochs over (E, ν) space. During the first 500 epochs, a fixed set of 200 material parameter configurations were used for training (selected using a Latin Hypercube Sampling (LHS) design), with learning rate of 1×10^{-4}. For the remaining epochs, the learning rate was reduced to 1×10^{-5}, and at each epoch, a new set of 200 material parameters were randomly sampled using a uniform distribution. To evaluate the trained emulator, 50 simulations were performed using the FEM to act as independent test set. Density plots for $err_{\boldsymbol{u}}$ and err_{I_1} on the test set are given in Fig. 7. The density plot of $err_{\boldsymbol{u}}$ shown in Panel (a) shows that the displacement predictions of the emulator are extremely accurate, achieving median $err_{\boldsymbol{u}}$ value of 8.5×10^{-4} mm with no errors exceeding 7.0×10^{-3} mm (for reference, the mean and max values of $\|\boldsymbol{u}\|_2$ over the test data were 1.1 mm and 3.5 mm respectively). From Panel (b), the errors in I_1 are also very low, with median err_{I_1} value of 1.2×10^{-2}%, while no errors are in excess of 2.0×10^{-1}%.

The emulation results for the test-set simulation for which the PI-GNN achieved the median value of Mean($err_{\boldsymbol{u}}$) are visualised in Fig. 8. Panel (a) shows the reference configuration of the cube, panel (b) the current configuration as given by the FEM simulation, while panel (d) shows the current configuration as predicted by the PI-GNN. No differences between the two results are apparent. Panel (c) shows the distribution of $err_{\boldsymbol{u}}$ on the surface of the reference geometry, and indicates that greater errors are incurred in the centre of the cube along the edges, although no prediction errors are observed in excess of 3.0×10^{-3} mm.



(a) $err_{\boldsymbol{u}}$ (mm) (b) err_{I_1} (%)

Fig. 7. Error density plots for `TwistingCube` model (343 FE nodes). The vertical lines indicate median values.



(a) Reference Configuration (b) Current Configuration (FEM)

(c) $err_{\boldsymbol{u}} = \|\boldsymbol{u} - \hat{\boldsymbol{u}}\|_2$ (d) Current Configuration (PI-GNN)

Fig. 8. Median out of sample emulation results for `TwistingCube` model (mm).

To investigate how the density of the FE mesh affects emulation results, we repeated the above experiments using four additional FE meshes, which had 1000 (4374), 4096 (20 250), 13 824 (73 002) and 21 952 (119 098) nodes (elements) respectively. All other model and implementation details remained the same, with the exception of the mesh with 13 824 (21 952) nodes, where emulator

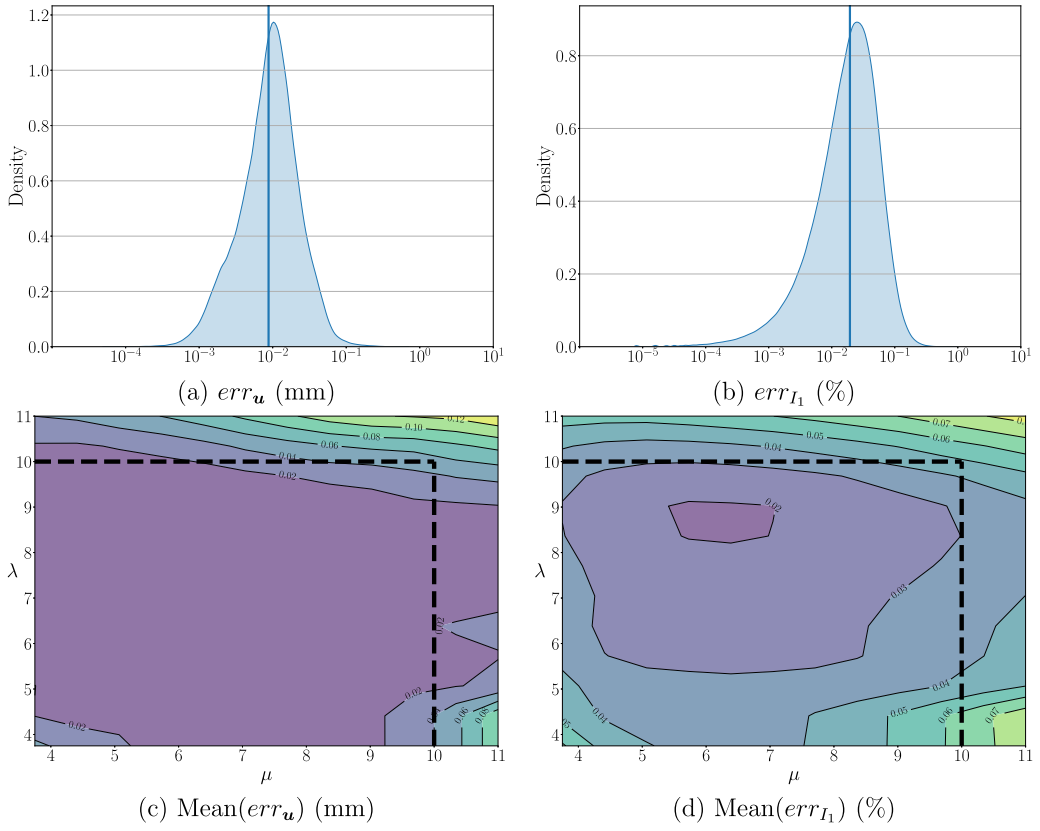
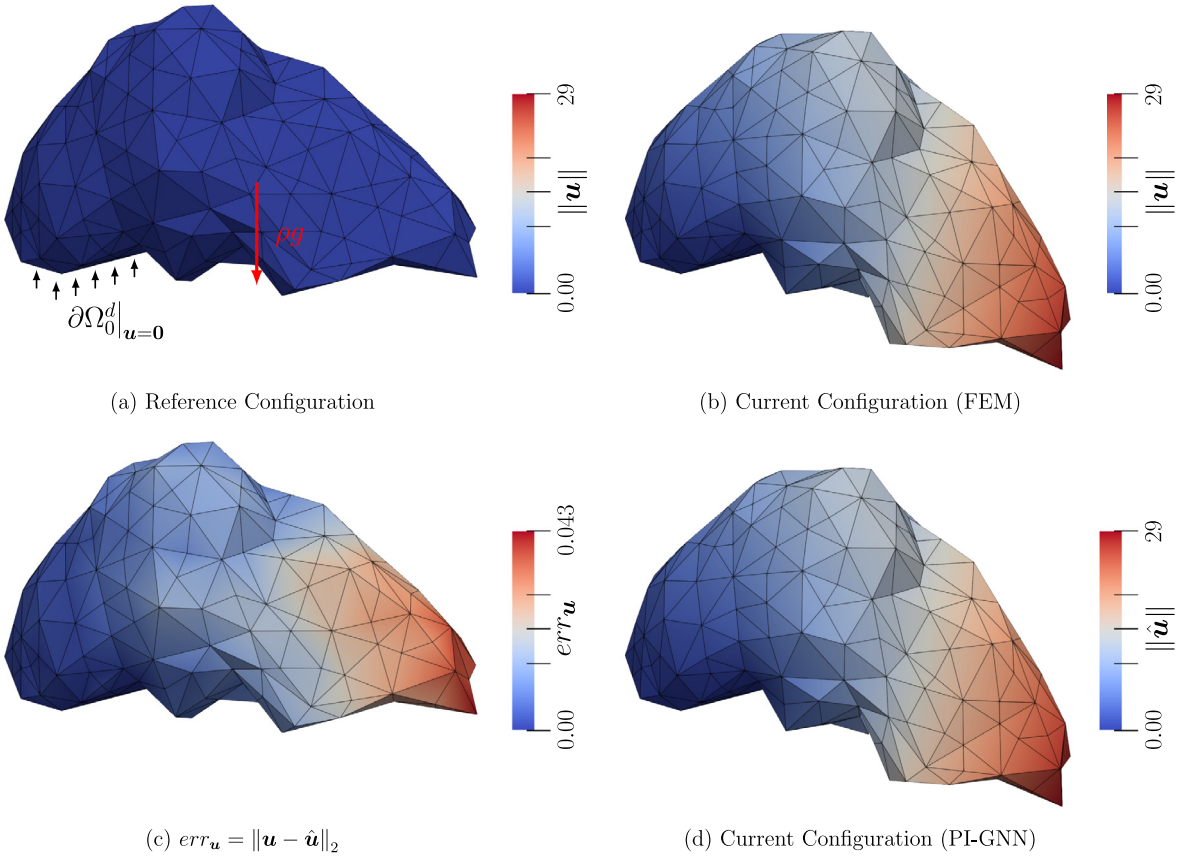


Fig. 9. Emulation results for `Liver` model. The top row shows density plots of $err_{\boldsymbol{u}}$ and err_{I_1}, where the vertical lines indicate median error values. The bottom row displays loss heatmaps of $err_{\boldsymbol{u}}$ and err_{I_1} over the space of Lame parameter (i.e. (λ, μ)) configurations considered. The dashed lines indicate the boundary of the domain considered during training.

Table 2
Summary of emulation results for `TwistingCube` model using different mesh densities. The final row presents prediction times when only the final, decoder stage of Algorithm 1 is evaluated.

N_{node}	343	1000	4096	13 824	21 952
N_{elem}	1296	4374	20 250	73 002	119 098
Mean($err_{\boldsymbol{u}}$)	1.3×10^{-3}	1.1×10^{-3}	1.2×10^{-3}	1.1×10^{-3}	1.7×10^{-3}
Train time	150 min	176 min	190 min	1166 min	2325 min
Prediction time	1.7×10^{-3} s	1.9×10^{-3} s	4.1×10^{-3} s	1.1×10^{-2} s	1.6×10^{-2} s
Decoder time	2.1×10^{-4} s	2.2×10^{-4} s	2.4×10^{-4} s	5.0×10^{-4} s	5.9×10^{-4} s

training was performed for an additional 4000 (6500) steps to ensure convergence of the total potential energy. Table 2 gives the average $err_{\boldsymbol{u}}$ values obtained at each mesh density, where the errors are considered for those points where $X_1 = 0.5$, that is a slice in the (X_2, X_3) plane. Errors remain roughly constant for all mesh densities, indicating that the accuracy of the PI-GNN is not sensitive to mesh density. Note however that fewer training epochs are required to reach the same level of accuracy for less dense meshes. Table 2 also presents training and prediction times for the different meshes. Training and prediction times scale by roughly the same order of magnitude. When the message passing stage of the GNN is precomputed however, prediction times for the decoder increase by less than a factor of three when the number of nodes rises by almost two orders of magnitude from 343 to 21 952. This illustrates the advantage of our GNN architecture design (see Algorithm 1). By decoupling material parameter information from the message-passing stage of the GNN, predictions can be made extremely efficiently for a fixed input geometry once training is complete, even for very dense meshes. This is useful for inverse problems for example, where a large number of forward evaluations may be required to allow the material parameters for a given subject to be inferred from experimental data [72].



(a) Reference Configuration

(b) Current Configuration (FEM)

(c) $err_{\boldsymbol{u}} = \|\boldsymbol{u} - \hat{\boldsymbol{u}}\|_2$

(d) Current Configuration (PI-GNN)

Fig. 10. Median out of sample emulation results for `Liver` model (mm).

3.3. *Liver*

We next consider a model of a human liver, the reference configuration of which can be seen in Fig. 10(a). No traction force is applied, with external loading consisting solely of gravitational force. The Dirichlet boundary consists of half of the bottom surface of the geometry, where zero displacement is allowed. The geometry used is that made available by the authors of Zhang and Chauhan (2020) [73]. In this work, we use the same Dirichlet boundary surface as [73], and also assume the same Neo-Hookean material model (see Eq. (2)). We additionally re-scaled the geometry to have a length of 150 mm, approximately the average value for an adult human [74].

We trained a GNN emulator for the `Liver` model over parameter space $\boldsymbol{\theta} = (\lambda, \mu) \in [3.5, 10]^2$ kPa initially for 500 epochs with a fixed learning rate of 1×10^{-4} using a fixed set of 200 material parameter configurations. Training was then continued for an additional 500 epochs with learning rate 1×10^{-5}, with randomly sampled material parameters at each epoch. The out of sample prediction results of the trained emulator are presented in Fig. 9. These results were evaluated on a set of 482 simulations performed using the FEM, each of which made use of a randomly sampled material parameter vector.

The emulator exhibits strong accuracy in prediction of the displacement values, as is illustrated in the density plot in Fig. 9(a). The median out of sample prediction error is 8.8×10^{-3} mm and only a tiny fraction of errors exceed one-tenth of a mm (for reference, the mean and max values of $\|\boldsymbol{u}\|_2$ over the test data were 7.9 mm and 58.5 mm respectively). From Panel (b), the errors in I_1 are also very low, with median err_{I_1} value of 1.9×10^{-2}%, while virtually no errors exceed half a percent. The two plots on the bottom row display loss heatmaps over material parameter space. We consider the performance of the emulator under extrapolation by extending the heatmaps beyond the domain considered during training, the boundary of which is illustrated by the dashed lines. The plots exhibit broadly similar patterns, with very low errors within the training domain, with a smooth deterioration in prediction accuracy as we move outside this area. Fig. 10 visualises results for the test data simulation where the emulator achieved median value of Mean($err_{\boldsymbol{u}}$). Panels (b) and (d) show the current configurations of the body outputted by the FEM and PI-GNN, respectively. It is difficult to visually discern any differences between the two results. The distribution of $err_{\boldsymbol{u}}$ values is given in Panel (c), which shows that errors are highest towards the end of the geometry where $\|\boldsymbol{u}\|_2$ is highest, however these errors do not exceed 4.3×10^{-2} mm.

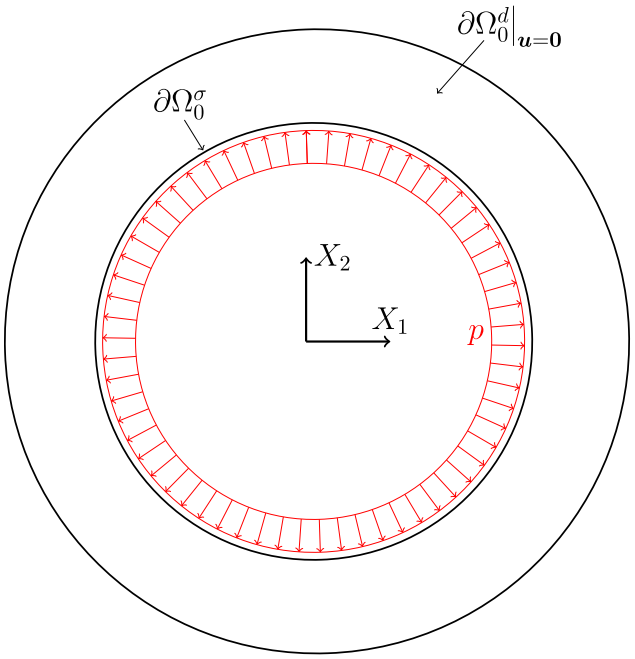


Fig. 11. Short-axis illustration of boundary conditions for `LeftVentricle` model at the base of the geometry (idealised, symmetric geometry only used for illustration purposes). The figure shows in the (X_1, X_2) plane — the X_3 direction runs from the apex to the base (see Fig. 13(a)). Here $\partial\Omega_0^d$ indicates the clamped base of the LV where zero displacements are allowed, $\partial\Omega_0^\sigma$ the inner surface of the LV (the endocardium) where outward pressure is applied, represented by p.

3.4. *LeftVentricle*

In the final emulation experiment, we consider a model for the passive mechanics of the left ventricle (LV) of the heart. The LV geometry considered is a real model, extracted from the cardiac magnetic resonance (CMR) imaging scans of a healthy volunteer at early diastole. Both long and short axis CMR scans were used for the reconstruction, which was performed using segmentation software developed in-house. For further details on LV geometry reconstruction, see [8,75]. A layered myofibre structure is typically incorporated into LV models, but imaging of myofibres in-vivo remains a challenging problem. For this reason, we adopt a rule based method (RBM) to describe the layered myofibres in this work [76]. Here a fibre-sheet-normal local material coordinate system is defined, where we vary the fibre angle linearly from $-90°$ at endocardium to $90°$ at epicardium. The geometry is approximately 75 mm in length from base to apex, and is discretised using 1570 nodes and 6176 elements. The passive filling of the myocardium is modelled by a linearly increased cavity pressure applied to the inner surface of the LV, with no body force applied. The base of the LV is fully constrained with zero displacements. These boundary conditions are illustrated via a 2-D diagram of the base of the LV in Fig. 11. The H-O material model is used to describe the mechanical response of the myocardium. The ranges considered for each material parameter (see Eq. (4)) were $a \in [0.1, 2.6]$ kPa, $b \in [1., 4.2]$, $a_\mathrm{f} \in [1.5, 5.18]$ kPa and $b_\mathrm{f} \in [1, 4.46]$. These ranges were chosen so as to match the mean parameter values found on real data [8], with roughly double the level of variance around the mean. The final pressure loading on the endocardium is allowed to range from 4 to 10 mmHg, and it is incorporated in the GNN by concatenation to the material parameter vector $\boldsymbol{\theta}$. We set the parameter $\mathcal{P}$ to equal 25 kPa (5 kPa was used in [44]) to allow some compressibility in the myocardium. Note that it is still debatable whether the myocardium shall be treated to be fully incompressible or compressible [77]. In future work, we will consider multi-field variational principles as an alternative approach to handle incompressibility — this is discussed further in Section 4.4. The reference configuration of the LV is shown in Fig. 13(a), while Panel (b) shows an FEM simulation result.

A PI-GNN emulator for the `LeftVentricle` model over the four dimensional material parameter space was trained for 15 000 epochs, with a learning rate of 1×10^{-4} during the first half of training and 1×10^{-5} for the second half. More training epochs were used here over previous models based on examination of the traceplots of the potential energy. A test data set of 150 points was generated using the FEM to evaluate the out of sample performance of the trained PI-GNN, where each simulation was performed with a different randomly sampled material parameter vector and pressure loading value. Note that further details of the training procedure used are given in Section 2.4.

Density plots of emulation error on the test set are presented in Fig. 12. From Panel (a), the bulk of displacement prediction errors fall within an order of magnitude of the median $err_{\boldsymbol{u}}$ value of 2.8×10^{-2} mm, however the tail of the distribution approaches 6.5×10^{-1} mm (for reference, the mean and max values of $\|\boldsymbol{u}\|_2$ over the test data were 6.8 mm and 26.3 mm respectively). The

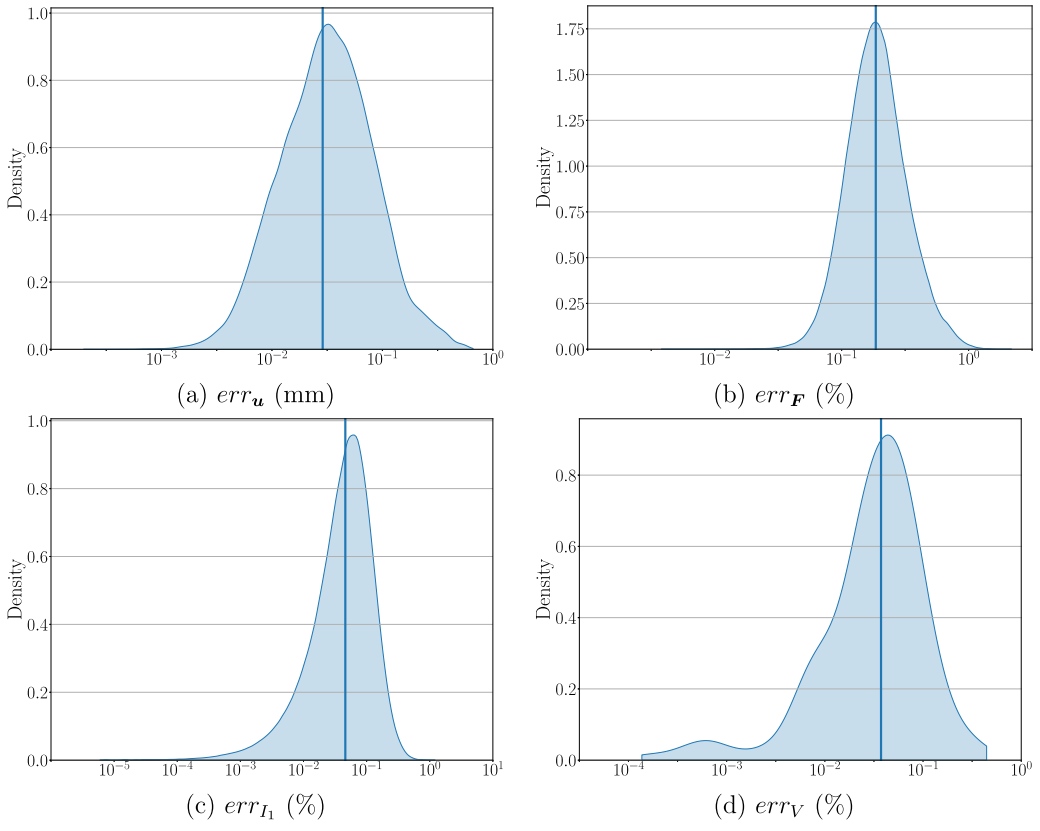


Fig. 12. Error density plots for `LeftVentricle` model. The vertical lines indicate median values.

distributions of $\boldsymbol{F}$ and err_{I_1} errors are both slightly more peaked around their median values of 1.9×10^{-1}% and 4.7×10^{-2}% respectively. We also consider the emulation error in predicting the volume V enclosed within the cavity of the LV, using the error measure

$$err_V = \left| \frac{V - \hat{V}}{V} \right| \times 100, \tag{28}$$

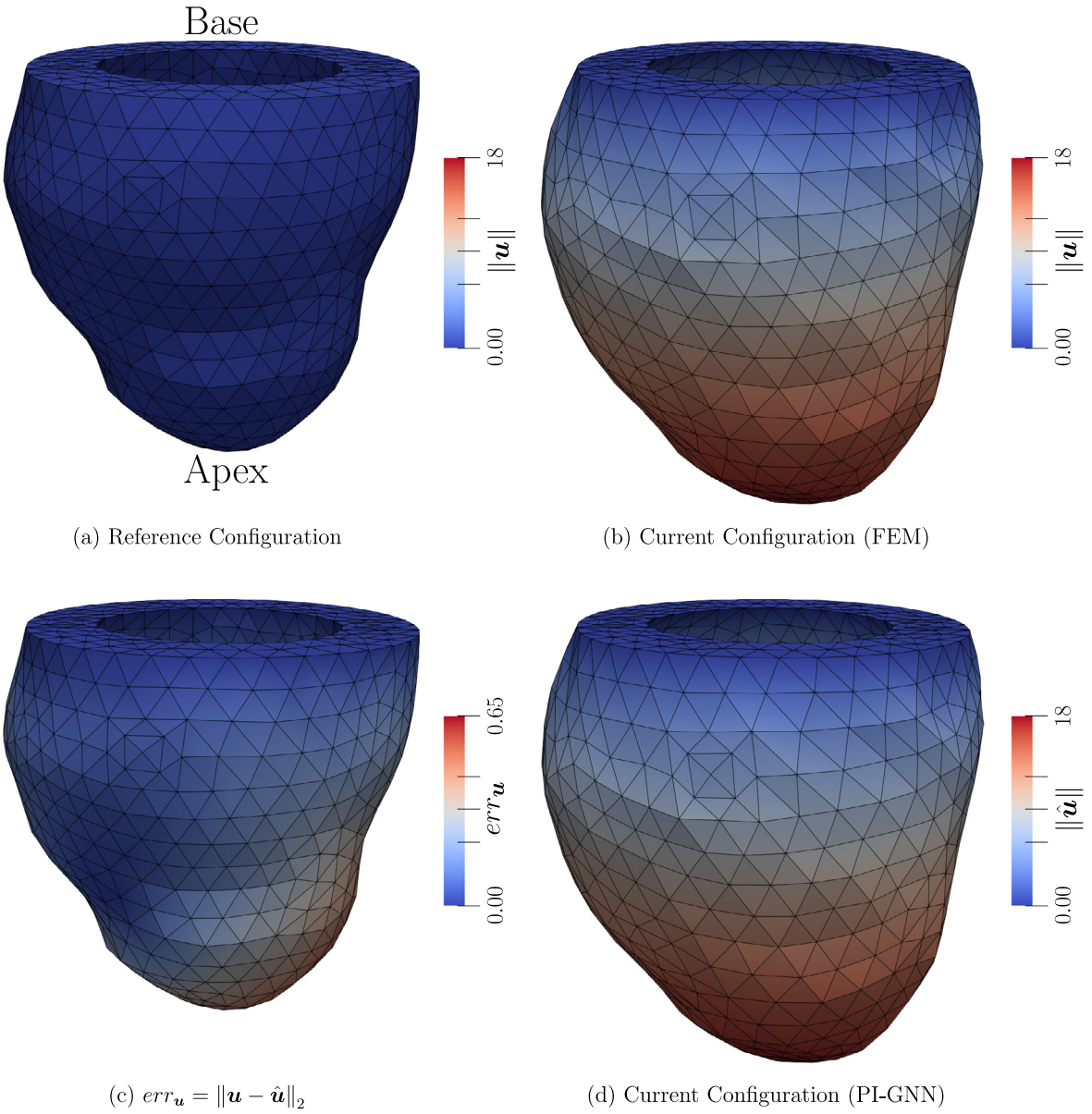
as cavity volume is an important quantity for clinicians in the diagnosis of certain cardiovascular diseases [78]. The distribution of err_V from Panel (d) indicates strong agreement between the simulator and emulator on the test data, with median error value of 3.7×10^{-2}%, while the largest error incurred was less than half a percent.

Emulation results for the `LeftVentricle` model for the test simulation where the emulator achieved the median value of Mean($err_{\boldsymbol{u}}$) are visualised in Fig. 13. Comparing the FEM and PI-GNN results from panels (b) and (d) respectively shows strong agreement between the two. The distribution of $err_{\boldsymbol{u}}$ values from panel (c) indicates that errors are highest further from the base of the LV, and approach 1×10^{-1} mm at the apex.

In addition to strong emulation accuracy, the emulator offers significant computational savings at prediction time over the FEM. A single forward evaluation for the `LeftVentricle` model takes 2.7×10^{-3} s, and once the message-passing stage is precomputed, only 2.3×10^{-4} s.[5] By contrast, simulations of the `LeftVentricle` model in FEniCS can take in excess of one hour.[6] Simulation times could however be reduced to the order of minutes by exploring parallel computing and solver optimisation in software such as ABAQUS. Nevertheless, even a one minute simulation time is prohibitively expensive for real-time applications where thousands of simulations need to be performed in sequence, as would be required for example with the use of a sampling-based Bayesian inference method for an inverse problem. As a consequence of the reduction in the computational costs, Bayesian sampling with our method becomes practically feasible. This indicates that our methodological improvements enable sound parameter inference with proper uncertainty

[5] Dual NVIDIA RTX A6000.

[6] Dual Xeon Gold 6254.



(a) Reference Configuration

(b) Current Configuration (FEM)

(c) $err_{\boldsymbol{u}} = \|\boldsymbol{u} - \hat{\boldsymbol{u}}\|_2$

(d) Current Configuration (PI-GNN)

Fig. 13. Median out of sample emulation results for `LeftVentricle` model (mm).

quantification in real time, which would otherwise would not be feasible, thereby paving the path to genuine impact in the clinic. It is also trivial to parallelise the GNN to handle multiple input configurations simultaneously, using the `vmap` functionality in the JAX library.

4. Discussion

4.1. Data-driven and physics-informed training comparison

The experiments from Section 3.1 reveal interesting differences between data-driven (DD) and physics-informed (PI) training. For the `OnceClampedBeam` model, the DD-GNN achieves better accuracy in prediction of the displacements, whereas under the `TwiceClampedBeam` model, where the magnitude of the displacements was lower, the results in displacement space were almost identical. However for both models, there is a clear divergence in how accurately the predictions of the two emulators recover the true minimum total potential energy state. For the DD-GNN, err_{Π} values tended to be in the range of 1%–10%, whereas the PI-GNN errors were typically at least one order of magnitude lower. This indicates that the PI-GNN is consistently capturing a more

physically realistic displacement that the DD-GNN. This is reflected in the errors observed in the prediction of the deformation gradient $\boldsymbol{F}$ and first invariant I_1, where for both models we see again lower error values incurred by the PI-GNN. In Appendix B we re-perform these experiments using a DD-GNN where the loss function also includes a penalty term on $\boldsymbol{F}$. We find that this leads to improved accuracy, but nevertheless PI-GNN achieves lower values of err_{Π}, $err_{\boldsymbol{F}}$ and err_{I_1}.

These results indicate that, while data-driven training may lead to accurate results in displacement space, these results are not guaranteed to respect the underlying physics. Therefore if further quantities of interest are required beyond the displacements, for example stress and strain values, the use of physics-informed training will lead to more accurate results.

4.2. Computational costs

The results from Table 2 indicate that an increase in mesh density leads to an increase in training and prediction times of a similar order of magnitude. We have not emphasised optimisation of training time in this work. Possible methods for doing so include mixed precision training, utilisation of multiple GPUs, or use of a second order optimisation approach such as conjugate gradients. Note that the particular advantage of a GNN emulator is that training can be done offline — again, see [56] for details.

Once the processor stage of the emulator is precomputed however, we see from Table 2 that prediction times only increase by a factor of three when the mesh density increase by approximately a factor of sixty. This illustrates the advantage of our architecture design, which combines the modelling benefits of a GNN with the computational efficiency of an FCNN at prediction time.

4.3. `Liver` *And* `LeftVentricle` *emulation results*

The strong emulation accuracy for the `Liver` and `LeftVentricle` models highlights the ability of our PI-GNN approach to handle realistic models involving complex soft-tissue geometries.[7] The PI-GNN can also be applied to a soft-tissue geometry on which it has not seen during the initial training phase, which is demonstrated in Appendix E.

Comparing the results for the two real soft-tissue models, we see that higher accuracy was obtained for the `Liver` model, which assumed a Neo-Hookean material, compared to the `LeftVentricle`, where the more nonlinear H-O material model was used. This suggests that it may be easier to train in a physics-informed manner for more linear constitutive laws. A further comparison is explored in Appendix C, where the `LeftVentricle` emulation experiments are re-performed under the Neo-Hookean model. The results indicate that the emulator can consistently obtain a better approximation to the true Π under the more linear model, with lower error values in both $\boldsymbol{u}$ and $\boldsymbol{F}$ observed in turn. Nevertheless, predictions for the `LeftVentricle` model using the Holzapfel–Ogden material model were still highly accurate. For instance, the worst error observed in LV cavity volume of approximately half a percent is an order of magnitude lower than typical error measurements from manual segmentation of CMR scans [79]. In addition to strong predictive accuracy, the emulator is also several orders of magnitude less computationally expensive at prediction time when compared to the FEM. Furthermore, our PI-GNN implementation can be automatically differentiated using JAX, allowing $\partial\hat{\boldsymbol{U}}/\partial\boldsymbol{\theta}$ to be computed to machine precision at negligible additional computational costs. This combination of rapid, highly accurate predictions with end-to-end differentiability completely changes the range of applications for which soft-tissue models can be deployed in real time. For example, the parameter inference problem for passive cardiac mechanics considered in [8], which took over one week using the FEM, could be performed in seconds using a PI-GNN emulator.

4.4. Limitations and future work

The principle of minimum total potential energy is a fundamental concept in mechanics and engineering, and has been used in this work to handle soft-tissue mechanics problems using PI-GNNs, in a similar manner to some recent studies [43,80]. An alternative approach can be derived from the weak formulation of PDE residuals based on the principle of virtual work, as done in [63]. With this approach, a physics-informed loss function may be defined as $L = \| \left(\int_{\Omega} \nabla \cdot \boldsymbol{\sigma} + \boldsymbol{b}\right) \cdot \boldsymbol{\eta} dv \|_2$, with $\boldsymbol{\eta}$ the test function. There are a number of potential advantages to this approach, including the flexibility of handling time-dependent dynamic problems, discontinuous problems and non-conservative systems, i.e. shocks.

The strain energy density functions of the myocardium used here are considered to be slightly compressible, making use of the so-called $\bar{F}$-bar method. To fully address incompressibility, future work can make use of the so-called multi-field variational principles [60, Chapter 8], where additional variables are introduced to take into account the incompressibility constraint.

In future directions of work we will extend the PI-GNN to consider higher-order finite elements with quadratic shape functions. This is to enhance the robustness of our method to the locking phenomenon which is a well-known issue when using linear tetrahedral elements for FE computations [81], in particular for incompressible or nearly-incompressible materials. A wider range of PDE systems with unstructured data could also be considered, for example material science [82], fluid dynamics (i.e. arterial blood flow [38,83]), structural mechanics [63], and other large-scale engineering systems [84].

In this study, a preliminary mesh convergence study was carried out using the `TwistingCube` model (see Table 2). These results show that PI-GNN can match the FEM results well for different mesh densities, thus it can be expected that with increased mesh density, the emulator will obtain more accurate and reliable results. In real applications for clinical decision making, the PI-GNN may also need a fine discretisation as is the case for the FEM. However, the mesh density required by the surrogate model

[7] In Appendix D we also consider emulation involving a biventricle cardiac geometry.

might be different from the classical FEM. There are some studies in the literature which explore this problem from a theoretical perspective, see for example He et al. [85] which draws an analogy between ReLU deep neural networks and linear finite elements.

Finally, the natural progression of the present work is to consider the inverse problem, that is to infer material parameters for soft-tissue bodies from clinical data using a PI-GNN, and quantify the inference uncertainty. For the left ventricle, we have previously found that some parameters are weakly identifiable given observed strain values at 24 landmark points on the myocardium [12]. Emulation of the entire displacement field with a PI-GNN may provide important additional information that will reduce the posterior uncertainty. However, a verification of this hypothesis is beyond the remit of the present article and will be addressed in our future work.

5. Conclusion

This paper has presented a PI-GNN emulation framework for application to soft-tissue mechanics. The GNN can operate directly on the unstructured mesh representation of a given soft-tissue geometry and is trained in a physics-informed manner by applying the principle of minimum total potential energy. Physics-informed training is enabled by the introduction of barrier transformation functions which stabilise the objective function by explicitly incorporating known physical constraints such as the impenetrability of matter. A range of hyper-elastic models are considered, including realistic models of the human liver and left ventricle. Furthermore, significant computational savings at prediction time are made compared to the FEM. The authors believe this work is an important step in the development of real-time clinical applications of computational soft-tissue mechanics.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Data availability

Data will be made available on request.

Acknowledgements

This work has been funded by the Engineering and Physical Sciences Research Council (EPSRC) of the United Kingdom, grant reference numbers EP/T017899/1, EP/S030875/1 and EP/S020950/1.

Appendix A. Effect of learning rate on emulator training

We use four beam models to explore the effect that learning rate has on emulator training. The first two models, `OnceClampedBeam` and `TwiceClampedBeam` were also considered in Section refsec:ddComparison for the data driven and physics informed training comparison. For the third model `TractionBeamHO`, we have $\Omega_0 = [0, 100] \times [0, 10]^2$ (mm), represented using 176 nodes and 440 elements. The beam is clamped at the left most end $\partial\Omega_0^d = \{\boldsymbol{X} \in \Omega_0 : X_1 = 0\}$, and subject to a pressure of 0.15 kPa on its bottom surface $\partial\Omega_0^\sigma = \{\boldsymbol{X} \in \Omega_0 : X_3 = 0\}$, with no body force applied. The H-O constitutive law is used (see Eq. (4)), with $\boldsymbol{\theta} \in [.75, 2.5]^4$ where a and a_f are in kPa, and a fixed myofibre field with $\boldsymbol{f}_0 = (1, 0, 0)^\top$ is assumed throughout Ω_0. The setup of the final beam model, `TractionBeamGucci` is identical with the exception that the Guccione material model is used (see Eq. (7)), with $\boldsymbol{\theta} \in [1.75, 3.5]^4$. The reference configuration and boundary conditions for both traction-force models is shown in Fig. A.14.

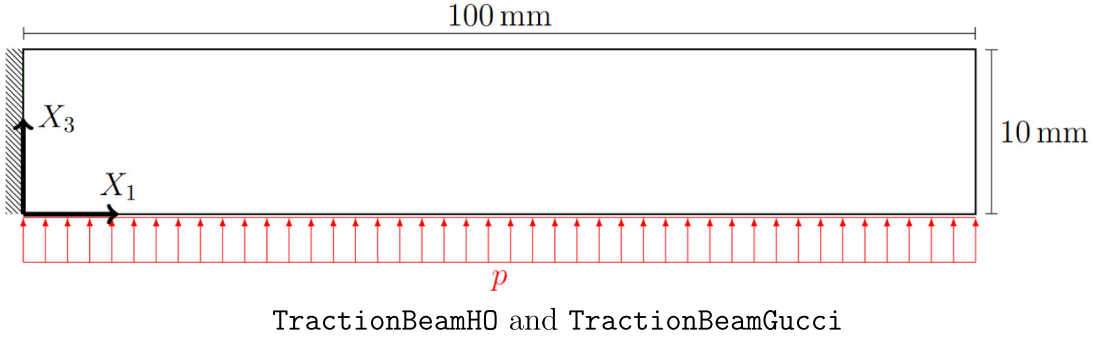


Fig. A.14. Illustration of the `TractionBeamHO` and `TractionBeamGucci` models as 2-D slices in the (X_1, X_3) plane (not to scale). The dashed lines indicate a clamped Dirichlet boundary, and p represents a pressure applied to a Neumann boundary surface.

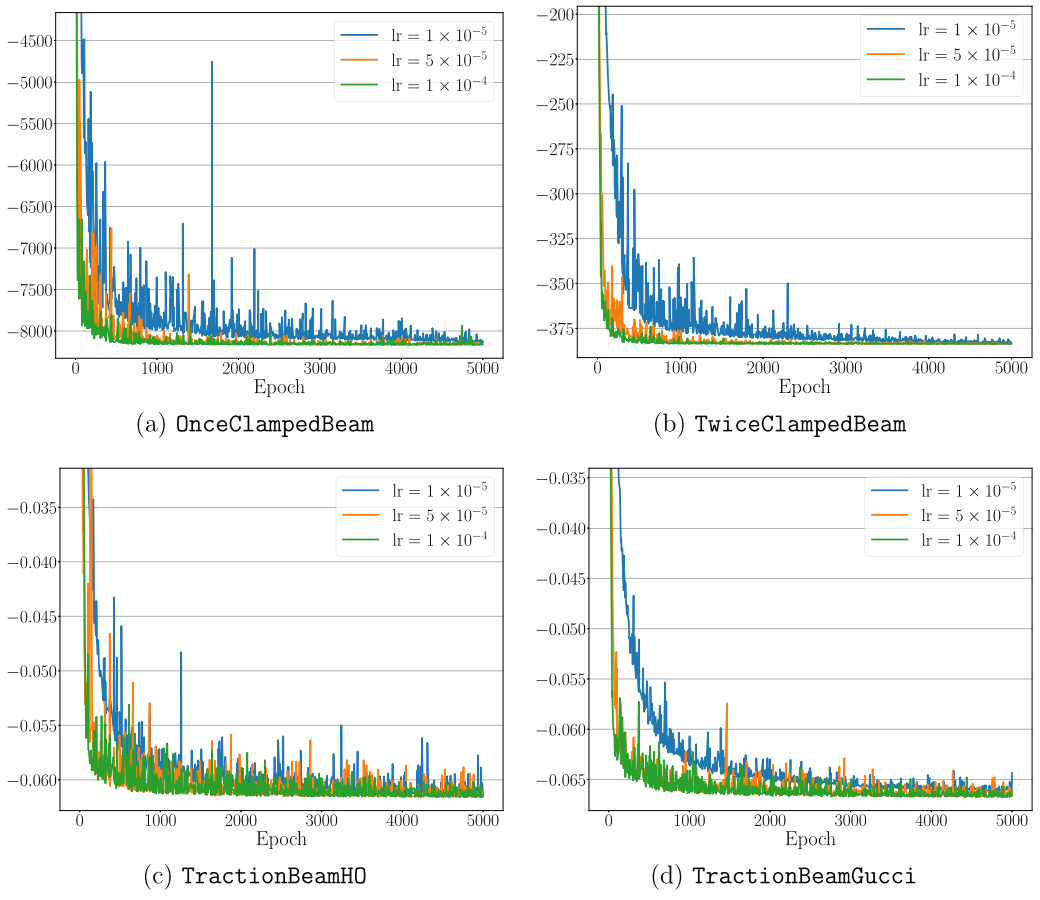


Fig. A.15. Traceplots of mean(Π) for four beam models using different learning rates.

For each model, training was performed over 200 randomly sampled input material parameter values. An additional 100 independent simulations were used as a test set for evaluation of out of sample performance. Three different fixed learning rates were considered; 1×10^{-5}, 5×10^{-5} and 1×10^{-4}. 5000 epochs were used for training, which was a sufficient number to make comparisons between the different rates. Learning curves for the four beam models are displayed in Fig. A.15, with the mean potential energy value obtained at the 200 training input configurations plotted against number of training epochs performed. For all models, the higher learning rate of 1×10^{-4} allows the emulator to reach the minimum of the potential energy most rapidly. Training diverged in each case when a higher learning rate of 5×10^{-4} was used(not shown in figure to avoid clutter). Fig. A.16 again displays learning curves, except here the mean $err_{\boldsymbol{u}}$ value across the 100 test simulations is plotted against number of epochs. A similar pattern is observed, whereby a high learning rate yields faster training. Significant variation in the $err_{\boldsymbol{u}}$ traceplots are present however when using a higher learning rate, suggesting that lowering its value at the end of training may be useful to settle on the optimal displacement values. For this reason we used a two-phase training approach for the emulation experiments in Sections 3.2–3.4 whereby a learning rate of 1×10^{-4} was used for the first half of training, before this was reduced by one order of magnitude for the second half of training. We found that this simple strategy was highly effective at finding the minimum total potential energy state, and hence we did not pursue more complex approaches in this work, such as second order methods. Note that we also initially experimented with an automatic learning rate finder approach similar to [86] and different decay schedules, however we consistently found that using a fixed learning rate of 1×10^{-4} was optimal during the early stages of training.

Appendix B. Additional data-driven and physics-informed emulation experiments

The results from Section 3.1 illustrate that while data-driven can achieve strong accuracy in terms of displacements, the results obtained do not approximate the true potential energy state to the same accuracy as with physics-informed training, which is reflected in greater errors in the deformation gradient $\boldsymbol{F}$ and first invariant I_1. A possible way of alleviating this issue would be to train a data-driven emulator against both displacement and deformation gradient data. To test this approach, we re-performed the

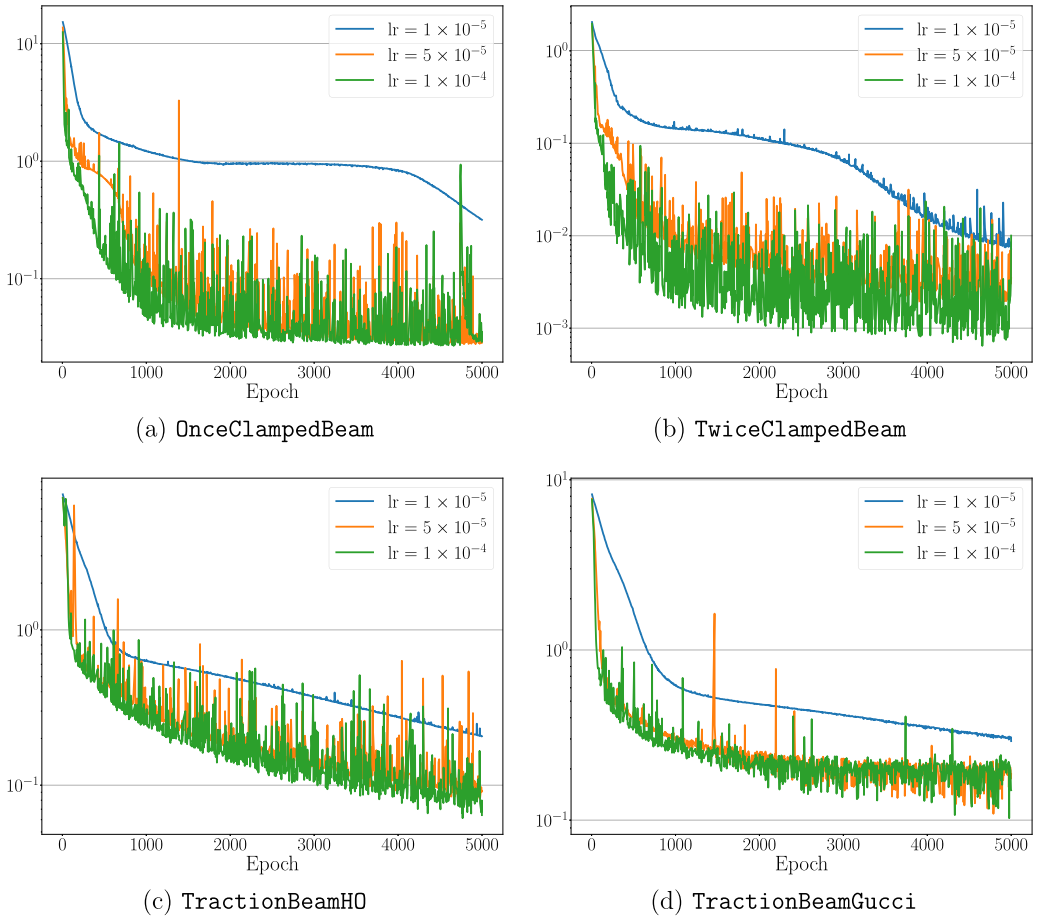


Fig. A.16. Traceplots of mean($err_{\boldsymbol{u}}$) for four beam models using different learning rates.

data-driven emulation experiments by making use of the following loss function

$$\boldsymbol{\omega}^* = \underset{\boldsymbol{\omega}}{\text{argmin}}\ \text{Mean}\left[\sum_{j=1}^{N_{\text{sim}}}\sum_{i=1}^{N_{\text{node}}} \|\boldsymbol{u}_{ji} - \hat{\boldsymbol{u}}_{ji}(\boldsymbol{\omega})\|_2 + \sum_{j=1}^{N_{\text{sim}}}\sum_{k=1}^{N_{\text{elem}}} \|\boldsymbol{F}_{jk} - \hat{\boldsymbol{F}}_{jk}(\boldsymbol{\omega})\|_F\right], \tag{B.1}$$

which includes a penalty term for in terms of the Frobenius norm $\|cdot\|_F$. We refer to a GNN trained using this loss function as an "FDD-GNN". Figs. B.17 and B.18 then replicate the corresponding figures from Section 3.1 for both the `OnceClampedBeam` and `TwiceClampedBeam` models, except this time the FDD-GNN results are also included. For each model, we see that including a penalty term on $\boldsymbol{F}$ in the data-driven loss function leads to improved emulation accuracy, particularly in terms of the err_{Π}, $err_{\boldsymbol{F}}$ and err_{I_1}, but nevertheless the PI-GNN still achieves better accuracy on these three error metrics.

Appendix C. Comparison of Neo-Hookean and Holzapfel–Ogden material models

The higher level of accuracy obtained for the `Liver` model in Section 3.3, which used the Neo-Hookean law, compared to the `LeftVentricle` emulation results in Section 3.4, where the Holzapfel–Ogden law was assumed, suggests that the PI-GNN is easier to train when the underlying constitutive relationship is linear or near-linear. For further insight here, in this section we re-perform the `LeftVentricle` emulation experiments using the Neo-Hookean material model. Specifically, was trained over material parameter space and LV pressure profiles between 4 and 10 mmHg, with Neo-Hookean constitutive law parameterised by Lame material parameters λ and μ, where both parameters varied in the range $[8, 10]$ kPa. Emulator performance of the surrogate was evaluated on a test set of 150 simulations obtained using the finite-element method (FEM). Fig. C.19 displays error density plots for this emulator, compared with results from Section 3.4 for the original PI-GNN which made use of the H-O law. In all cases, we see noticeably better performance under the Neo-Hookean model. For relative errors in u seen in panel (a), median errors under the Neo-Hookean model are almost one order of magnitude lower than for the H-O model. For err_{I_1} displayed in panel (b), there is even greater disparity between the results, with very little overlap between the two densities. If we consider err_{Π} in panel (c), we see that the PI-GNN consistently achieves a better approximation to the true minimum potential energy value, indicating it is easier to

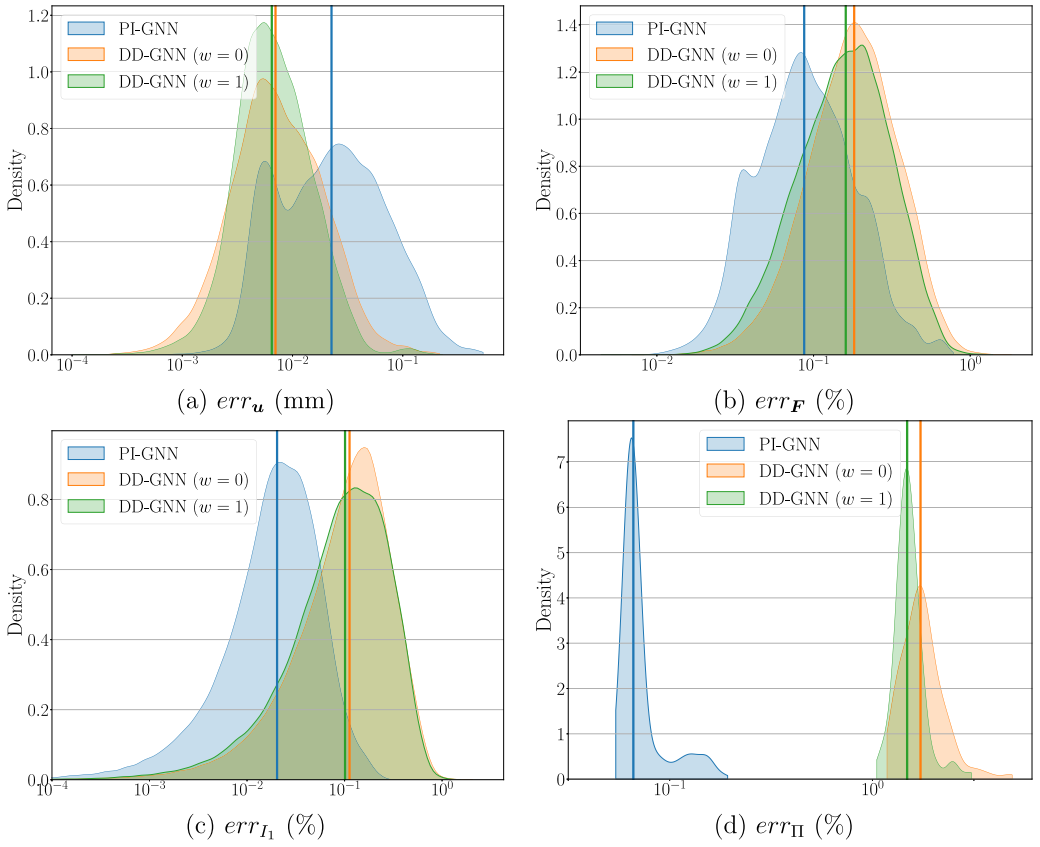


Fig. B.17. Comparison of data-driven and physics-informed emulation results on `OnceClampedBeam` model. The PI-GNN is following Eq. (16) of the manuscript, the DD-GNN is trained purely on displacement data as in Eq. (15) of the manuscript, while the FDD-GNN additionally incorporates a loss on the deformation gradient $\boldsymbol{F}$ following Eq. (B.1) above.

train the emulator when the material model is near linear. Finally, for err_V in panel (d), the difference between the two results is less pronounced. While lower errors are achieved under the Neo-Hookean model, there is a high degree of overlap between the two distributions. This results suggest that further experimentation could be required here to determine how PI-GNN accuracy can be increased for more non-linear material models. One possible approach could be the use of feature transformations of the material parameter vector $\boldsymbol{\theta}$.

Appendix D. Application to biventricle cardiac geometry

Here we demonstrate the application of a PI-GNN to the filling of a biventricular cardiac geometry. For this illustrative example, idealised symmetric ventricles are considered, the Neo-Hookean material model is assumed and the same cardiac pressure is used in each cavity. A PI-GNN emulator was trained for 1000 steps. Prediction results against an FEM simulation are shown in Fig. D.20, where the mean $err_{\boldsymbol{u}}$ value is 4.8×10^{-2} mm. Visually we see strong agreement, however relative errors are higher than seen in the `LeftVentricle` model, which may be alleviated by more training steps or a larger number of message passing steps.

Appendix E. Emulation on new LV geometry

The geometry of a soft-tissue body will vary from patient to patient, therefore it is essential that the emulator can generalise to a new geometry not seen in the training phase. Note we have already addressed this in earlier work [56], where a GNN was trained on a dataset where each simulation was performed using a different left ventricle geometry. In this work, we take a simpler approach by considering the generalisation of the PI-GNN for the `LeftVentricle` model in the case of one new LV geometry, again extracted from cardiac imaging scans. Note that this geometry had a different mesh topology (1268 nodes and 4847 elements) to the original LV (1570 nodes and 6176 elements).

To test how the GNN can generalise to the new geometry, we took the pre-trained PI-GNN from the original LV, and transfer learned for 12 epochs or approximately 90 s on the new geometry (we refer to this as the "Transfer Learned" emulator). For baseline comparison, we consider a PI-GNN trained for the same computational budget (i.e. 12 steps/90 s), but from randomly initialised network parameters (we refer to this as the "Baseline" emulator). Comparing the transfer learned results with the baseline allows the gain achieved by sharing information from the original geometry to be quantified. Finally, as reference we consider the results

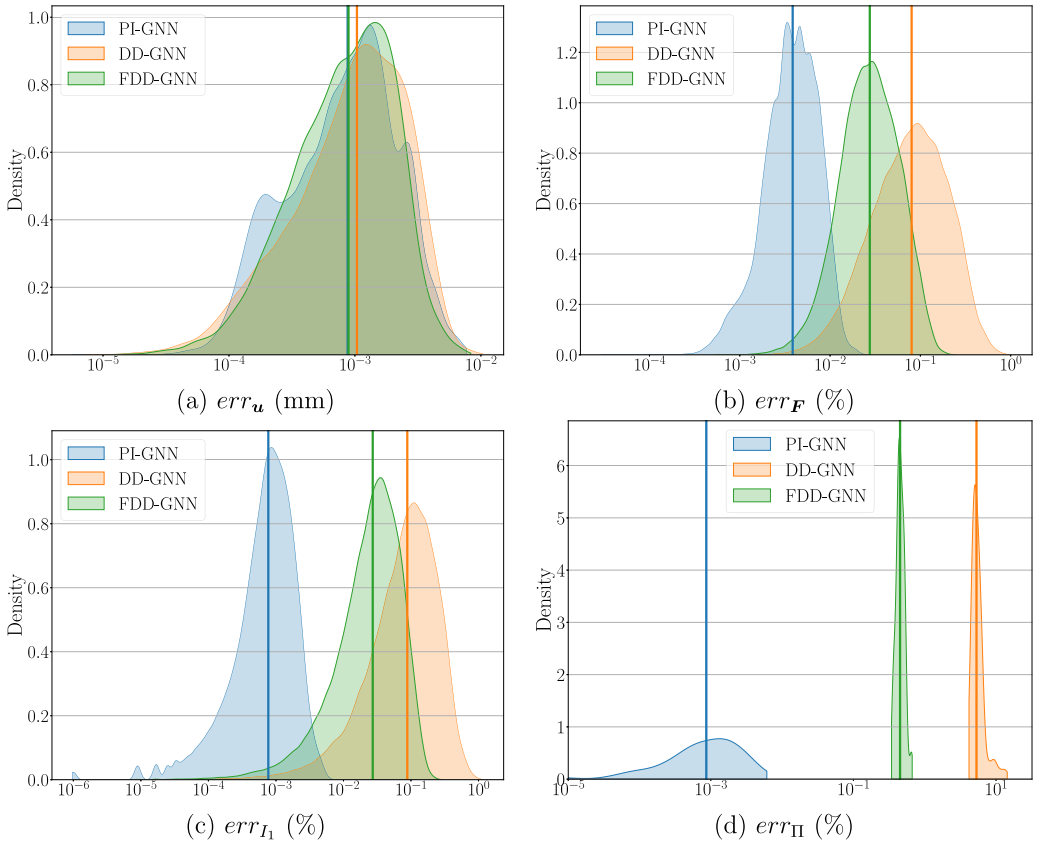


Fig. B.18. Comparison of data-driven and physics-informed emulation results on `TwiceClampedBeam` model. The PI-GNN is following Eq. (16) of the manuscript, the DD-GNN is trained purely on displacement data as in Eq. (15) of the manuscript, while the FDD-GNN additionally incorporates a loss on the deformation gradient $\boldsymbol{F}$ following Eq. (B.1) above.

we the performance of a PI-GNN again trained from scratch, but this time for 15 000 epochs (we refer to this as the "Reference" emulator). After training, emulation performance of the three emulators was then evaluated on a test set of 150 simulations run using the same material model (Holzapfel–Ogden) and material parameter domain as the original LV geometry, with fixed cardiac pressure of 6 mmHg.

Fig. E.21 shows density plots of test set emulation errors for the three emulators considered. In each case, we see a clear gain in performance when re-using the network weights over the Baseline PI-GNN with randomly initialised weights. With $err_{\boldsymbol{u}}$ for instance, the increase in accuracy is approximately one order of magnitude. The accuracy of the full-trained Reference emulator is then one order of magnitude lower again. Performing over one thousand times as many training steps has allowed the Reference emulator to obtain clearly more accurate results.

Panel (c) shows the error in total potential energy Π incurred by the emulator, which illustrate allowed the fully-trained Reference PI-GNN to reach a better approximation of the true minimum potential energy state than the two PI-GNNs that were trained for 12 steps. However, the transfer learned emulator has clearly benefited from the initialisation at the weights found on the other LV geometry, as the worst err_{Π} is lower seen in the transfer-learning case is lower than the smallest error seen with the Baseline, randomly initialised PI-GNN. A similar pattern is observed in panel (d), which displays density plots of err_V. Here however the transfer-learned emulator more closely matches the performance of the fully-trained Reference emulator. While lower errors are attained in general with full training, there is significant overlap between the two error distributions, and for the transfer-learned PI-GNN, only a small fraction of errors exceed 1%.

Fig. E.22 displays emulation plots for the new geometry for the simulation which gave median out of sample error. In the left column, we see that the errors incurred by the transfer learned PI-GNN are consistently higher across the surface of the geometry than for the Reference PI-GNN. In the right column, differences between the two are only slightly visually apparent, and the err_V values were very similar for each surrogate — for the Reference PI-GNN, err_V error was 0.03%, while for the transfer-learned PI-GNN, the corresponding error was 0.04%.

These results suggest that for macro level quantities of interest such as cardiac volume, where precise node-wise accuracy levels, the PI-GNN exhibits reasonable generalisation performance when applied to a new geometry. It is likely that more accurate results can be obtained by incorporating a larger number of different geometries into the training routine as done in [56], which is the direction we will pursue in future work.

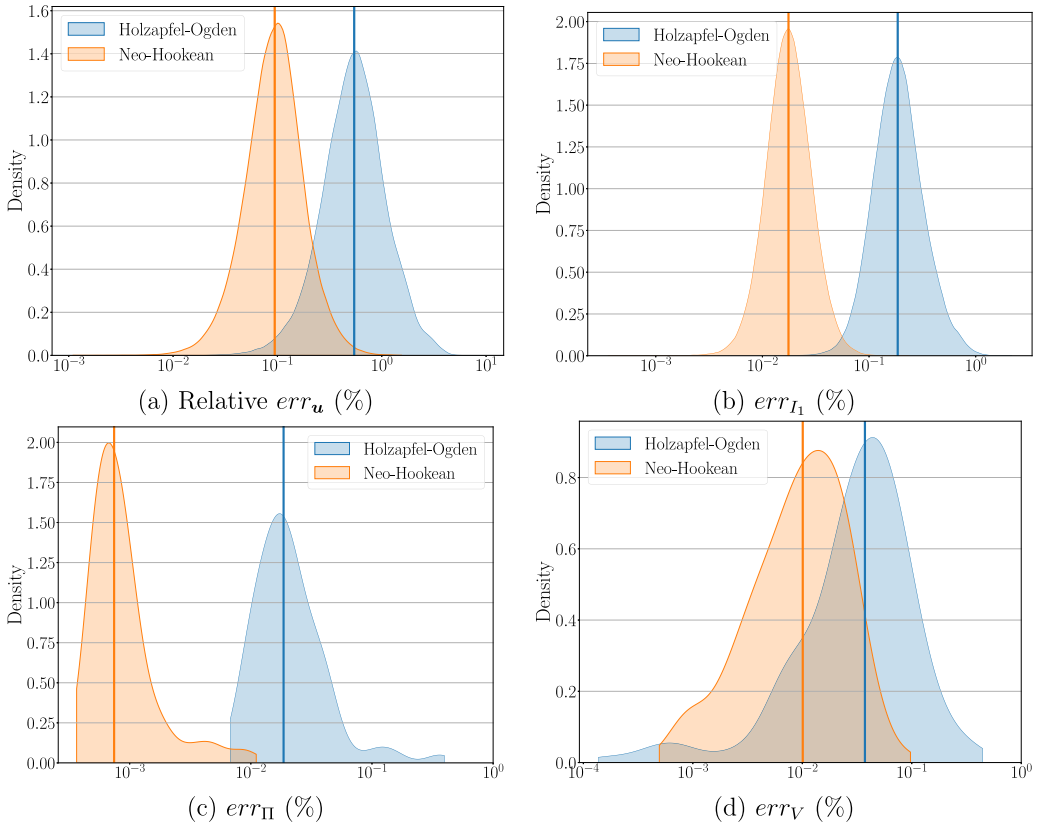
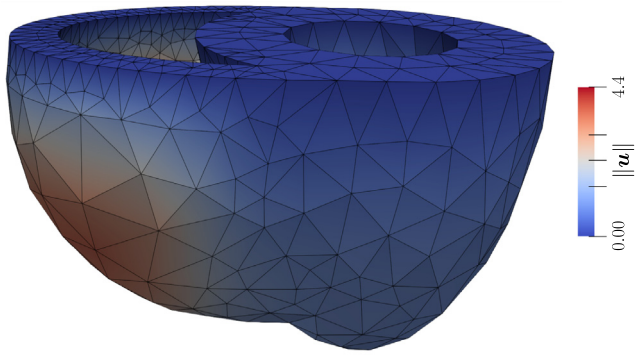
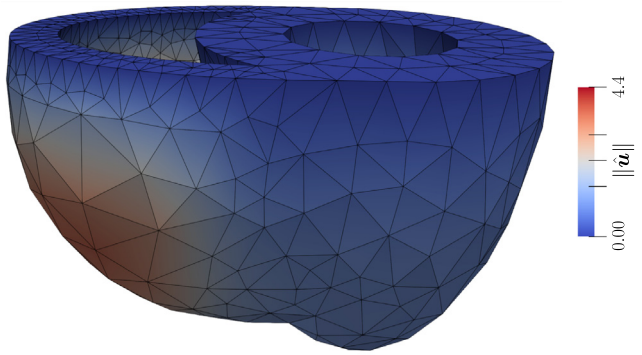


Fig. C.19. Comparison of emulation results for `LeftVentricle` model under the Neo-Hookean and Holzapfel–Ogden material models.



(a) Current Configuration (FEM)



(b) Current Configuration (PI-GNN)

Fig. D.20. Comparison of simulation and emulation results for biventricular cardiac geometry.

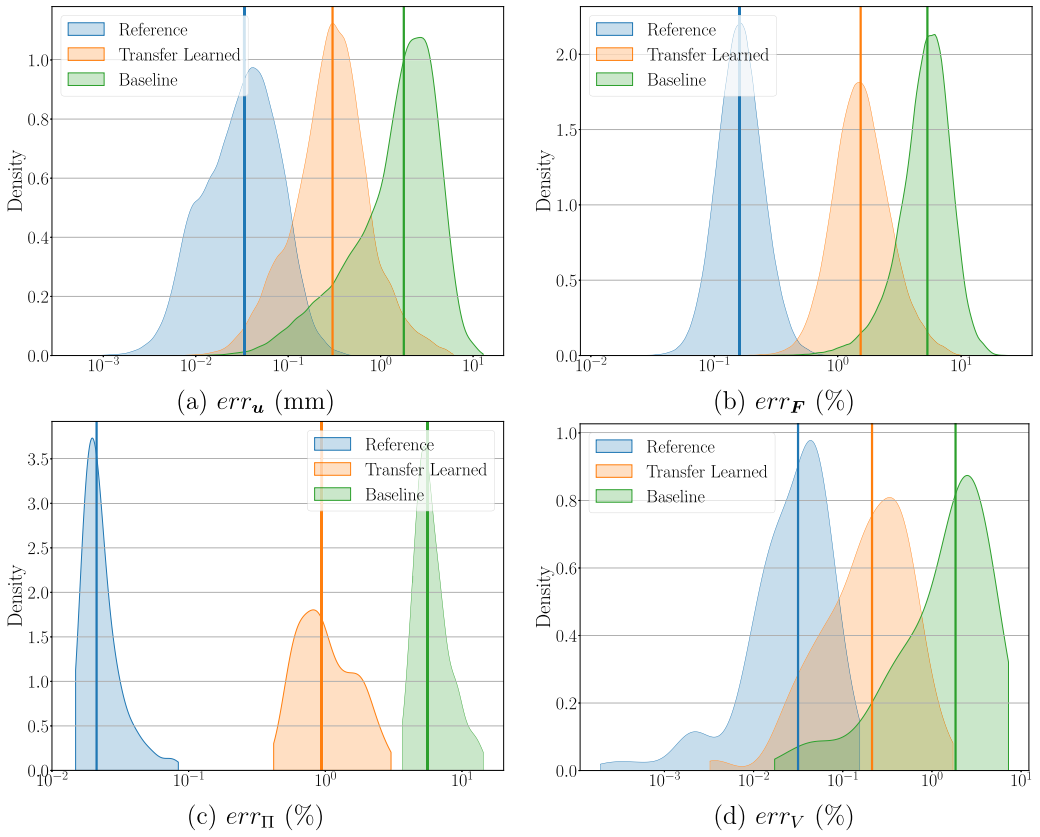
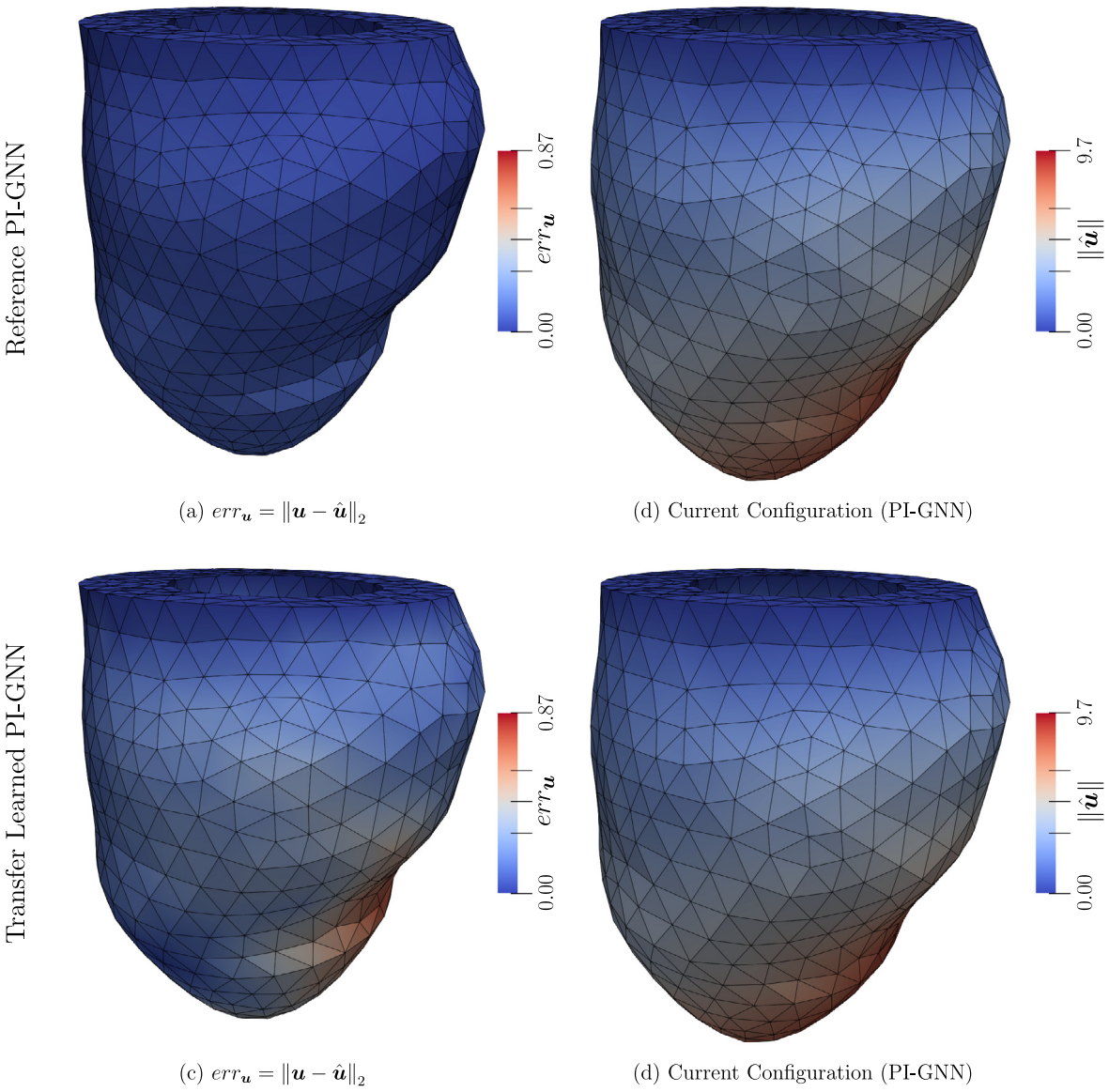


Fig. E.21. Comparison of emulation results for new LV geometry for GNN trained from scratch for 15 000 epochs (Reference), versus a transfer-learned GNN trained for 12 epochs (Transfer Learned) and a randomly initialised GNN trained for 12 epochs (Baseline).



(a) $err_{\boldsymbol{u}} = \|\boldsymbol{u} - \hat{\boldsymbol{u}}\|_2$ (d) Current Configuration (PI-GNN)

(c) $err_{\boldsymbol{u}} = \|\boldsymbol{u} - \hat{\boldsymbol{u}}\|_2$ (d) Current Configuration (PI-GNN)

Fig. E.22. Median out of sample emulation results for `LeftVentricle` model (mm) using new geometry. Top row shows a fully trained PI-GNN, bottom row shows a PI-GNN transfer-learned for 90 s.

References

[1] A. Al-Mayah, Biomechanics of Soft Tissues: Principles and Applications, CRC, 2018.

[2] S. Marchesseau, S. Chatelin, H. Delingette, Nonlinear biomechanical model of the liver, in: Biomechanics of Living Organs, Elsevier, ISBN: 978-0-12-804009-6, 2017, pp. 243–265, http://dx.doi.org/10.1016/B978-0-12-804009-6.00011-0, URL https://linkinghub.elsevier.com/retrieve/pii/B9780128040096000110.

[3] G.A. Holzapfel, R.W. Ogden, Constitutive modelling of passive myocardium: a structurally based framework for material characterization, Philos. Trans. R. Soc. Lond. A 367 (1902) (2009) 3445–3475.

[4] S. Budday, T.C. Ovaert, G.A. Holzapfel, P. Steinmann, E. Kuhl, Fifty shades of brain: A review on the mechanical testing and modeling of brain tissue, Arch. Comput. Methods Eng. 27 (4) (2020) 1187–1230, http://dx.doi.org/10.1007/s11831-019-09352-w, URL http://link.springer.com/10.1007/s11831-019-09352-w.

[5] V. Klika, E.A. Gaffney, Y.-C. Chen, C.P. Brown, An overview of multiphase cartilage mechanical modelling and its role in understanding function and pathology, J. Mech. Behav. Biomed. Mater. 62 (2016) 139–157.

[6] G.A. Holzapfel, R.W. Ogden, An arterial constitutive model accounting for collagen content and cross-linking, J. Mech. Phys. Solids (ISSN: 00225096) 136 (2020) 103682, http://dx.doi.org/10.1016/j.jmps.2019.103682, URL https://linkinghub.elsevier.com/retrieve/pii/S0022509619305642.

[7] N.M.E. Ayad, S. Kaushik, V.M. Weaver, Tissue mechanics, an important regulator of development and disease, Philos. Trans. R. Soc. B 374 (1779) (2019) 20180215, http://dx.doi.org/10.1098/rstb.2018.0215, URL https://royalsocietypublishing.org/doi/10.1098/rstb.2018.0215.

[8] H. Gao, A. Aderhold, K. Mangion, X. Luo, D. Husmeier, C. Berry, Changes and classification in myocardial contractile function in the left ventricle following acute myocardial infarction, J. R. Soc. Interface 14 (132) (2017) 20170203, http://dx.doi.org/10.1098/rsif.2017.0203.

[9] J. Corral-Acero, F. Margara, M. Marciniak, C. Rodero, F. Loncaric, Y. Feng, A. Gilbert, J.F. Fernandes, H.A. Bukhari, A. Wajdan, M.V. Martinez, M.S. Santos, M. Shamohammdi, H. Luo, P. Westphal, P. Leeson, P. DiAchille, V. Gurev, M. Mayr, L. Geris, P. Pathmanathan, T. Morrison, R. Cornelussen, F. Prinzen, T. Delhaas, A. Doltra, M. Sitges, E.J. Vigmond, E. Zacur, V. Grau, B. Rodriguez, E.W. Remme, S. Niederer, P. Mortier, K. McLeod, M. Potse, E. Pueyo, A. Bueno-Orovio, P. Lamata, The 'Digital Twin' to enable the vision of precision cardiology, Eur. Heart J. (2020) ehaa159, http://dx.doi.org/10.1093/eurheartj/ehaa159.

[10] S.A. Niederer, M.S. Sacks, M. Girolami, K. Willcox, Scaling digital twins from the artisanal to the industrial, Nat. Comput. Sci. 1 (5) (2021) 313–320.

[11] J.O. Campos, J. Sundnes, R.W. dos Santos, B.M. Rocha, Uncertainty quantification and sensitivity analysis of left ventricular function during the full cardiac cycle, Phil. Trans. R. Soc. A 378 (2173) (2020) 20190381, http://dx.doi.org/10.1098/rsta.2019.0381, URL https://royalsocietypublishing.org/doi/full/10.1098/rsta.2019.0381, Publisher: Royal Society.

[12] A. Lazarus, D. Dalton, D. Husmeier, H. Gao, Sensitivity analysis and inverse uncertainty quantification for the left ventricular passive mechanics, Biomech. Model. Mechanobiol. (2022) 1–30.

[13] G.D. Maso Talou, T.P. Babarenda Gamage, M.P. Nash, Efficient Ventricular Parameter Estimation Using AI-Surrogate Models, (ISSN: 1664-042X) 2021, http://dx.doi.org/10.3389/fphys.2021.732351, URL https://researchspace.auckland.ac.nz/handle/2292/58764, Accepted: 2022-04-21T23:59:53Z Artwork Medium: Electronic-eCollection Interview Medium: Electronic-eCollection Publisher: Frontiers Media SA.

[14] D. Neumann, T. Mansi, B. Georgescu, A. Kamen, E. Kayvanpour, A. Amr, F. Sedaghat-Hamedani, J. Haas, H. Katus, B. Meder, J. Hornegger, D. Comaniciu, Robust image-based estimation of cardiac tissue parameters and their uncertainty from noisy data, Med. Image Comput. Comput.-Assist. Interv.: MICCAI ... Int. Conf. Med. Image Comput. Comput.-Assist. Interv. 17 (Pt 2) (2014) 9–16, http://dx.doi.org/10.1007/978-3-319-10470-6_2.

[15] M. Hadjicharalambous, R. Chabiniok, L. Asner, E. Sammut, J. Wong, G. Carr-White, J. Lee, R. Razavi, N. Smith, D. Nordsletten, Analysis of passive cardiac constitutive laws for parameter estimation using 3D tagged MRI, Biomech. Model. Mechanobiol. (ISSN: 1617-7940) 14 (4) (2015) 807–828, http://dx.doi.org/10.1007/s10237-014-0638-9.

[16] S. Monaci, M. Strocchi, C. Rodero, K. Gillette, J. Whitaker, R. Rajani, C.A. Rinaldi, M. O'Neill, G. Plank, A. King, et al., In-silico pace-mapping using a detailed whole torso model and implanted electronic device electrograms for more efficient ablation planning, Comput. Biol. Med. 125 (2020) 104005.

[17] G. Luraghi, J.F. Rodriguez Matas, F. Migliavacca, In silico approaches for transcatheter aortic valve replacement inspection, Expert Rev. Cardiovasc. Ther. 19 (1) (2021) 61–70.

[18] L. Paun, A. Schmidt, S. Mcginty, D. Husmeier, Statistical inference for optimisation of drug delivery from stents, in: Proceedings of the International Conference on Statistics: Theory and Applications, 2022, http://dx.doi.org/10.11159/icsta22.138.

[19] P. Wriggers, Nonlinear Finite Element Methods, Springer Science & Business Media, 2008.

[20] M.C. Kennedy, A. O'Hagan, Bayesian calibration of computer models, J. R. Stat. Soc. Ser. B Stat. Methodol. (ISSN: 1467-9868) 63 (3) (2001) 425–464, http://dx.doi.org/10.1111/1467-9868.00294, URL https://onlinelibrary.wiley.com/doi/abs/10.1111/1467-9868.00294, _eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1111/1467-9868.00294.

[21] R.B. Gramacy, Surrogates: Gaussian Process Modeling, Design and Optimization for the Applied Sciences, Chapman Hall/CRC, Boca Raton, Florida, 2020, http://bobby.gramacy.com/surrogates/.

[22] C.E. Rasmussen, K.I. Williams, Gaussian Processes for Machine Learning, MIT Press, Cambridge, MA., 2006.

[23] C. Soize, R. Ghanem, Physical Systems with Random Uncertainties: Chaos Representations with Arbitrary Probability Measure, SIAM J. Sci. Comput. (ISSN: 1064-8275) 26 (2) (2004) 395–410, http://dx.doi.org/10.1137/S1064827503424505, URL https://epubs.siam.org/doi/10.1137/S1064827503424505, Publisher: Society for Industrial and Applied Mathematics.

[24] K.P. Murphy, Probabilistic Machine Learning: An Introduction, MIT Press, 2022, URL probml.ai.

[25] Y. Guan, A. Chattopadhyay, A. Subel, P. Hassanzadeh, Stable a posteriori LES of 2D turbulence using convolutional neural networks: Backscattering analysis and generalization to higher Re via transfer learning, J. Comput. Phys. (ISSN: 0021-9991) 458 (2022) 111090, http://dx.doi.org/10.1016/j.jcp.2022.111090, URL https://www.sciencedirect.com/science/article/pii/S0021999122001528.

[26] D. Rajaram, T.G. Puranik, A. Renganathan, W.J. Sung, O.J. Pinon-Fischer, D.N. Mavris, A. Ramamurthy, Deep Gaussian process enabled surrogate models for aerodynamic flows, in: AIAA Scitech 2020 Forum, in: AIAA SciTech Forum, American Institute of Aeronautics and Astronautics, 2020, http://dx.doi.org/10.2514/6.2020-1640, URL https://arc.aiaa.org/doi/10.2514/6.2020-1640.

[27] A. Chattopadhyay, A. Subel, P. Hassanzadeh, Data-Driven Super-Parameterization Using Deep Learning: Experimentation With Multiscale Lorenz 96 Systems and Transfer Learning, J. Adv. Modelling Earth Syst. (ISSN: 1942-2466) 12 (11) (2020) e2020MS002084, http://dx.doi.org/10.1029/2020MS002084, URL https://onlinelibrary.wiley.com/doi/abs/10.1029/2020MS002084, _eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1029/2020MS002084.

[28] N. Chen, Y. Li, H. Liu, Conditional Gaussian nonlinear system: A fast preconditioner and a cheap surrogate model for complex nonlinear systems, Chaos (ISSN: 1054-1500) 32 (5) (2022) 053122, http://dx.doi.org/10.1063/5.0081668, URL https://aip.scitation.org/doi/10.1063/5.0081668, Publisher: American Institute of Physics.

[29] H. Lee, I.S. Kang, Neural algorithm for solving differential equations, J. Comput. Phys. (ISSN: 0021-9991) 91 (1) (1990) 110–131, http://dx.doi.org/10.1016/0021-9991(90)90007-N, URL https://www.sciencedirect.com/science/article/pii/002199919090007N.

[30] A. Meade, A. Fernandez, The numerical solution of linear ordinary differential equations by feedforward neural networks, Math. Comput. Modelling (ISSN: 0895-7177) 19 (12) (1994) 1–25, http://dx.doi.org/10.1016/0895-7177(94)90095-7, URL https://www.sciencedirect.com/science/article/pii/0895717794900957.

[31] I. Lagaris, A. Likas, D. Fotiadis, Artificial neural networks for solving ordinary and partial differential equations, IEEE Trans. Neural Netw. 9 (5) (1998) 987–1000, http://dx.doi.org/10.1109/72.712178.

[32] T. Graepel, Solving noisy linear operator equations by Gaussian processes: Application to ordinary and partial differential equations, in: Proceedings, Twentieth International Conference on Machine Learning, Vol. 1, 2003, pp. 234–241, URL https://www.scopus.com/inward/record.uri?eid=2-s2.0-1942421122&partnerID=40&md5=9e2cf2fed04ff19a053f90a2b439992f, Cited by: 45.

[33] S. Särkkä, Linear operators and stochastic partial differential equations in Gaussian process regression, in: T. Honkela, W. Duch, M. Girolami, S. Kaski (Eds.), Artificial Neural Networks and Machine Learning – ICANN 2011, Springer Berlin Heidelberg, Berlin, Heidelberg, ISBN: 978-3-642-21738-8, 2011, pp. 151–158.

[34] M. Raissi, P. Perdikaris, G.E. Karniadakis, Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations, J. Comput. Phys. (ISSN: 0021-9991) 378 (2019) 686–707, http://dx.doi.org/10.1016/j.jcp.2018.10.045, URL https://www.sciencedirect.com/science/article/pii/S0021999118307125.

[35] E. Weinan, B. Yu, The Deep Ritz Method: A deep learning-based numerical algorithm for solving variational problems | SpringerLink, 2018, pp. 1–12, URL https://link.springer.com/article/10.1007/s40304-018-0127-z.

[36] V.M. Nguyen-Thanh, X. Zhuang, T. Rabczuk, A deep energy method for finite deformation hyperelasticity, Eur. J. Mech. A Solids (ISSN: 0997-7538) 80 (2020) 103874, http://dx.doi.org/10.1016/j.euromechsol.2019.103874, URL https://www.sciencedirect.com/science/article/pii/S0997753819305352.

[37] J. He, D. Abueidda, S. Koric, I. Jasiuk, On the use of graph neural networks and shape-function-based gradient computation in the deep energy method, Internat. J. Numer. Methods Engrg. (ISSN: 1097-0207) 124 (4) (2023) 864–879, http://dx.doi.org/10.1002/nme.7146, URL https://onlinelibrary.wiley.com/doi/abs/10.1002/nme.7146, _eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1002/nme.7146.

[38] G. Kissas, Y. Yang, E. Hwuang, W.R. Witschey, J.A. Detre, P. Perdikaris, Machine learning in cardiovascular flows modeling: Predicting arterial blood pressure from non-invasive 4D flow MRI data using physics-informed neural networks, Comput. Methods Appl. Mech. Engrg. (ISSN: 0045-7825) 358 (2020) 112623, http://dx.doi.org/10.1016/j.cma.2019.112623, URL https://www.sciencedirect.com/science/article/pii/S0045782519305055.

[39] A. Kovacs, L. Exl, A. Kornell, J. Fischbacher, M. Hovorka, M. Gusenbauer, L. Breth, H. Oezelt, M. Yano, N. Sakuma, A. Kinoshita, T. Shoji, A. Kato, T. Schrefl, Conditional physics informed neural networks, Commun. Nonlinear Sci. Numer. Simul. (ISSN: 1007-5704) 104 (2022) 106041, http://dx.doi.org/10.1016/j.cnsns.2021.106041, URL https://www.sciencedirect.com/science/article/pii/S1007570421003531.

[40] M. Islam, M.S.H. Thakur, S. Mojumder, M.N. Hasan, Extraction of material properties through multi-fidelity deep learning from molecular dynamics simulation, Comput. Mater. Sci. (ISSN: 0927-0256) 188 (2021) 110187, http://dx.doi.org/10.1016/j.commatsci.2020.110187, URL https://www.sciencedirect.com/science/article/pii/S0927025620306789.

[41] G.E. Karniadakis, I.G. Kevrekidis, L. Lu, P. Perdikaris, S. Wang, L. Yang, Physics-informed machine learning, Nat. Rev. Phys. (ISSN: 2522-5820) 3 (6) (2021) 422–440, http://dx.doi.org/10.1038/s42254-021-00314-5, URL https://www.nature.com/articles/s42254-021-00314-5, Number: 6 Publisher: Nature Publishing Group.

[42] Z. Hao, S. Liu, Y. Zhang, C. Ying, Y. Feng, H. Su, J. Zhu, Physics-informed machine learning: A survey on problems, methods and applications, 2022, URL https://arxiv.org/abs/2211.08064.

[43] S. Buoso, T. Joyce, S. Kozerke, Personalising left-ventricular biophysical models of the heart using parametric physics-informed neural networks, Med. Image Anal. (ISSN: 1361-8415) 71 (2021) 102066, http://dx.doi.org/10.1016/j.media.2021.102066, URL https://www.sciencedirect.com/science/article/pii/S1361841521001122.

[44] W. Zhang, D.S. Li, T. Bui-Thanh, M.S. Sacks, Simulation of the 3D hyperelastic behavior of ventricular myocardium using a finite-element based neural-network approach, Comput. Methods Appl. Mech. Engrg. (ISSN: 0045-7825) 394 (2022) 114871, http://dx.doi.org/10.1016/j.cma.2022.114871, URL https://www.sciencedirect.com/science/article/pii/S0045782522001724.

[45] A.S.-G. Tobias Pfaff, P.W. Battaglia, Learning mesh-based simulation with graph networks, in: In Proceedings of the International Conference on Learning Representations, 2021.

[46] J. He, C. Chadha, S. Kushwaha, S. Koric, D. Abueidda, I. Jasiuk, Deep energy method in topology optimization applications, Acta Mech. 234 (4) (2022) 1365–1379, http://dx.doi.org/10.1007/s00707-022-03449-3.

[47] J. Gilmer, S.S. Schoenholz, P.F. Riley, O. Vinyals, G.E. Dahl, Neural message passing for quantum chemistry, 2017, CoRR, abs/1704.01212 URL http://arxiv.org/abs/1704.01212.

[48] X.-M. Zhang, L. Liang, L. Liu, M.-J. Tang, Graph neural networks and their current applications in bioinformatics, Front. Genet. (ISSN: 1664-8021) 12 (2021) 690049, http://dx.doi.org/10.3389/fgene.2021.690049, URL https://www.ncbi.nlm.nih.gov/pmc/articles/PMC8360394/.

[49] M. Zheng, Y. Zhou, D. Ceylan, J. Barbič, A deep emulator for secondary motion of 3D characters, 2021, URL https://arxiv.org/abs/2103.01261.

[50] Q. Tan, N. Liu, X. Hu, Deep representation learning for social network analysis, Front. Big Data (ISSN: 2624-909X) 2 (2019) http://dx.doi.org/10.3389/fdata.2019.00002, URL https://www.frontiersin.org/article/10.3389/fdata.2019.00002.

[51] Z. Cui, K. Henrickson, R. Ke, Y. Wang, High-order graph convolutional recurrent neural network: A deep learning framework for network-scale traffic learning and forecasting, 2018, CoRR, abs/1802.07007 URL http://arxiv.org/abs/1802.07007.

[52] P.W. Battaglia, J.B. Hamrick, V. Bapst, A. Sanchez-Gonzalez, V.F. Zambaldi, M. Malinowski, A. Tacchetti, D. Raposo, A. Santoro, R. Faulkner, Ç. Gülçehre, H.F. Song, A.J. Ballard, J. Gilmer, G.E. Dahl, A. Vaswani, K.R. Allen, C. Nash, V. Langston, C. Dyer, N. Heess, D. Wierstra, P. Kohli, M.M. Botvinick, O. Vinyals, Y. Li, R. Pascanu, Relational inductive biases, deep learning, and graph networks, 2018, CoRR, abs/1806.01261 URL http://arxiv.org/abs/1806.01261.

[53] P. Battaglia, R. Pascanu, M. Lai, D. Jimenez Rezende, k. kavukcuoglu, Interaction networks for learning about objects, relations and physics, in: D. Lee, M. Sugiyama, U. Luxburg, I. Guyon, R. Garnett (Eds.), Advances in Neural Information Processing Systems, Vol. 29, Curran Associates, Inc., 2016, URL https://proceedings.neurips.cc/paper/2016/file/3147da8ab4a0437c15ef51a5cc7f2dc4-Paper.pdf.

[54] V.G. Satorras, E. Hoogeboom, M. Welling, E(n) equivariant graph neural networks, 2021, URL https://arxiv.org/abs/2102.09844.

[55] C. Yang, W. Gao, D. Wu, C. Wang, Learning to simulate unseen physical systems with graph neural networks, 2022, URL https://arxiv.org/abs/2201.11976.

[56] D. Dalton, H. Gao, D. Husmeier, Emulation of cardiac mechanics using graph neural networks, Comput. Methods Appl. Mech. Engrg. (2022).

[57] M. Destrade, L. Dorfmann, G. Saccomandi, The ogden model of rubber mechanics: 50 years of impact on nonlinear elasticity, Phil. Trans. R. Soc. A 380 (2234) (2022) 20210332.

[58] J.M. Guccione, K.D. Costa, A.D. McCulloch, Finite element stress analysis of left ventricular mechanics in the beating dog heart, J. Biomech. 28 (10) (1995) 1167–1177.

[59] E.D.S. Neto, F.A. Pires, D. Owen, F-bar-based linear triangles and tetrahedra for finite strain analysis of nearly incompressible solids. Part I: formulation and benchmarking, Internat. J. Numer. Methods Engrg. 62 (3) (2005) 353–383.

[60] G.A. Holzapfel, Nonlinear Solid Mechanics: A Continuum Approach for Engineering Science, Kluwer Academic Publishers Dordrecht, 2002.

[61] F. e Avila Belbute-Peres, T. D. Economon, J.Z. Kolter, Combining differentiable PDE solvers and graph neural networks for fluid flow prediction, in: International Conference on Machine Learning, Vol. 37, 2020.

[62] A. Gruber, M. Gunzburger, L. Ju, Z. Wang, A comparison of neural network architectures for data-driven reduced-order modeling, Comput. Methods Appl. Mech. Engrg. (ISSN: 0045-7825) 393 (2022) 114764, http://dx.doi.org/10.1016/j.cma.2022.114764, URL https://www.sciencedirect.com/science/article/pii/S004578252200113X.

[63] H. Gao, M.J. Zahr, J.-X. Wang, Physics-informed graph neural Galerkin networks: A unified framework for solving PDE-governed forward and inverse problems, Comput. Methods Appl. Mech. Engrg. (ISSN: 0045-7825) 390 (2022) 114502, http://dx.doi.org/10.1016/j.cma.2021.114502, URL https://www.sciencedirect.com/science/article/pii/S0045782521007076.

[64] J.L. Ba, J.R. Kiros, G.E. Hinton, Layer normalization, 2016, arXiv preprint arXiv:1607.06450.

[65] C.M. Bishop, Training with noise is equivalent to tikhonov regularization, Neural Comput. (ISSN: 0899-7667) 7 (1) (1995) 108–116, http://dx.doi.org/10.1162/neco.1995.7.1.108.

[66] J.T. Barron, Continuously differentiable exponential linear units, 2017, URL https://arxiv.org/abs/1704.07483.

[67] D.P. Kingma, J. Ba, Adam: A method for stochastic optimization, 2017, arXiv:1412.6980.

[68] H.P. Langtangen, A. Logg, Solving PDEs in Python, Springer International Publishing, 2016, http://dx.doi.org/10.1007/978-3-319-52462-7, URL http://link.springer.com/10.1007/978-3-319-52462-7.

[69] J. Bradbury, R. Frostig, P. Hawkins, M.J. Johnson, C. Leary, D. Maclaurin, G. Necula, A. Paszke, J. VanderPlas, S. Wanderman-Milne, Q. Zhang, JAX: composable transformations of python+numpy programs, 2018, URL http://github.com/google/jax.

[70] J. Heek, A. Levskaya, A. Oliver, M. Ritter, B. Rondepierre, A. Steiner, M. van Zee, Flax: A neural network library and ecosystem for JAX, 2020, URL http://github.com/google/flax.

[71] I. Babuschkin, K. Baumli, A. Bell, S. Bhupatiraju, J. Bruce, P. Buchlovsky, D. Budden, T. Cai, A. Clark, I. Danihelka, C. Fantacci, J. Godwin, C. Jones, R. Hemsley, T. Hennigan, M. Hessel, S. Hou, S. Kapturowski, T. Keck, I. Kemaev, M. King, M. Kunesch, L. Martens, H. Merzic, V. Mikulik, T. Norman, J. Quan, G. Papamakarios, R. Ring, F. Ruiz, A. Sanchez, R. Schneider, E. Sezener, S. Spencer, S. Srinivasan, L. Wang, W. Stokowiec, F. Viola, The DeepMind JAX ecosystem, 2020, URL http://github.com/deepmind.

[72] V. Davies, U. Noè, A. Lazarus, H. Gao, B. Macdonald, C. Berry, X. Luo, D. Husmeier, Fast parameter inference in a biomechanical model of the left ventricle by using statistical emulation, J. R. Stat. Soc. Ser. C. Appl. Stat. (ISSN: 0035-9254) 68 (5) (2019) 1555–1576, http://dx.doi.org/10.1111/rssc.12374.

[73] J. Zhang, S. Chauhan, Fast computation of soft tissue thermal response under deformation based on fast explicit dynamics finite element algorithm for surgical simulation, Comput. Methods Programs Biomed. (ISSN: 0169-2607) 187 (2020) 105244, http://dx.doi.org/10.1016/j.cmpb.2019.105244, URL https://www.sciencedirect.com/science/article/pii/S0169260719311344.

[74] W. Kratzer, V. Fritz, R.A. Mason, M.M. Haenle, V. Kaechele, R.S. Group, Factors affecting liver size, J. Ultrasound Med. 22 (11) (2003) 1155–1161, http://dx.doi.org/10.7863/jum.2003.22.11.1155, URL https://onlinelibrary.wiley.com/doi/abs/10.7863/jum.2003.22.11.1155.

[75] W. Li, H. Gao, K. Mangion, C. Berry, X. Luo, Apparent growth tensor of left ventricular post myocardial infarction–In human first natural history study, Comput. Biol. Med. 129 (2020) 104168.

[76] H.M. Wang, H. Gao, X.Y. Luo, C. Berry, B.E. Griffith, R.W. Ogden, T.J. Wang, Structure-based finite strain modelling of the human left ventricle in diastole, Int. J. Numer. Methods Biomed. Eng. (ISSN: 2040-7947) 29 (1) (2013) 83–103, http://dx.doi.org/10.1002/cnm.2497.

[77] H. Liu, J.S. Soares, J. Walmsley, D.S. Li, S. Raut, R. Avazmohammadi, P. Iaizzo, M. Palmer, J.H. Gorman, R.C. Gorman, et al., The impact of myocardial compressibility on organ-level simulations of the normal and infarcted heart, Sci. Rep. 11 (1) (2021) 1–15.

[78] O. Bernard, A. Lalande, C. Zotti, F. Cervenansky, X. Yang, P.-A. Heng, I. Cetin, K. Lekadir, O. Camara, M.A.G. Ballester, et al., Deep learning techniques for automatic MRI cardiac multi-structures segmentation and diagnosis: is the problem solved? IEEE Trans. Med. Imaging 37 (11) (2018) 2514–2525.

[79] A. Rabbani, H. Gao, A. Lazarus, D. Dalton, Y. Ge, K. Mangion, C. Berry, D. Husmeier, Image-based estimation of the left ventricular cavity volume using deep learning and Gaussian process with cardio-mechanical applications, Comput. Med. Imaging Graph. (ISSN: 0895-6111) 106 (2023) 102203, http://dx.doi.org/10.1016/j.compmedimag.2023.102203, URL https://www.sciencedirect.com/science/article/pii/S0895611123000216.

[80] E. Weinan, B. Yu, The Deep Ritz Method: A deep learning-based numerical algorithm for solving variational problems, Commun. Math. Stat. 1 (6) (2018) 1–12.

[81] S.H. Sheen, E. Larionov, D.K. Pai, Volume preserving simulation of soft tissue with skin, Proc. ACM Comput. Graph. Interact. Tech. 4 (3) (2021) 1–23.

[82] P. Reiser, M. Neubert, A. Eberhard, L. Torresi, C. Zhou, C. Shao, H. Metni, C. van Hoesel, H. Schopmans, T. Sommer, et al., Graph neural networks for materials science and chemistry, Commun. Mater. 3 (1) (2022) 93.

[83] L. Pegolotti, M.R. Pfaller, N.L. Rubio, K. Ding, R.B. Brufau, E.F. Darve, A.L. Marsden, Learning reduced-order models for cardiovascular simulations with graph neural networks, 2023, ArXiv, abs/2303.07310.

[84] E. Haghighat, D. Amini, R. Juanes, Physics-informed neural network simulation of multiphase poroelasticity using stress-split sequential training, Comput. Methods Appl. Mech. Engrg. 397 (2022) 115141.

[85] J. He, L. Li, J. Xu, C. Zheng, ReLU deep neural networks and linear finite elements, J. Comput. Math. 38 (3) (2020) 502–527, http://dx.doi.org/10.4208/jcm.1901-m2018-0160, URL http://arxiv.org/abs/1807.03973 arXiv:1807.03973 [math].

[86] L.N. Smith, A disciplined approach to neural network hyper-parameters: Part 1 – learning rate, batch size, momentum, and weight decay, 2018, arXiv:1803.09820.