Proceedings of the 3rd International Conference on Statistics: Theory and Applications (ICSTA'21)
~~Prague, Czech Republic~~ Virtual Conference – August, 2021
Paper No. 127
DOI: 10.11159/icsta21.127

Graph Neural Network Emulation of Cardiac Mechanics

David Dalton[1], Alan Lazarus[1] Arash Rabbani[1], Hao Gao[1], Dirk Husmeier[1]
[1]School of Mathematics and Statistics, University of Glasgow,
Glasgow G12 8SQ, UK
d.dalton.1@research.gla.ac.uk; a.lazarus.1@research.gla.ac.uk; arash.rabbani@glasgow.ac.uk;
hao.gao@glasgow.ac.uk; dirk.husmeier@glasgow.ac.uk

Abstract - This paper compares the performance of two graph neural network architectures on the emulation of a cardiac mechanic model of the left ventricle of the heart. These models can be applied directly on the same computational mesh of the left ventricle geometry that is used by the expensive numerical forward solver, precluding the need for a low-order approximation of the true geometry. Our results show that these emulation approaches incur negligible loss in accuracy compared in the forward simulator, while making predictions multiple orders of magnitude more quickly, raising the prospect for their use in both forward and inverse problems in cardiac modelling.

***Keywords*:** Cardiac Mechanics, Holzapfel-Ogden Model, Emulation, Graph Neural Networks

1. Introduction

Mathematical modelling of the biomechanics of soft-tissue physiological processes has seen notable advances in recent years [1]. In particular, the increasing sophistication of mechanical models of the dynamics of the heart during the cardiac pump-cycle has raised the prospect of their use in-clinic for real-time decision support [2]. In this paper, we study one such model, the Holzapfel-Ogden (HO) constitutive law [3], which characterises the passive properties of the myocardium of the left ventricle (LV) of the heart. These properties determine the dynamics of the LV during diastole, the period of the cardiac cycle when the LV fills with blood. It has been shown that calibration of the LV myocardial passive properties predicted by the HO model against the corresponding properties observed in cardiac imaging can deliver insights into cardiac function that are of relevance to clinicians [4]. The problem with directly applying the cardiac biomechanical model along with the HO model in clinic is that the equations describing the passive dynamics of the heart admit no analytical solutions. Instead, the specific LV geometry of a given subject must be discretised into a computational mesh before the equations can be solved numerically. Using for example the finite element method, finding a numerical solution can take up to 10 minutes, even on a high-performance computer. In order for in-clinic iterative calibration of subject-specific myocardial passive properties however, this procedure would have to be performed at each calibration step. Since hundreds of iterations may be required for accurate calibration, the computational expense that would be incurred prohibits the use of this approach in real-time clinical decision support.

The problem of expensive numerical forward solvers is general to the physical sciences and has led to the development of the field of emulation, or surrogate modelling [5]. An emulator, commonly a Gaussian process, is a statistical model which approximates the numerical solver, or simulator, at much lower computational expense at prediction time. The emulator is trained on a set of numerical forward simulations, which are designed to densely fill the domain of the simulator. In cardiac mechanics, the domain of interest is the space of physiologically plausible myocardial material parameter values, blood pressure profiles and LV geometries, respectively. The issue here is that each LV geometry must be represented by a computational mesh comprised of hundreds or potentially thousands of nodes to obtain accurate numerical simulations - generating a training data set with a dense coverage over a space of this dimensionality is not feasible. For this reason, previous research in this area has used dimensionality reduction techniques to find a lower order representation of the LV geometry mesh, before then training an emulator in this reduced space [6,7]. The problem with this approach is that the low-order representation of the LV geometry mesh will not perfectly match the true geometry. Unfortunately, we have found that that it is difficult to get a good representation of the LV mesh that is of sufficiently low order where traditional emulation techniques can be used effectively.

There has recently been substantial interest in the machine learning community in the application of graph neural networks (GNNs) [8, Chapter 23] to the task of emulation of mesh-based simulation models. GNNs explicitly account for the local connectivity structure of the computational mesh, in a manner analogous to how a convolutional neural network accounts for the local connectivity of an image, allowing them to be applied to high density meshes. Our objective in this paper is to take two of the latest GNN architectures which have demonstrated strong performance on mesh-based emulation: `DeepEmulators` [9] and `MeshGraphNets` [10], and compare their performance on the emulation of passive cardiac mechanics. This is a challenging emulation problem, as each LV has a highly non-Euclidean geometry and is also non-isotropic due to the existence of muscle fibres in the myocardium. The convoluted geometry of these fibres around the LV perimeter as well as in different tissue layers, means the amount of strain and tensile forces within the generated meshes are highly variable in different locations and directions [11]. In Section 3 below, we outline how we created training and test data sets on which the two GNN emulators could be trained and evaluated, before the two architectures are explained in Section 4. The emulation results are then presented and discussed in Sections 5 and 6 respectively. Firstly however, we give a brief of the description of the biomechanical model of the left ventricle we consider here.

2. The Biomechanical Left Ventricle Model

In this study, we consider the passive dynamics of the left ventricle under a constant end-diastolic pressure of 8 mmHg. The myocardium is characterized by an incompressible, transversely isotropic fibre-reinforced material, a reduced form of the widely used Holzapfel and Ogden model [3], denoted as the HO model. The fibre field is considered as generated by the Laplace-Dirichlet Rule-Based (LDRB) algorithm [12]. The strain energy density function is given by

$$\mathbf{\Psi} = \frac{a}{2b}\left(e^{b(I_1-3)} - 1\right) + \frac{a_\mathrm{f}}{2b_\mathrm{f}}\left(e^{b_\mathrm{f}(I_{4\mathrm{f}}-1)^2} - 1\right) + \frac{1}{2}K(J-1)^2\,, \tag{1}$$

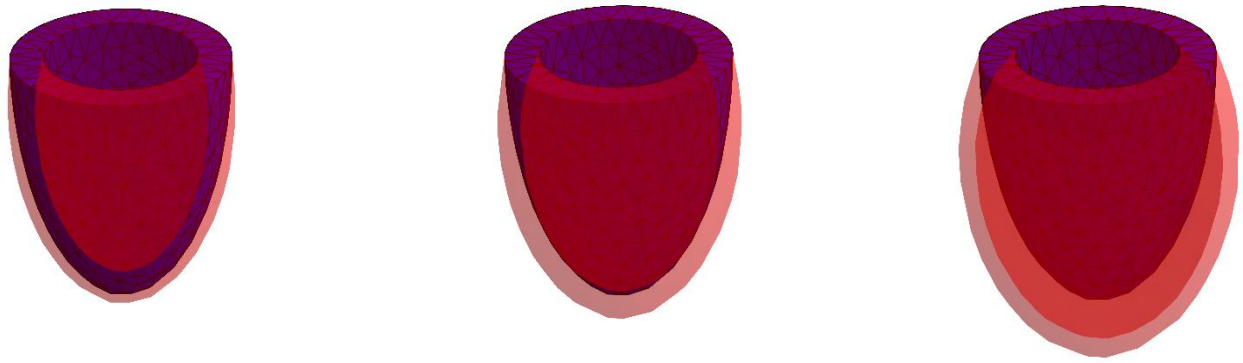
where a, b, a_f and b_f are the material parameters, which we denote collectively as $\boldsymbol{\theta}$. The parameters a and b characterise the isotropic response of the myocardium, while a_f and b_f describe the reinforced stiffness along the myofibres. The invariants I_1 and $I_{4\mathrm{f}}$ are defined as $I_1 = \mathrm{trace}\left(\mathbf{F}^\mathrm{T}\mathbf{F}\right)$ and $I_{4\mathrm{f}} = \mathrm{f}_0 \cdot \left(\mathbf{F}^\mathrm{T}\mathbf{F}\right)\mathrm{f}_0$, where $\mathbf{F}$ is the deformation gradient and f_0 is the unit myofiber direction at the reference configuration. The last term in (1) is to enforce the incompressibility constraint with the Lagrangian multiplier K and $J = \det(\mathbf{F})$. The quasi-static boundary value problem for the LV passive filling in the current configuration ($\boldsymbol{\Omega}$) is defined as

$$\begin{cases} \boldsymbol{\nabla}\cdot\boldsymbol{\sigma} + \mathbf{b} = 0 & \text{in } \boldsymbol{\Omega}, \\ \boldsymbol{\sigma}\cdot\mathbf{n} = \mathbf{t} & \text{in } \partial\boldsymbol{\Omega}^N, \\ \mathbf{u} = \bar{\mathbf{u}} & \text{in } \partial\boldsymbol{\Omega}^D, \end{cases} \tag{2}$$

where $\boldsymbol{\sigma}$ is derived from the HO model by differentiating $\mathbf{\Psi}$ with respect to F, $\mathbf{b}$ is the body force per unit volume, $\mathbf{n}$ is the normal direction of the external surface of the Neumann boundaries ($\partial\boldsymbol{\Omega}^N$) and $\mathbf{t} = -\boldsymbol{p}\boldsymbol{n}$ with $\boldsymbol{p}$ the applied cavity pressure. $\bar{\mathbf{u}}$ is the vector of displacement boundaries at the top of the LV geometry, the Dirichlet boundaries ($\partial\boldsymbol{\Omega}^D$). Eq. (2) must be solved numerically to find the displacement of the LV geometry from start to end-diastole, given a value of the material properties $\boldsymbol{\theta}$. We do so using the open-source library Pulse [13], which makes use of the finite element method (FEM) to solve for the displacement of a LV geometry that has been discretised into a finite set of nodes with a local connectivity structure. The reader is directed to [13] for further details.

3. Data

We denote the numerical forward solver described at the end of the previous section as $\boldsymbol{\mathcal{M}}$, which takes as input a material parameter vector $\boldsymbol{\theta}$ and a computational LV geometry mesh G, and returns a displacement matrix $\mathbf{U} \in \mathbb{R}^{M\times 3}$, where M is the number of finite-element nodes in G. That is, we have a function:



(a) $\boldsymbol{\theta}$ = [2.0,4.0,2.0,4.0] (b) $\boldsymbol{\theta}$ = [1.0,2.0,1.0,2.0] (c) $\boldsymbol{\theta}$ = [0.5,1.0,0.5,1.0]

Fig. 1: Undeformed LV geometry (blue) against displaced geometry (red) from the simulator, run for three different values of the material parameter vector $\boldsymbol{\theta}$. Lower myocardial stiffness levels lead to larger displacements from start to end-diastole.

$$\boldsymbol{\mathcal{M}}: \boldsymbol{\Theta} \times \mathcal{G} \rightarrow \mathbb{R}^{M \times 3} \tag{3}$$

which we seek to approximate with a surrogate model $\widehat{\boldsymbol{\mathcal{M}}}$, where $\boldsymbol{\Theta}$ and $\mathcal{G}$ are the spaces of physiologically plausible material parameter values and LV geometries, respectively. In order to build an emulator, we require a training data set of input-output pairs from $\boldsymbol{\mathcal{M}}$:

$$\boldsymbol{\mathcal{D}} = \{([\boldsymbol{\theta}_i, G_i], \mathbf{U}_i)_{i=1}^{N}\} \tag{4}$$

In addition, an independent test set is required to evaluate the accuracy of the surrogate. Creating these datasets in turn requires a selection of material parameter and LV geometry configurations from which to run simulations. Considering first the material parameters, we would ideally select their values for the simulations by sampling from the true population distribution, however this distribution is unknown. Another possibility would be to specify very wide feasible bounds on each individual parameter value and set $\boldsymbol{\Theta}$ to be a large box-domain encompassing these bounds. Then, a low-discrepancy sampling method such as a Sobol sequence [14] could be used to choose points in this space. This approach however would tend to favour running simulations from high material stiffness values. Such values are unlikely to be encountered in real patient data, as they would constrain the myocardium from expanding enough during diastole to pump sufficient blood into the systemic circulation through the aorta. This is illustrated in Figure 1, where for a fixed LV geometry, we run a simulation of the HO model for three different values of the material stiffness parameters $\boldsymbol{\theta}$. We see that, as the stiffness values *decrease*, the displacement in the geometry between start and end-diastole *increases*. For this reason, we instead decide to sample the material parameters using a Sobol sequence of length 1000 on *log* parameter space. The first 600 points were used for training, and the following 100 were used as a validation set on which the training of the emulators was monitored. The final 300 points were reserved as an independent test set. With this approach, our dataset is biased towards lower material parameter values, which will yield larger, more realistic displacements from the simulator. We chose the sample so that the material parameters a and a_{f} would be in the range $[0.1,10]$ kPa (the original space) and b and b_{f} would be in the range $[0.5,10]$. To create the LV geometry meshes, we used functionality in the Pulse library to generate a simplified artificial mesh with 543 finite element nodes. We then deformed the length/width/breadth this base mesh randomly to create 1000 meshes from which to run training, validation and test simulations. The ranges within which these random deformations were applied were chosen to ensure that the dimensions of the artificial geometries were in ranges that are plausible for real LV geometries. Note that this procedure ensured that each mesh was unique; in particular, the meshes run for the test simulations were not used to generate the training simulations. Nevertheless, the meshes generated by this procedure were more symmetric than is the case for real LV meshes. Using real LV geometry meshes will the remit of future work. Once the material parameter and LV geometry inputs were specified, we ran a simulation of the diastolic filling of the LV to obtain the resulting displacement for all 1,000 input configurations. Due to the bias in our material sampling procedure towards lower stiffness values, a large variety of magnitudes in LV displacements were observed. For example, the increases in LV cavity volume from start to end-diastole for the 300 test simulation points ranged from 15% to 500%.

4. Emulation Approaches

Our objective is to build an emulator $\widehat{\boldsymbol{\mathcal{M}}}(\boldsymbol{\theta}, G)$which approximates the true forward model, $\boldsymbol{\mathcal{M}}$. Before doing so, we first processed the computational LV meshes from our dataset into a format suitable for emulator training. Specifically, each finite element node i was assigned a feature vector $\mathbf{v}_i$ defined as follows:

$$\mathbf{v}_i = (b_i, \mathbf{f}_i, \log \boldsymbol{\theta}) \tag{5}$$

Where b_i is a Boolean variable indicating if the node lies on the Dirichlet boundary, $\mathbf{f}_i \in \mathbb{R}^3$is the fibre orientation vector at the node, and finally $\boldsymbol{\theta}$ is the four-dimensional material parameter vector, common to all nodes in each individual geometry. We take the log of the material parameters because the displacement response of the myocardium increases non-linearly as the stiffness values decrease, and we find that working on the log parameter scale increases emulator performance. We also create a directed edge to node i from each node j which is a member of a common finite element to node i. The set of all nodes j which send a directed edge to node i is called the *local neighbourhood* of node i, which we denote $\mathcal{N}_i$. Each directed edge is assigned the following feature vector:

$$\mathbf{e}_{ji} = \left(\mathbf{x}_i - \mathbf{x}_j, || \mathbf{x}_i - \mathbf{x}_j ||_2\right) \tag{6}$$

where $\mathbf{x}_i$ are the euclidean coordinates of the receiving node i and $\mathbf{x}_j$ are the coordinates of the sender node j. Note that *absolute* edge positions are not inputted to the model, precluding the need for the use of a low order mesh representation. Instead, *relative* node positions are encoded, meaning the emulator must learn to predict displacements based only on local attributes of the LV geometry. We then applied two state-of-the-art GNN architectures [9,10] to our processed data, each of which we describe now.

4.1. DeepEmulator

The `DeepEmulator` architecture [9] was inspired by the intuition that within short-time frames, the displacement of a given node i is primarily a function of its own inertia, and the internal forces from those nodes in its immediate local neigbourhood $\mathcal{N}_i$. It makes predictions for the displacement $\mathbf{u}_i$ at each node i as follows:

$$\begin{aligned}
\mathbf{z}_i^{\text{inertia}} &= f_\alpha^{\text{inertia}}(\mathbf{v}_i) \\
\mathbf{z}_{ji}^{\text{internalForce}} &= f_\beta^{\text{internalForce}}\left(\mathbf{e}_{ji}, \mathbf{v}_j, \mathbf{v}_i\right) \\
\mathbf{u}_i &= g_\gamma\left(\mathbf{z}_i^{\text{inertia}}, \textstyle\sum_{j \in \mathcal{N}_i} \mathbf{z}_{ji}^{\text{internalForce}}\right)
\end{aligned} \tag{7}$$

where $f_\alpha^{\text{inertia}}$, $f_\beta^{\text{internalForce}}$ and g_γ are each individual multi-layer perceptrons (MLPs). Note that the inputs to $f_\alpha^{\text{inertia}}$ and $f_\beta^{\text{internalForce}}$ described above differ slightly from those used in [9]. This is because the authors introduce the `DeepEmulator` architecture to predict secondary motions of 3D character animations, given deterministic primary motions, whereas we apply the method here in the context of cardiac mechanics. The MLP $f_\alpha^{\text{inertia}}$ focuses solely on the individual characteristics of the centre node i, and it embeds the raw values $\mathbf{v}_i$ into a higher dimensional latent vector $\mathbf{z}_i^{\text{inertia}}$. The weights and biases of this MLP are updated during the training procedure to find a latent representation which allows for accurate displacement predictions to be made. The MLP $f_\beta^{\text{internalForce}}$ on the other hand encodes the internal forces acting from each node $j \in \mathcal{N}_i$ on the center node i into a latent vector $\mathbf{z}_{ji}^{\text{internalForce}}$. The internal forces depend both on the characteristics $\mathbf{e}_{ji}$ of the directed edge from node j to node i, but also the individual characteristics $\mathbf{v}_j$ and $\mathbf{v}_i$ of the nodes themselves. Finally, the MLP g_γ predicts the displacement of each node i, given the inertia vector $\mathbf{z}_i^{\text{inertia}}$ and the sum of the internal force vectors $\mathbf{z}_{ji}^{\text{internalForce}}$ from all neighbouring nodes j. The idea here is that all geometrical information has been encoded into these latent vectors, allowing final predictions to be made for each node

individually. All three MLPs $f_\alpha^{\text{inertia}}$, $f_\beta^{\text{internalForce}}$ and g_γ were implemented with two hidden layers each of width 96, using the tanh activation function. In addition, the outputs of $f_\alpha^{\text{inertia}}$ and $f_\beta^{\text{internalForce}}$ were processed with a LayerNorm. The latent vectors $\mathbf{z}_i^{\text{inertia}}$ and $\mathbf{z}_{ji}^{\text{internalForce}}$ were chosen to be of dimensionality 32 and 64, respectively.

4.2. MeshGraphNets

`MeshGraphNets` is an encode-process-decode GNN architecture introduced in [10] for the emulation of mesh- based simulators. The authors were interested in simulations over many time steps and with simulations involving the interaction between different bodies. In our case, we are interested in the simulation of the displacement of a single LV geometry at one specific time point, end-diastole, and we have tailored our discussion of the architecture to this problem domain. Given a computational mesh whose vertices and edges have been assigned feature vectors $\mathbf{v}_i$ and $\mathbf{e}_{ij}$ respectively, `MeshGraphNets` first encodes these features into higher dimensional latent vectors $\mathbf{v}_i^0$ and $\mathbf{e}_{ij}^0$. The encodings are performed using MLPs denoted ϵ^V and ϵ^E. These latent vectors are then sequentially updated in the processor stage for $l = 1,2,..L$ message-passing steps as follows:

$$\begin{aligned} \mathbf{e}_{ij}^l &= f^E\left(\mathbf{e}_{ij}^{l-1}, \mathbf{v}_i^{l-1}, \mathbf{v}_j^{l-1}\right) \\ \mathbf{v}_i^l &= f^V\left(\mathbf{v}_i^{l-1}, \textstyle\sum_{j\in\mathcal{N}_i} \mathbf{e}_{ji}^l\right) \end{aligned} \tag{8}$$

where f^V and f^E are MLPs with a residual connection. The final decode step predicts the displacement $\mathbf{u}_i$ for each node i using the final latent node representation $\mathbf{v}_i^L$ using the prediction MLP δ^V. For this work, all MLPs ϵ^V, ϵ^E, f^V, f^E and δ^V were implemented with two hidden layers of with 96 with tanh activation function, as in 4.1 above. The outputs of all MLPs apart from δ^V were processed with a LayerNorm. The latent embeddings for the node and edge features were each of length 24. Finally, we used $L = 3$ message passing steps.

4.3. Architecture Comparison

The primary difference between the two architectures is the message passing steps used in `MeshGraphNets`, which means that information from nodes outside the local neighbourhood $\mathcal{N}_i$ can be used to predict the displacement at node i. This also means however that the network will require more parameters, and hence incurs longer training and prediction times. Another difference is the form of the final prediction MLP for each architecture. The prediction MLP for the `DeepEmulator` is more physics-informed in the sense that it tries to model the inertia and internal forces acting on each node from each of its neighbours, whereas the `MeshGraphNets` prediction MLP only uses information from the final node representation to make predictions.

4.4. Model Training and Evaluation

We trained both networks using a per-node mean-squared-error loss between the true displacements $\mathbf{u}_i$ and those predicted by the model, $\widehat{\mathbf{u}}_i$. All input features and output displacements were normalised to mean zero and unit variance before training. Training was carried out for 5,000 epochs using the Adam algorithm with a mini-batch size of 10 and a fixed learning rate of 5×10^{-4}. Each network was implemented in Python using the graph_nets library [16] and training was performed using Tensorflow on an NVIDIA Quadro P4000 GPU. Note that for each architecture, our choice of latent hyper-parameters such as MLP width and latent dimension sizes are smaller than those used in the original publications. This is because we saw negligible difference in prediction accuracy when we increased the size of the latent values beyond those outlined above, which is possibly due to our dataset being significantly smaller than those used in [9] and [10].

5. Results

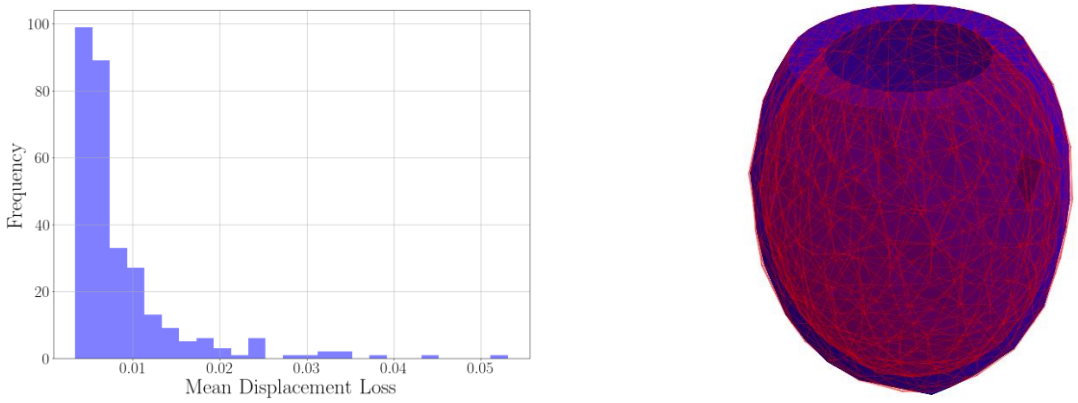
Having trained the two GNN emulation approaches as described above, we used the fitted models to predict the displacement of the 300 independent test simulations. For each simulation $i = 1,2,\ldots 300$ and finite element point $j = 1,2,\ldots 543$, we calculated the Euclidean distance $||\ \mathbf{u}_{ij} - \widehat{\mathbf{u}}_{ij}\ ||_2$ between the true simulated displacement $\mathbf{u}_{ij}$ and the corresponding prediction from the emulator $\widehat{\mathbf{u}}_{ij}$, giving a total of $300 \times 543 = 162{,}900$ loss values for each emulator. The

empirical 25th, 50th (median) and 75th percentiles of the losses are displayed in Table 1. These summary statistics show the `MeshGraphNets` emulator outperforms the `DeepEmulator` by greater than a factor of two, with only 25% of its nodal displacement predictions incurring a loss of more than one-tenth of a millimetre.

Table 1: Test set node-wise prediction loss summary statistics (cm)

Emulation Method	**25th Percentile**	**Median**	**75th Percentile**
`DeepEmulator`	8.0×10^{-3}	1.4×10^{-2}	2.2×10^{-2}
`MeshGraphNets`	3.6×10^{-3}	6.2×10^{-3}	1.0×10^{-2}

The For each test geometry i, we then calculated the mean displacement loss over all finite element nodes: $\frac{1}{M}\sum_{j=1}^{M} || \mathbf{u}_{ij} - \hat{\mathbf{u}}_{ij} ||_2$ when using the `MeshGraphNets` emulator. A histogram of these per-geometry loss values is displayed in Figure 3 (a), which illustrates that there was a long tail in the distribution of test-set losses, with one outlier test geometry having a mean displacement loss of 5.2×10^{-2}, more than one order of magnitude larger than the test points with lowest loss. Figure 3 (b) plots the true simulated LV geometry for this outlier point in red mesh, against the `MeshGraphNets` prediction in blue.



(a) (b)

Fig. 2: (a) Histogram of mean displacement loss (cm) for the 300 test simulations. (b) Plot of true displaced geometry (red mesh) against the corresponding prediction from the `MeshGraphNets` emulator (blue) for the outlier point in (a) with mean displacement error of 5.2×10−2. The displacement of this geometry was simulated with a material parameter vector $\boldsymbol{\theta}$ = [0.11,0.6,0.16,8.16].

Finally, Table 2 compares the wall-clock prediction time of the `MeshGraphNets` emulator, shown in the first column, with the prediction times of the simulator. Simulation times are highly dependent on the values of the material stiffness parameters, with lower stiffness values leading to significantly longer prediction times. For this reason, two simulation times are shown for the simulator in Table 2. The second column of the table shows the time to convergence of a forward simulation with material parameter values $a = a_{\mathrm{f}} = 2.2$ and $b = b_{\mathrm{f}} = 3.2$ which are approximately the average values from the training data, while the third column then shows the time to convergence of a simulation with $a = 0.2$ and $b = 0.6$, with a_{f} and b_{f} fixed as before. The prediction times for the emulator are independent of the material parameter values so only one time is shown for the surrogate model. This time was found as the average of an ensemble of 100 predictions from the surrogate run on an NVIDIA Quadro P4000 GPU, whereas the numerical forward simulations were run on an Intel Xeon Platinum 8276L CPU. The final two columns of Table 2 show the speedup factor on the prediction times for the emulator compared to the simulator, which is in excess of three orders of magnitude for the higher material parameter stiffness configuration, and over four orders of magnitude for the lower stiffness configuration.

Table 2: Prediction times (seconds) for emulator ($\mathbf{t}_{\texttt{MeshGraphNet}}$) and true forward model ($\mathbf{t}_{\text{simulator}}$).

$\mathbf{t}_{\texttt{MeshGraphNet}}$	$\mathbf{t}_{\text{simulator}}$ $a = 2.2, b = 3.2$	$\mathbf{t}_{\text{simulator}}$ $a = 0.2, b = 0.6$	**Speedup** $a = 2.2, b = 3.2$	**Speedup** $a = 0.2, b = 0.6$
1.0×10^{-2}	13.6	551	1.4×10^{3}	5.5×10^{4}

6. Discussion

The results presented in Table 1 demonstrate the ability of both the `DeepEmulator` and `MeshGraphNets` models to emulate the HO law with a high degree of accuracy, with the `MeshGraphNets` approach outperforming `DeepEmulator` by approximately a factor of two. This suggests that geometrical information outside the immediate neighbourhood of a node is useful for predicting the resulting nodal displacement. The histogram of per-geometry mean displacement errors in Figure 3 (b) illustrates that the summary statistics in Table 1 hide a long tail in the distribution of prediction errors for the `MeshGraphNets` emulator. The points in the tail of the histogram tend to correspond to simulations with low values of the material parameter values, even though our design of training inputs heavily favoured these regions of low stiffness. This suggests that a model-based design, such as those discussed in [5], rather than a space-filling one may be required when selecting the location of the training inputs to ensure that accurate predictions are made across the entire material parameter space. Figure 3 (b) displays in blue the ground truth displaced geometry against in red mesh the prediction of the `MeshGraphNets` emulator for the outlier simulation seen in the tail of Figure 3 (a), which had a mean displacement error of 5.2×10^{-2} between ground truth and prediction. Even for this outlier point, the difference between the true simulation and the prediction of the emulator appears qualitatively slight. While the `MeshGraphNets` emulator incurs a small loss in accuracy compared to the true forward simulator, it is significantly more computationally efficient at prediction time, as can be seen in Table 2, especially for simulations with low material stiffness values. Note that we have not optimised our implementation of `MeshGraphNets` for prediction speed and so there is potential for further performance gains here. The ability of the `MeshGraphNets` approach to make both accurate and fast predictions that generalise across different LV geometries means that it could be useful both for forward problems, such as sensitivity analyses, and inverse problems, where the objective is to estimate the material parameters $\boldsymbol{\theta}$. Before this approach is applied to such areas, however, it is important that we evaluate its emulation performance on real LV geometries, rather than the artificial geometries considered here. Real LV geometries can be highly asymmetrical with varying wall thickness, which will make emulation more challenging. Specifically, a larger data set and greater number of message-passing steps than those used in this study may be required to train a `MeshGraphNets` emulator that can account for the complexities of real LV geometries. In addition, it is important for both forward and inverse problems that the model can accurately estimate strains internal to the LV wall. This will require a much denser mesh than considered in this study, potentially with tens of thousands of finite element nodes. Both the `DeepEmulator` and `MeshGraphNets` architectures have been shown however to obtain strong performance on meshes of this size; the reader is directed to the discussions in [9] and [10] for details.

7. Conclusion

This paper has compared the performance of two graph neural network architectures: DeepEmulator [9] and MeshGraphNets [10], on emulation of the Holzapfel-Ogden model for the passive dynamics of the left ventricle. We found that both architectures obtained high accuracy, with MeshGraphNets outperforming DeepEmulator. In addition, we have shown that the MeshGraphNets emulator makes predictions multiple orders of magnitude more quickly than the numerical forward simulator. However, our study only considered simplified, artificial left ventricle geometries. Future work will apply these methods to real left ventricle geometries, which are extracted from cardiac imaging scans.

Acknowledgements

This work was carried out as part of the SofTMech project, funded by EPSRC, grant reference numbers EP/S030875/1 and EP/T017899/1.

References

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

[2] K. Mangion, H. Gao, D. Husmeier, X. Luo, and C. Berry, "Advances in computational modelling for personalised medicine after myocardial infarction," *Heart*, vol. 104, no. 7, pp. 550–557, 2018.

[3] G. A. Holzapfel and R. W. Ogden, "Constitutive modelling of passive myocardium: a structurally based framework for material characterization," *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences*, vol. 367, no. 1902, pp. 3445–3475, 2009

[4] H. Gao, A. Aderhold, K. Mangion, X. Luo, D. Husmeier, and C. Berry, "Changes and classification in myocardial contractile function in the left ventricle following acute myocardial infarction," *Journal of The Royal Society Interface*, vol. 14, no. 132, p. 20170203, 2017.

[5] R. B. Gramacy, *Surrogates: Gaussian process modeling, design, and optimization for the applied sciences.* Florida, USA: Chapman Hall/CRC, 2020.

[6] D. Dalton, A. Lazarus, and D. Husmeier, "Comparative evaluation of different emulators for cardiac mechanics," in *Proceedings of the 2nd International Conference on Statistics: Theory and Applications*, 2020.

[7] G. D. Maso Talou, T. P. Babarenda Gamage, M. Sagar, and M. P. Nash, "Deep learning over reduced intrinsic domains for efficient mechanics of the left ventricle," *Frontiers in Physics*, vol. 8, p. 30, 2020.

[8] K. P. Murphy, *Probabilistic Machine Learning: An introduction.* MIT Press, 2021.

[9] D. C. Mianlun Zheng, Yi Zhou and J. Barbic, "A deep emulator for secondary motion of 3d characters," *arXiv preprint* arXiv:2103.01261, 2021.

[10] A. S.-G. Tobias Pfaff, Meire Fortunato and P. W. Battaglia, "Learning mesh-based simulation with graph networks," *in Proceedings of the International Conference on Learning Representations*, 2021.

[11] D. S. Li, R. Avazmohammadi, S. S. Merchant, T. Kawamura, E. W. Hsu, J. H. Gorman III, R. C. Gorman, and M. S.Sacks, "Insights into the passive mechanical behavior of left ventricular myocardium using a robust constitutive model based on full 3d kinematics," *Journal of the mechanical behavior of biomedical materials*, vol. 103, p. 103508, 2020

[12] J. Bayer, R. Blake, G. Plank, and N. Trayanova, "A novel rule-based algorithm for assigning myocardial fiber orientation to computational heart models."*Annals of biomedical engineering*, vol. 40,10, 2012.

[13] H. Finsberg, "pulse: A python package based on fenics for solving problems in cardiac mechanics," *The Journal of Open Source Software*, vol. 4, no. 41, p. 1539, 2019.

[14] K.T. Fang, R. Li, and A. Sudjianto, "*Design and modeling for computer experiments*," Chapman & Hall/CRC, 2005.

[15] DeepMind, "Graph nets library," 2018. [Online]. Available: https://github.com/deepmind/graph_nets