Gaussian Process Emulation to Accelerate Parameter Estimation in a Mechanical Model of the Left Ventricle: a Critical Step towards Clinical End-user Relevance

Umberto Noè[1,*], Alan Lazarus[2,*], Hao Gao[2], Vinny Davies[2,3], Benn Macdonald[2], Kenneth Mangion[4,5], Colin Berry[4,5], Xiaoyu Luo[2], and Dirk Husmeier[2,c]

[1]German Center for Neurodegenerative Diseases (DZNE), Bonn, Germany.
[2]School of Mathematics and Statistics, University of Glasgow, Glasgow, UK.
[3]School of Computing Science, University of Glasgow, Glasgow, UK.
[4]BHF Glasgow Cardiovascular Research Centre, University of Glasgow, Glasgow, UK.
[5]West of Scotland Heart and Lung Centre, Golden Jubilee National Hospital, Clydebank, UK.
[*]Authors contributed equally.
[c]Corresponding author: dirk.husmeier@glasgow.ac.uk

Abstract

In recent years we have witnessed substantial advances in the mathematical modelling of the biomechanical processes underlying the dynamics of the cardiac soft-tissue. Gao et al. (2017) demonstrated that the parameters underlying the biomechanical model have diagnostic value for prognosticating the risk of myocardial infarction. However, the computational costs of parameter estimation are prohibitive when the goal lies in building real-time clinical decision support systems. This is due to the need to repeatedly solve the mathematical equations numerically using finite element discretization during an iterative optimization routine. The present article presents a method for accelerating the inference of the constitutive parameters by using statistical emulation with Gaussian processes. We demonstrate how the computational costs can be reduced by about three orders of magnitude, with hardly any loss in accuracy, and we assess various alternative techniques in a comparative evaluation study based on simulated data obtained by solving the left-ventricular model with the finite element method, and real MRI data for a human volunteer.

Keywords Left ventricular mechanics, Holzapfel-Ogden strain energy function, constitutive parameters, finite element discretization, simulation, emulation, Gaussian processes, parameter estimation, optimization.

1 Introduction

Mathematical modelling in cardiac physiology, reviewed for instance in Loret and Simoes (2016), is a topical research area that promises to substantially advance early diagnosis of ventricular dysfunction and risk of myocardial infarction. In recent years, there have been substantial advances in the development of realistic multiscale mathematical models, linking the properties of individual cells and fibres to the soft-tissue mechanical processes in the heart; see e.g. Holzapfel and Ogden (2018) for details. However, a considerable challenge is to infer the biophysical parameters that determine the mechanical properties of the tissues and fibres non-invasively from magnetic resonance images (MRI). In principle, this is achieved by comparing strains extracted from the MRI scans with those predicted from the mathematical model, quantifying the mismatch with an objective function, and applying multivariate optimization algorithms to find the parameters that minimize this function. In a recent proof-of-concept study (Gao et al., 2017) based on the constitutive model of passive myocardium proposed in Holzapfel and Ogden (2009), we successfully applied this approach to a population of 11 patients suffering from myocardial infarction (MI, commonly known as heart attack) and 27 healthy controls. Building a Gaussian process classifier in a six-dimensional biophysical parameter space, we achieved an out-of-sample sensitivity of 75% and specificity of 95% (Gao et al., 2017). The results demonstrate the diagnostic value of these parameters for clinical decision making. Unfortunately, the method does not immediately lead to a decision support tool for the clinical practice. The reason is that the soft-tissue mechanical equations have no closed-form solution and require a numerical procedure based on finite-element discretization (Rao, 2018), typically using programs like ABAQUS or Ansys, which even on a high-performance parallel computer cluster takes several minutes of CPU time. This procedure has to be repeated hundreds or thousands of times during the numerical optimization of the objective func-

tion, leading to computer run times of several days or weeks.

To deal with the high computational complexity and make progress towards a clinical decision support system that can make predictions in real time, recent research efforts have focused on statistical emulation (e.g. Kennedy and O'Hagan (2001); Conti et al. (2009); Conti and O'Hagan (2010)), which has recently been explored in the closely related contexts of cardiovascular fluid dynamics (Melis et al., 2017), the pulmonary circulatory system (Noè et al., 2017) and ventricular mechanics (Achille et al., 2018). The idea is to approximate the computationally expensive mathematical model (the *simulator*) with a computationally cheap statistical surrogate model (the *emulator*) by a combination of massive parallelization and nonlinear regression, so as to exploit computational resources before the patient arrives at the clinic. Starting from a space-filling design in parameter space, the underlying partial differential equations are solved numerically with finite element discretization on a parallel computer cluster, and methods from nonparametric Bayesian statistics based on Gaussian processes (Rasmussen and Williams, 2006) are applied for multivariate smooth interpolation. When new data become available, e.g. in the form of MRI scans, the resulting proxy objective function can be minimized at low computational costs, without the need for any further computationally expensive simulations from the original mathematical model.

The present article follows up on a recent proof-of-concept study that explored the application of univariate-output Gaussian processes (GPs) in the context of emulating left-ventricular cardiac dynamics (Davies et al., 2019). We extend this work with a more realistic emulator that allows for spatial correlations between strains at different positions in the LV wall, using multivariate output GPs, an extended comparative analysis with different emulations strategies, and a quantification of the efficiency versus accuracy trade-off in comparison with standard numerical procedures for the original mathematical model. Our article is structured as follows. Section 2 provides an overview of the biomechanical model of the left ventricle (LV) dynamics. Section 3 gives an overview of the statistical methodology used, focusing on a comparison between the concepts of *simulation* and *emulation*. Section 4 compares different emulation strategies. Section 5 describes how we have applied the different emulation frameworks to the inference of the constitutive parameters of the LV. Section 6 describes the data used in our study. The results of our comparative performance assessment and the application to cine MRI data is presented in Section 7. Section 8 concludes with a discussion and an outlook on future work.

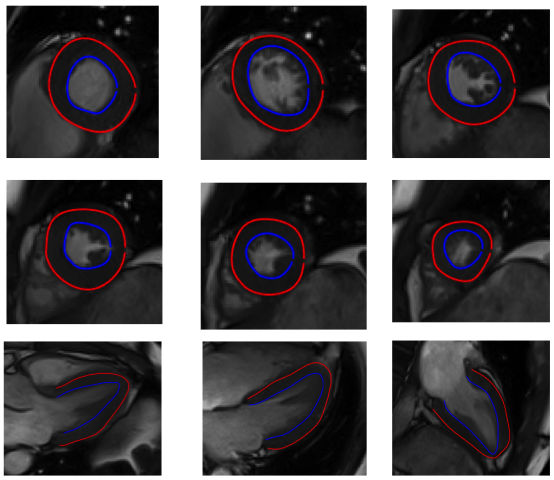


Figure 1: LV wall boundary segmentation from in vivo MR images of a healthy volunteer at early-diastole when the MV just opens. Red: epicardial boundary, blue: endocardial boundary. The first two rows are short-axis cine images from the base to apex, and figures in the last row are long-axis cine images.

2 Biomechanical Model of Left Ventricular Dynamics

The starting point of biomechanically modelling LV dynamics is the reconstruction of the LV geometry at early-diastole, as shown in Figures 1–2 and described in more detail in Section 6.2. Experimental studies have demonstrated that myocytes organize in a highly layered architecture, rotating continuously from endocardial to epicardial surfaces (Figure 2(b)). To describe the myofibre architecture, a local material coordinate system fibre ($\mathbf{f}$)-sheet ($\mathbf{s}$)-normal ($\mathbf{n}$) (LeGrice et al., 1995) is defined as shown in Figure 2(c). In brief, myofibres ($\mathbf{f}$) are assumed to rotate from -60° in the endocardial surface to 60° in the epicardial surface, shown in Figure 2(b). These myocytes (usually 4–6) form a sheet plane, which is orthogonal to $\mathbf{f}$, a unit vector within the sheet plane is then defined, $\mathbf{s}$, for the sheet direction and rotates from -45° to 45° from endocardium to epicardium. Accordingly, $\mathbf{n}$ is the sheet normal. A rule-based fibre-generation method is used to define the **f-s-n** system. Refer to Gao et al. (2014b) for more details on LV model construction from in vivo MR imaging.

Constitutive modelling of passive myocardium has progressed from isotropic linear material to nonlinear and fibre-reinforced laws by considering the intrinsic structural information. In this study, we use the incompressible invariant-based constitutive law

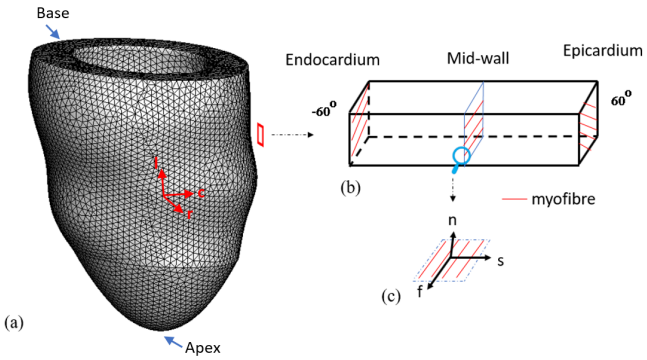


Figure 2: The biomechanical LV model reconstructed from in vivo MR images at early-diastole from a healthy volunteer. (a) the LV mesh with 25,933 nodes and 133,042 tetrahedron elements, **r** is the transmural direction, **l** is the longitudinal direction from the apex towards the centre of the LV base, and **c** is the circumferential direction; (b) a schematic illustration of myofibre (myocyte) rotation from endocardium to epicardium; (c) a local $\mathbf{f}-\mathbf{s}-\mathbf{n}$ material coordinate system.

(Holzapfel and Ogden, 2009), namely the HO law,

$$\begin{aligned}\Psi &= \frac{a}{2b}\{\exp[b(I_1-3)]-1\}\\ &+\sum_{i\in\{\mathrm{f,s}\}}\frac{a_i}{2b_i}\{\exp[b_i(I_{4i}-1)^2]-1\}\\ &+\frac{a_{\mathrm{fs}}}{2b_{\mathrm{fs}}}[\exp(b_{\mathrm{fs}}I_{8\,\mathrm{fs}}^2)-1],\end{aligned}\tag{1}$$

in which $\boldsymbol{q}=(a,b,a_{\mathrm{f}},b_{\mathrm{f}},a_{\mathrm{s}},b_{\mathrm{s}},a_{\mathrm{fs}},b_{\mathrm{fs}})$ are unknown material parameters, I_1, I_{4i}, and $I_{8\mathrm{fs}}$ are the invariants,

$$\begin{aligned}I_1&=\mathrm{trace}(\mathbf{C}), & I_{4\mathrm{f}}&=\mathbf{f}_0\cdot(\mathbf{C}\mathbf{f}_0),\\ I_{4\mathrm{s}}&=\mathbf{s}_0\cdot(\mathbf{C}\mathbf{s}_0), & I_{8\mathrm{fs}}&=\mathbf{f}_0\cdot(\mathbf{C}\mathbf{s}_0),\end{aligned}\tag{2}$$

in which $\mathbf{f}_0$ and $\mathbf{s}_0$ are the myofibre and sheet orientations in the reference configuration, which are known before the simulations. $\mathbf{C}=\mathbf{F}^\top\mathbf{F}$, and $\mathbf{F}$ is the deformation gradient

$$\mathbf{F}=\mathbf{I}+\frac{\partial\mathbf{u}}{\partial\mathbf{X}},\tag{3}$$

where $\mathbf{I}$ is the identity matrix, $\mathbf{u}$ is the displacement vector, and $\mathbf{X}$ is the position in the reference configuration.

We further decompose the deformation gradient $\mathbf{F}$ into volumetric ($\mathbf{F}_{\mathrm{vol}}$) and isochoric ($\bar{\mathbf{F}}$) parts, that is

$$\mathbf{F}=\bar{\mathbf{F}}\mathbf{F}_{\mathrm{vol}},\quad \mathbf{F}_{\mathrm{vol}}=J^{\frac{1}{3}}\mathbf{I},\quad \bar{\mathbf{F}}=J^{-\frac{1}{3}}\mathbf{F}$$

where $J=\det(\mathbf{F})$, and the modified right Cauchy-Green tensor is

$$\bar{\mathbf{C}}=J^{-\frac{2}{3}}\mathbf{C}.$$

In the same way, the modified strain invariants (shown with over bar) now are defined using $\bar{\mathbf{C}}$ instead of $\mathbf{C}$. Thus the strain energy function in (1) can be rewritten in terms of modified strain invariants in the form

$$\begin{aligned}\Psi &= \frac{a}{2b}\{\exp[b(\bar{I}_1-3)]-1\}\\ &+\sum_{i\in\{\mathrm{f,s}\}}\frac{a_i}{2b_i}\{\exp[b_i(\bar{I}_{4i}-1)^2]-1\}\\ &+\frac{a_{\mathrm{fs}}}{2b_{\mathrm{fs}}}[\exp(b_{\mathrm{fs}}\bar{I}_{8\,\mathrm{fs}}^2)-1]+\frac{1}{2}K(J-1)^2\end{aligned}\tag{4}$$

where the term $\frac{1}{2}K(J-1)^2$ accounts for the incompressibility of the material, and K is a constant bulk modulus (10^6 Pa). The second Piola-Kirchoff stress tensor ($\mathbf{S}$) can be derived from Equation (4) as

$$\begin{aligned}\mathbf{S}&=2\frac{\partial\Psi}{\partial\mathbf{C}}=K(J-1)J\mathbf{C}^{-1}\\ &+a\exp[b(\bar{I}_1-3)]\,(J^{-\frac{2}{3}}\mathbf{I}-\frac{1}{3}\bar{I}_1\mathbf{C}^{-1})\\ &+2a_{\mathrm{f}}(\bar{I}_{4\mathrm{f}}-1)\,\exp[b_{\mathrm{f}}(\bar{I}_{4\mathrm{f}}-1)^2]\,(\bar{\mathbf{f}}_0\otimes\bar{\mathbf{f}}_0-\frac{1}{3}\bar{I}_{4\mathrm{f}}\mathbf{C}^{-1})\\ &+2a_{\mathrm{s}}(\bar{I}_{4\mathrm{s}}-1)\,\exp[b_{\mathrm{s}}(\bar{I}_{4\mathrm{s}}-1)^2]\,(\bar{\mathbf{s}}_0\otimes\bar{\mathbf{s}}_0-\frac{1}{3}\bar{I}_{4\mathrm{s}}\mathbf{C}^{-1})\\ &+a_{\mathrm{fs}}\bar{I}_{8\mathrm{fs}}\exp[b_{\mathrm{fs}}\bar{I}_{8\mathrm{fs}}^2]\,(\bar{\mathbf{s}}_0\otimes\bar{\mathbf{f}}_0+\bar{\mathbf{f}}_0\otimes\bar{\mathbf{s}}_0-\frac{1}{3}\bar{I}_{8\mathrm{fs}}\mathbf{C}^{-1}),\end{aligned}\tag{5}$$

and the Cauchy stress is

$$\boldsymbol{\sigma}=\frac{1}{J}\mathbf{F}\mathbf{S}\mathbf{F}^\top.$$

The boundary value problem for the LV dynamics to be solved in the current configuration is

$$\begin{cases}\nabla\cdot\boldsymbol{\sigma}+\mathbf{b}=0 & \text{in } \Omega\\ \boldsymbol{\sigma}\cdot\mathbf{n}=\mathbf{t} & \text{in } \Gamma^N\\ \mathbf{u}=\mathbf{u}_0 & \text{in } \Gamma^D\end{cases}\tag{6}$$

where $\mathbf{b}$ is the body force density per unit volume, $\mathbf{n}$ is the normal direction of $\partial\Omega$, $\mathbf{t}$ is the traction force, Γ^N and Γ^D are the Neumann and Dirichlet boundaries. This problem is solved using the finite element approach implemented in a general-purpose finite elements package ABAQUS[1], which is used to simulate the biomechanical LV models during diastolic filling. A linearly ramped end-diastolic pressure is applied to the endocardial surface. A simulation (133,042 tetrahedron elements and 25,933 nodes) without using parallelization takes about 18 minutes on our local Linux workstation[2], and around 4.5 minutes if parallelized with 6 CPUs.

Gao et al. (2015) found that the 8 unknown parameters in the strain-stress relationship (1) are strongly correlated, and it can be very challenging to uniquely determine them from limited and noisy in vivo measurements. However, despite the non-uniqueness of

[1]Simulia, Providence, RI, USA
[2]Intel(R) Xeon(R) CPU, 2.9GHz, 32G memory

the HO parameters, they found that the strain-stress relation in the myofibre direction can be reliably estimated based on limited in vivo data (Gao et al., 2015). Hadjicharalambous et al. (2016) estimated passive myocardial stiffness in healthy subjects and in patients with non-ischemic dilated cardiomyopathy by using a reduced form of the HO law (1 unknown parameter instead of 8) in order to strike a balance between the model fidelity and unique parametrization. Following our previous study (Gao et al., 2015), we have grouped the eight parameters of (1) into four, so that:

$$\begin{aligned} a &= q_1\, a_0 & b &= q_1\, b_0 \\ a_{\mathrm{f}} &= q_2\, a_{\mathrm{f0}} & a_{\mathrm{s}} &= q_2\, a_{\mathrm{s0}} \\ b_{\mathrm{f}} &= q_3\, b_{\mathrm{f0}} & b_{\mathrm{s}} &= q_3\, b_{\mathrm{s0}} \\ a_{\mathrm{fs}} &= q_4\, a_{\mathrm{fs0}} & b_{\mathrm{fs}} &= q_4\, b_{\mathrm{fs0}} \end{aligned} \tag{7}$$

where $\boldsymbol{q} = (q_1, \ldots, q_4) \in [0.1, 5]^4$ are now the four parameters to be inferred from in vivo data, while a_0, b_0, a_{f0}, a_{s0}, b_{f0}, b_{s0}, a_{fs0}, b_{fs0} are reference values from the published literature (Gao et al., 2017)[3].

3 Statistical Methodology

The key interest of our study is the estimation of the biomechanical parameters $\boldsymbol{q}$, introduced in the previous section, non-invasively from cine MR images. This is motivated by a previous study that has established their diagnostic power for prognostication of the risk of myocardial infarction (Gao et al., 2017). The estimation of $\boldsymbol{q}$ follows the procedure described in Gao et al. (2015), which is based on applying an iterative optimization algorithm to find the parameter values that maximize the agreement (or minimize the mismatch) between patterns extracted from the cine MR images and the corresponding predictions from the mathematical model.

The patterns extracted from the cine MR images are peak circumferential strains; see Section 4 of the supplementary material for a discussion of their advantage over longitudinal and radial strains. The peak circumferential strains were measured at 24 positions on the LV wall corresponding to the myocardial segments in the short-axis cine images, combined with the LV chamber volume taken at end of diastole; see Section 6.2 and Gao et al. (2017) for details. The same features can be obtained from the soft-tissue mechanical model described in the previous section. Following the dynamics defined in (6), displacements with respect to the initial geometry are extracted, then the deformation gradient $\mathbf{F}$ at each location is calculated according to (3). From this the circumferential strains are obtained via

$$E_c = \mathbf{c} \cdot [\frac{1}{2}(\mathbf{F}^\top \mathbf{F} - \mathbf{I})\, \mathbf{c}] \tag{8}$$

in which $\mathbf{c}$ is the circumferential direction at each point in the LV wall. This vector, $\mathbf{c}$, is defined as the cross product of the transmural direction and the long-axis direction from the centre of the LV base plane to the apex shown in Figure 2. Finally peak circumferential strains are averaged within each segment within the LV wall corresponding to the segments from the short-axis cine images, giving a 24-dimensional vector. We add to this the LV cavity volume, which is calculated by triangulating the deformed endocardial surface first and then summing over the tetrahedron volume elements. To transform the LV volume and the circumferential strains onto the same scale, they are non-dimensionalized[4], as in Gao et al. (2017). Henceforth we refer to the non-dimensionalized patterns extracted from the MR images as data or observations, $\boldsymbol{y}^{\mathrm{obs}}$, and the corresponding output from the mathematical model, $\mathbf{m}(\boldsymbol{q})$, as the simulations. Here, the argument, $\boldsymbol{q}$, indicates dependence of the outputs on the biomechanical parameters, and the word *simulation* emphasizes that the dynamical equation (6) from Section 2 have no closed-form solution but have to be numerically solved with finite element discretization. Due to the high computational costs, this procedure is not viable in the context of a clinical decision support system. In Section 3.2 we therefore describe an alternative procedure that bypasses simulations from the soft-tissue mechanical model. But first, in Section 3.1, we revise some basic concepts from statistical inference.

3.1 Parameter estimation

Given experimental data $\boldsymbol{y}^{\mathrm{obs}}$, the goal is to find the optimal parameter vector $\hat{\boldsymbol{q}}$ leading to a prediction $\mathbf{m}(\hat{\boldsymbol{q}})$ as close as possible to the data $\boldsymbol{y}^{\mathrm{obs}}$. Let the *target loss* be the non-negative function

$$\ell_{\mathbf{m}}(\boldsymbol{q}) = \alpha\, d\left(\mathbf{m}(\boldsymbol{q}), \boldsymbol{y}^{\mathrm{obs}}\right)^2 + Z, \tag{9}$$

where $d(\cdot, \cdot)$ is a metric measuring the distance between the prediction from the mathematical model (the simulator) at $\boldsymbol{q}$ and the experimental data, while α and Z are some positive constants. The estimate $\hat{\boldsymbol{q}}$ is the value of $\boldsymbol{q}$ that minimizes the loss (9):

$$\hat{\boldsymbol{q}} = \underset{\boldsymbol{q} \in \mathcal{Q}}{\arg\min}\, \ell_{\mathbf{m}}(\boldsymbol{q}). \tag{10}$$

A standard choice for $d(\cdot, \cdot)$ is the Euclidean distance:

$$d_2(\boldsymbol{y}_i, \boldsymbol{y}_j) = \|\boldsymbol{y}_i - \boldsymbol{y}_j\| = \left[\sum_{t=1}^{k} (y_{it} - y_{jt})^2\right]^{1/2}, \tag{11}$$

where $\boldsymbol{y}_i, \boldsymbol{y}_j \in \mathbb{R}^k$. If the data $\boldsymbol{y}^{\mathrm{obs}}$ come from an i.i.d. Gaussian distribution centred at the true simulation

[3] The reference values are, up to 2 decimal places: $a_0 = 0.22$, $b_0 = 1.62$, $a_{\mathrm{f0}} = 2.43$, $a_{\mathrm{s0}} = 0.56$, $b_{\mathrm{f0}} = 1.83$, $b_{\mathrm{s0}} = 0.77$, $a_{\mathrm{fs0}} = 0.39$, and $b_{\mathrm{fs0}} = 1.70$.

[4] We transform the LV cavity volume V via $V \to \frac{V - V_o}{V_o}$, with the reference volume V_o to be taken as the measured volume; see Gao et al. (2017).

output with variance $\sigma^2 \boldsymbol{I}$, minimizing the squared Euclidean loss corresponds to maximizing the log likelihood, which has several optimality criteria (consistency, asymptotic unbiasedness and asymptotic efficiency; see e.g. Casella and Berger (2002)). Several global optimization algorithms have been proposed in the literature, such as genetic algorithms, multistart and simulated annealing methods, reviewed in Locatelli and Schoen (2013). However, all these methods require many iterative function evaluations, and each individual evaluation of the objective function $\ell_{\mathbf{m}}(\boldsymbol{q})$ involves a costly forward simulation from the model, $\mathbf{m}(\boldsymbol{q})$. This leads to high computational costs overall; e.g. those reported in Gao et al. (2015) are in the order of over a week. The direct optimization of the loss function (11) is thus unsuitable for a clinical decision support system.

3.2 Emulation

In order to reduce the computational burden brought by the numerical solution of the dynamical equations (6) from Section 2 with finite element discretization, increasing attention has been drawn to the concept of *emulation* (Kennedy and O'Hagan, 2001; O'Hagan, 2006). An *emulator* $\hat{\mathbf{m}}$, also known as *surrogate model* or *metamodel*, is a statistical approximation of the underlying dynamical model $\mathbf{m}$ based on a set of costly *training runs*:

$$\mathcal{D} = \{\boldsymbol{q}_i, \boldsymbol{y}_i = \mathbf{m}(\boldsymbol{q}_i)\}_{i=1}^n. \tag{12}$$

The training simulations should be obtained by exploiting the fact that all n runs used to fit the surface can be done in parallel, even before seeing any experimental data. Whenever a simulation from the underlying mathematical model is needed at a point which has not been visited before, the costly value $\mathbf{m}(\boldsymbol{q})$ is replaced by a fast prediction from the surrogate model $\hat{\mathbf{m}}(\boldsymbol{q})$. Figure 3 shows a diagram comparing the concepts of simulation (i.e. numerically solving the dynamical model equations) and emulation. The next section, Section 3.3, reviews the statistical model commonly used for emulation, Section 3.4 describes the design of the training set, and Section 4 discusses various alternative emulation strategies.

3.3 Gaussian Processes

The present section provides a brief review of nonparametric Bayesian modelling with Gaussian processes (GPs). For a more comprehensive overview, the reader is referred to Rasmussen and Williams (2006). A stochastic process $\{f(\boldsymbol{q}) : \boldsymbol{q} \in \mathcal{Q}\}$ is said to be a Gaussian process (GP) if and only if, for every n and inputs $\boldsymbol{q}_1, \ldots, \boldsymbol{q}_n$, the random vector $\boldsymbol{f} = (f(\boldsymbol{q}_1), \ldots, f(\boldsymbol{q}_n))$ has a multivariate Gaussian distribution:

$$\boldsymbol{f} \sim \mathrm{N}(\boldsymbol{m}, \boldsymbol{K}).$$

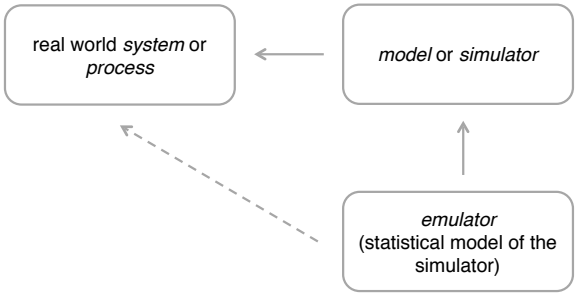


Figure 3: Diagram illustrating the concepts of simulation and emulation. A simulator is based on a mathematical model (in the present application this is the model described in Section 2) and approximates the real world system (solid horizontal arrow). An emulator is a computationally cheap statistical surrogate model that approximates the simulator (solid vertical arrow). Being a double approximation, the emulator indirectly models the real world system (dashed arrow).

Similarly to a multivariate Gaussian, completely specified by a mean vector and a covariance matrix, the GP is parametrized by a mean and a covariance *function*:

$$m(\boldsymbol{q}) = \mathbb{E}[f(\boldsymbol{q})] \tag{13}$$

$$k(\boldsymbol{q}_i, \boldsymbol{q}_j) = \mathrm{Cov}[f(\boldsymbol{q}_i), f(\boldsymbol{q}_j)], \tag{14}$$

respectively returning the mean of a random variable and the covariance between two random variables, as function of the inputs only. In this work we consider a constant mean function $m(\boldsymbol{q}) = c$ for the local GP method (see Section 4.2) and a linear mean function $\boldsymbol{m}(\boldsymbol{q}) = \boldsymbol{b}^\top \boldsymbol{h}(\boldsymbol{q})$, with $\boldsymbol{h}^\top(\boldsymbol{q}) = (1, \boldsymbol{q}^\top)$, for the multivariate output GP method as suggested in Conti et al. (2009); Conti and O'Hagan (2010). The covariance function considered is the ARD Squared Exponential kernel[5]:

$$k(\boldsymbol{q}_i, \boldsymbol{q}_j) = \sigma_f^2 \exp\left\{-\frac{1}{2}\sum_{k=1}^{d} \frac{(q_{ik} - q_{jk})^2}{\lambda_k^2}\right\} + \sigma^2 \delta_{ij}. \tag{15}$$

The GP model hyperparameters, $\boldsymbol{\theta} = (\lambda_1, \ldots, \lambda_d, \sigma_f, \sigma)$, can be estimated either by maximizing the log marginal likelihood or by MCMC sampling, see Rasmussen and Williams (2006) for more details. They represent the signal variance (σ_f^2), the noise variance (σ^2), and intrinsic coordinate-specific lengthscales for function variation along a given dimension ($\lambda_1, \ldots, \lambda_d$).

[5]This kernel is referred to as ARD (for 'automatic relevance determination') in the literature due to the fact that the inference of the length scales λ_k 'automatically' indicates the relevance of the corresponding parameters (Rasmussen and Williams, 2006)

Denote the set of observed data as $\mathcal{D} = \{(\boldsymbol{q}_1, y_1), \ldots, (\boldsymbol{q}_n, y_n)\}$, where $\boldsymbol{q}$ is the input and y is a real-valued output variable. We collectively denote the training outputs as $\boldsymbol{y} = (y_1, \ldots, y_n)$. Conditioning the GP prior on the observed data gives rise to the posterior GP (see Rasmussen and Williams (2006) for more details):

$$
\begin{aligned}
f(\boldsymbol{q}) &\sim \mathrm{GP}(\hat{f}(\boldsymbol{q}), s(\boldsymbol{q}, \boldsymbol{q}')) \qquad (16)\\
\hat{f}(\boldsymbol{q}) &= m(\boldsymbol{q}) + \boldsymbol{k}(\boldsymbol{q})^\top (\boldsymbol{K} + \sigma^2 \boldsymbol{I})^{-1} (\boldsymbol{y} - \boldsymbol{m})\\
s(\boldsymbol{q}, \boldsymbol{q}') &= k(\boldsymbol{q}, \boldsymbol{q}') - \boldsymbol{k}(\boldsymbol{q})^\top (\boldsymbol{K} + \sigma^2 \boldsymbol{I})^{-1} \boldsymbol{k}(\boldsymbol{q}'),
\end{aligned}
$$

where $\boldsymbol{m} = (m(\boldsymbol{q}_1), \ldots, m(\boldsymbol{q}_n))$ is the prior mean at the training points, $\boldsymbol{K} = [k(\boldsymbol{q}_i, \boldsymbol{q}_j)]_{i,j=1}^n$ is the training covariance matrix, while $\boldsymbol{k}(\boldsymbol{q}) = (k(\boldsymbol{q}_1, \boldsymbol{q}), \ldots, k(\boldsymbol{q}_n, \boldsymbol{q}))$ is the n-vector of covariances between the training points and the test point. The conditional expectation function of the posterior process is the best predictor in the sense of minimizing the mean squared prediction error. Hence, the prediction from the emulator at a generic point $\boldsymbol{q}$ is given by $\hat{f}(\boldsymbol{q})$ and the conditional expectation function $\hat{f}(\cdot)$ is often referred to as the *surrogate model*.

3.4 Design of Training Runs

To learn the emulator one has to make a decision on how to design the inputs of the training data (12). Because of the computational complexity of each simulation, we aim to pick each training input $\boldsymbol{q}_i$ in order to cover the whole parameter domain $\mathcal{Q}$ as effectively as possible. Let $\boldsymbol{q} = (q_1, \ldots, q_d)$ denote a generic element of $\mathcal{Q}$. The simplest approach involves defining a grid $\boldsymbol{g}_k \in \mathbb{R}^G$ between a lower and upper bound for each coordinate q_k $(k = 1, \ldots, d)$:

$$
\boldsymbol{g}_k : \mathrm{lb}_k = q_{k1} < \cdots < q_{kG} = \mathrm{ub}_k.
$$

The total number of points $\boldsymbol{q}_i$ at which a simulation is required equals G^d, which for high dimensions d becomes computationally prohibitive. A naive alternative would be to draw samples from a uniform distribution in the d-dimensional domain. However, this can easily lead to points being clustered together, and hence an ineffective coverage of the space. To address this shortcoming, space-filling designs are widely used in the emulation literature, with the Latin hypercube design or the Sobol sequence being the most widely used options; see e.g. Jones et al. (1998); Santner et al. (2003) and Fang et al. (2006). An illustration is given in Figure 4. Improving the design with advanced methods from computational Bayesian statistics is a topical research area, see e.g. Overstall and Woods (2017), but this is beyond the remit of the present study.

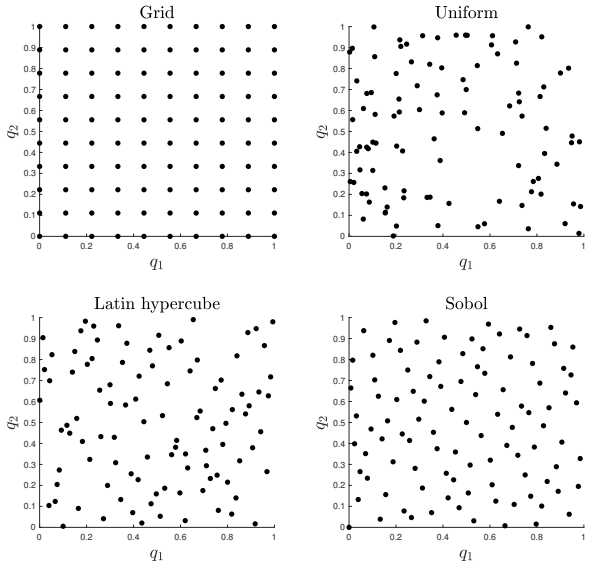


Figure 4: A comparison of different design choices for the training inputs. The plots show 100 points $\{\boldsymbol{q}_i\}$ in the 2D space $[0, 1]^2$ using different design choices: regular grid (top left), sampled from a uniform distribution (top right), Latin hypercube design (bottom left), and from a Sobol sequence (bottom right).

4 Emulation strategies

There are various decisions that one has to take in practical applications of an emulator: emulate the output functions or emulate the loss function; fit separate univariate output GPs or fit a single multivariate-output GP; and how to deal with large training data: fit a local GP, or fit a sparse GP? The present section provides a methodological overview. We will compare the alternative emulation strategies in a comparative evaluation study in Section 7.

4.1 Emulating the outputs versus emulating the loss function

4.1.1 Output emulation

Output emulation represents the strategy of directly emulating the model output, i.e. replacing $\mathbf{m}(\boldsymbol{q})$ by $\hat{\mathbf{m}}(\boldsymbol{q})$. The estimation problem in (10) can be approximated by replacing any evaluation of the computationally expensive mathematical model (the simulator) $\mathbf{m}(\boldsymbol{q})$ by the output from this surrogate model $\hat{\mathbf{m}}(\boldsymbol{q})$. This leads to a loss function that does not involve any further costly simulations and can be optimized using standard optimization algorithms; see e.g. Locatelli and Schoen (2013). The *surrogate-based loss*, given a metric $d(\cdot, \cdot)$, is the positive function:

$$
\ell_{\hat{\mathbf{m}}}(\boldsymbol{q}) = \alpha \, d\left(\hat{\mathbf{m}}(\boldsymbol{q}), \boldsymbol{y}^{\mathrm{obs}}\right)^2 + Z. \qquad (17)
$$

For the Euclidean metric, $d(\boldsymbol{q}_i, \boldsymbol{q}_j) = \|\boldsymbol{q}_i - \boldsymbol{q}_j\|$, we have $\alpha = 1/(2\sigma^2)$ and $Z = Z(\sigma)$; and minimizing the

loss is equivalent to maximizing the likelihood of the data $\boldsymbol{y}^{\text{obs}}$ under the assumptions stated in Section 3.1. The estimate

$$\hat{\boldsymbol{q}} = \arg\min_{\boldsymbol{q}\in\mathcal{Q}} \ell_{\hat{\mathsf{m}}}(\boldsymbol{q}) \tag{18}$$

represents an approximate, but computationally feasible, solution to the minimization of the target loss defined in (10).

The advantage of output emulation is that the process of training the emulator is complete by the time the patient comes into the clinic, and that, at that time, only the above optimization problem has to be solved. The drawback is that either k independent emulators $\hat{\mathsf{m}}_j$ or a k-dimensional output emulator have to be trained, leading to higher computational costs than emulating the loss function directly.

4.1.2 Loss emulation

Loss emulation represents the strategy of directly emulating the loss function (9), and then replacing the original optimization problem (10) by

$$\hat{\boldsymbol{q}} = \arg\min_{\boldsymbol{q}\in\mathcal{Q}} \hat{\ell}_{\mathsf{m}}(\boldsymbol{q}), \tag{19}$$

where $\hat{\ell}_{\mathsf{m}}(\boldsymbol{q})$ is the surrogate loss from the emulation. The advantage is a reduction of the training complexity, as a k-dimensional vector $\boldsymbol{y} = \mathsf{m}(\boldsymbol{q})$ is replaced by a scalar $\ell_{\mathsf{m}}(\boldsymbol{q})$ as the target function. The training data used to build the emulator $\hat{\ell}_{\mathsf{m}}(\boldsymbol{q})$ are represented by the pairs $\{\boldsymbol{q}_i, \ell_{\mathsf{m}}(\boldsymbol{q}_i)\}_{i=1}^n$. These are obtained by transforming the previously collected kD outputs of the space-filling design evaluations to 1D values:

$$\boldsymbol{y}_i = \mathsf{m}(\boldsymbol{q}_i) \quad \longrightarrow \quad \ell_{\mathsf{m}}(\boldsymbol{q}_i).$$

The disadvantage is that, as opposed to output emulation, the emulator can only be trained *after* the patient has come into the clinic and the training data have become available. This is because, in order to compute the distances between the simulations and the patient data, we need the patient data. We need to emphasize, though, that the computational costs of training the emulator "on demand" are still orders of magnitude lower than the repeated finite element discretizations required for the original mathematical model of Section 2. As an aside, we note that the extension of loss emulation along the lines of Bayesian optimization (Shahriari et al., 2016) is not feasible in a clinical context, as this would require additional simulations from the model to be run at the time a clinical decision has to be made, which as opposed to training the emulator is not computationally viable.

4.2 Sparse GP versus local GP

4.2.1 The need for sparsity.

For training the emulator we can in principle generate an arbitrarily large training set from the simulator, i.e. the original mathematical model from Section 2. However, when the sample size n is large, it is usually not feasible to use exact GP regression on the full dataset as described in Section 3.3, due to the $O(n^3)$ computational complexity of the $n \times n$ training covariance matrix $\boldsymbol{K}$ inversion. There are two strategies to address this difficulty. The approach of *sparse GPs* is based on a replacement of the n actual data points by $m \ll n$ so-called inducing points that capture most of the information in the data. In this way, the computational complexity is reduced from $O(n^3)$ to $O(nm^2)$. The alternative approach of *local GPs* is based on a selection of a subset of the data that are closest to the input where a new prediction is to be made. In what follows, we provide a brief methodological summary. An empirical evaluation can be found in Section 7.

4.2.2 Sparse GPs.

The approach of sparse GPs, discussed in Titsias (2009) as an improvement on an earlier method proposed in Snelsen and Ghahramani (2005), considers a fixed number of m inducing variables $\boldsymbol{u} = (u_1, \ldots, u_m)$, with $m \ll n$, corresponding to inputs $\boldsymbol{Z} = [\boldsymbol{z}_1, \ldots, \boldsymbol{z}_m]^\top$. The locations of the inducing points and the kernel hyperparameters are chosen with variational inference to maximize the evidence lower bound (ELBO), i.e. a lower bound on the log marginal likelihood. Denoting by $\boldsymbol{y}$ noisy observations of the unknown true latent function $\boldsymbol{f}$, the ELBO can be derived by applying Jensen's inequality:

$$\begin{aligned}
\log p(\boldsymbol{y}) &= \log \int\int p(\boldsymbol{y}, \boldsymbol{f}, \boldsymbol{u}) \mathrm{d}\boldsymbol{u}\mathrm{d}\boldsymbol{f} \\
&= \log \int\int q(\boldsymbol{f}, \boldsymbol{u}) \frac{p(\boldsymbol{y}, \boldsymbol{f}, \boldsymbol{u})}{q(\boldsymbol{f}, \boldsymbol{u})} \mathrm{d}\boldsymbol{u}\mathrm{d}\boldsymbol{f} \\
&\geq \int\int q(\boldsymbol{f}, \boldsymbol{u}) \log \frac{p(\boldsymbol{y}, \boldsymbol{f}, \boldsymbol{u})}{q(\boldsymbol{f}, \boldsymbol{u})} \mathrm{d}\boldsymbol{u}\mathrm{d}\boldsymbol{f} \\
&= \underbrace{\int\int p(\boldsymbol{f}|\boldsymbol{u}) q(\boldsymbol{u}) \log \frac{p(\boldsymbol{y}|\boldsymbol{f}) p(\boldsymbol{u})}{q(\boldsymbol{u})} \mathrm{d}\boldsymbol{u}\mathrm{d}\boldsymbol{f}}_{\mathcal{F}(q(\boldsymbol{u}))}
\end{aligned}$$

where $q(\boldsymbol{f}, \boldsymbol{u}) = p(\boldsymbol{f}|\boldsymbol{u})q(\boldsymbol{u})$, $p(\boldsymbol{y}|\boldsymbol{f})$ is the probability of the observations given the latent function, which is assumed to be an i.i.d. Gaussian distribution, and $p(\boldsymbol{f}|\boldsymbol{u})$ is the probability of the latent function given the function values at the inducing points, which can be obtained from the Gaussian process. Note that $p(\boldsymbol{f}|\boldsymbol{u})$ has cancelled out inside the log. The ELBO, $\mathcal{F}(q(\boldsymbol{u}))$, is first maximized with respect to the variational distribution $q(\boldsymbol{u})$. This can be done analyti-

cally, as shown in Titsias (2009), and leads to

$$\begin{aligned}\mathcal{F} &= \max_{q(\boldsymbol{u})} \mathcal{F}(q(\boldsymbol{u})) \\ &= \log\left[\mathrm{N}(\boldsymbol{y}|\boldsymbol{0}, \sigma^2\boldsymbol{I} + \boldsymbol{Q}_{nn}\right] - \frac{1}{2\sigma^2}\mathrm{trace}(\tilde{\boldsymbol{K}}_{nn})\end{aligned}$$

where $\boldsymbol{Q}_{nn} = \boldsymbol{K}_{nm}\boldsymbol{K}_{mm}^{-1}\boldsymbol{K}_{mn}$, $\tilde{\boldsymbol{K}}_{nn} = \boldsymbol{K}_{nn} - \boldsymbol{Q}_{nn}$, $\boldsymbol{K}_{nn}$ is the covariance matrix of the original n training points, $\boldsymbol{K}_{mm}$ is the covariance matrix at the m inducing points, and $\boldsymbol{K}_{mn}$, $\boldsymbol{K}_{nm}$ are the cross-covariance matrices between the n original training and m inducing points. Note that the computational costs of computing $\boldsymbol{Q}_{mm}$, and hence $\mathcal{F}$, is $O(nm^2)$, as opposed to $O(n^3)$ for the original GP. The second step of the optimization procedure consists in finding the inducing points $\boldsymbol{Z} = [\boldsymbol{z}_1, \ldots, \boldsymbol{z}_m]^\top$ that maximize $\mathcal{F}$. To this end, we use the iterative procedure proposed in Titsias (2009).

4.2.3 Local GPs.

A local Gaussian process approach based on the K-nearest-neighbours was proposed in Gramacy and Apley (2015). This method uses the standard GP prediction formulas described in Section 3.3, but subsetting the training data. Whenever we require a prediction at a given input, we find the training inputs representing the K nearest neighbours in input-domain, which will form the local set of training inputs, and the corresponding outputs will represent the local training outputs. Note that every time we query a prediction at a different input, the training sets need to be re-computed and the GP needs to be re-trained. However, because of the small number of neighbours $K \ll 1000$ usually selected, this method is computationally fast; see Gramacy and Apley (2015) for further details.

For training data $\mathcal{D} = \{(\boldsymbol{q}_1, y_1), \ldots, (\boldsymbol{q}_n, y_n)\} = \{\boldsymbol{Q}, \boldsymbol{y}\}$, we can summarize the algorithm as follows:

Algorithm 1. *Predicting from a local Gaussian process at $\boldsymbol{q}_*$:*

1. Find the indices $\mathcal{N}(\boldsymbol{q}_*)$ of the points in $\boldsymbol{Q}$ having the K smallest Euclidean distances from $\boldsymbol{q}_*$;
2. Training inputs: $\boldsymbol{Q}_K(\boldsymbol{q}_*) = \{\boldsymbol{q}'_1, \ldots, \boldsymbol{q}'_K\} = \{\boldsymbol{q}_i : i \in \mathcal{N}(\boldsymbol{q}_*)\}$;
3. Training outputs: $\boldsymbol{y}_K(\boldsymbol{q}_*) = \{y'_1, \ldots, y'_K\} = \{y_i : i \in \mathcal{N}(\boldsymbol{q}_*)\}$;
4. Train a GP using the data $\mathcal{D}_K(\boldsymbol{q}_*) = \{\boldsymbol{Q}_K(\boldsymbol{q}_*), \boldsymbol{y}_K(\boldsymbol{q}_*)\}$;
5. Predictive mean: $\hat{f}(\boldsymbol{q}_*) = m(\boldsymbol{q}_*) + \boldsymbol{k}(\boldsymbol{q}_*)^\top[\boldsymbol{K} + \sigma^2\boldsymbol{I}]^{-1}(\boldsymbol{y}_K(\boldsymbol{q}_*) - \boldsymbol{m})$;
6. Predictive variance: $s^2(\boldsymbol{q}_*) = k(\boldsymbol{q}_*, \boldsymbol{q}_*) - \boldsymbol{k}(\boldsymbol{q}_*)^\top[\boldsymbol{K} + \sigma^2\boldsymbol{I}]^{-1}\boldsymbol{k}(\boldsymbol{q}_*)$.

In the algorithm above, $\boldsymbol{K} = [k(\boldsymbol{q}'_i, \boldsymbol{q}'_j)]_{i,j=1}^K$ is the $K \times K$ training covariance matrix, the $K \times 1$ vector of covariances between the training points and the test point is $\boldsymbol{k}(\boldsymbol{q}_*) = (k(\boldsymbol{q}'_1, \boldsymbol{q}_*), \ldots, k(\boldsymbol{q}'_K, \boldsymbol{q}_*))$ and $\boldsymbol{m} = (m(\boldsymbol{q}'_1), \ldots, m(\boldsymbol{q}'_K))$ is the $K \times 1$ prior mean vector. Note that whenever the target parameters change, e.g. during an iterative optimization, the algorithm has to be repeated. We consider a constant mean function $m(\boldsymbol{x}) = c$ and the squared exponential kernel (15), as widely used in the emulation of computer codes literature; see e.g. Fang et al. (2006); Santner et al. (2003). The model hyperparameters are estimated by maximizing the log marginal likelihood using the Quasi-Newton method, with σ initialized at a small value, $\sigma = 10^{-2}$ (since the model described in Section 2 is deterministic).

This method was found to be the best in the comprehensive comparison presented in Davies et al. (2019) and represents a benchmark for the current study, along with the expensive optimization problem solved in Gao et al. (2015).

4.3 Separate univariate output GPs versus multivariate output GP

Recall that output emulation is the strategy of directly emulating the model output, i.e. replacing $\mathsf{m}(\boldsymbol{q})$ by $\hat{\mathsf{m}}(\boldsymbol{q})$. If the model is multivariate, i.e. $\mathsf{m} = (\mathsf{m}_1, \ldots, \mathsf{m}_k)$, a straightforward approach is to fit k independent real-valued emulators $\hat{\mathsf{m}}_j(\boldsymbol{q})$ of $y_j = \mathsf{m}_j(\boldsymbol{q})$ for $j = 1, \ldots, k$, and consider the multivariate surrogate model as the vector $\hat{\mathbf{m}} = (\hat{\mathsf{m}}_1, \ldots, \hat{\mathsf{m}}_k)$. A prediction from $\hat{\mathbf{m}}(\boldsymbol{q})$ is then obtained by predicting from each univariate component $\hat{\mathsf{m}}_j(\boldsymbol{q})$ for $j = 1, \ldots, k$. If multiple cores are available on the machine, it is possible to take advantage of the parallel nature of the fitting and prediction tasks by fitting (or predicting from) a univariate emulator on each core and obtaining k emulators (or predictions) at the cost of one.

Independence between the individual outputs is a restrictive assumption to place on the system, though. In this section, we briefly review a method for relaxing this constraint, proposed by Conti et al. (2009) and Conti and O'Hagan (2010). We found that an alternative method based on latent Gaussian processes, proposed by Alvarez and Lawrence (2011), suffered either from low accuracy or excessive computational costs in the context of our study, and we therefore relegate these details to the supplementary material. Conti et al. (2009) and Conti and O'Hagan (2010) introduce a GP prior over the outputs of the simulator $\boldsymbol{y} = \boldsymbol{f}(\cdot)$ as follows:

$$\boldsymbol{f}(\cdot)|\mathbf{B}, \boldsymbol{\Sigma}, \mathbf{r} \sim \mathrm{N}(\boldsymbol{m}(\cdot), k(\cdot,\cdot)\boldsymbol{\Sigma}) \tag{20}$$

where $k(\cdot,\cdot)$ is the kernel function providing the spatial correlation over the parameter space (equal for each output), and $\boldsymbol{\Sigma}$ provides the covariance between the outputs of the simulator. Let $[\boldsymbol{A}]_{ij}$ denote entry

(i, j) of matrix $\boldsymbol{A}$. The covariance structure can be summarized by the following:

$$\mathrm{Cov}(f_1(\boldsymbol{q}_3), f_2(\boldsymbol{q}_4)) = k(\boldsymbol{q}_3, \boldsymbol{q}_4)[\boldsymbol{\Sigma}]_{12}, \tag{21}$$

where $f_l(\cdot)$ is the lth output such that for $\boldsymbol{Y}$ a matrix with lth column containing output l evaluated over the design space, the following holds:

$$\mathrm{Cov}(\mathrm{vec}(\boldsymbol{Y}), \mathrm{vec}(\boldsymbol{Y})) = \boldsymbol{\Sigma} \otimes \boldsymbol{K}. \tag{22}$$

The matrix $\boldsymbol{K}$ contains the evaluations of the kernel $k(\cdot, \cdot)$ over the design space and $\mathrm{vec}(\cdot)$ is the matrix vectorization operator which stacks together the columns of the matrix in order to form a vector. Letting $\boldsymbol{H}$ be a matrix with ith column containing $\boldsymbol{h}(\boldsymbol{q}_i)$, this leads to the matrix normal distribution as a generalization of the multivariate Gaussian:

$$\mathbf{Y}|\mathbf{B}, \boldsymbol{\Sigma}, \mathbf{r} \sim \mathrm{MN}(\mathbf{HB}, \boldsymbol{\Sigma} \otimes \boldsymbol{K}) \tag{23}$$

which combined with the prior in (20) gives a full posterior for the latent variable of the form:

$$f(\cdot)|\boldsymbol{B}, \boldsymbol{\Sigma}, \boldsymbol{r}, \boldsymbol{Y} \sim \mathrm{N}(\boldsymbol{m}^*(\cdot), k^*(\cdot, \cdot)\boldsymbol{\Sigma}) \tag{24}$$

where $k^*(\boldsymbol{q}_1, \boldsymbol{q}_2)$ and $\boldsymbol{m}^*(\cdot)$ can be found using standard Gaussian identities (see Section 2.3 in Bishop (2006)). One can sample from a posterior distribution of the roughness hyperparameter by adopting a prior of the form $\pi(\boldsymbol{B}, \boldsymbol{\Sigma}, \boldsymbol{r}) = \pi(\boldsymbol{r})\pi(\boldsymbol{B}, \boldsymbol{\Sigma}|\boldsymbol{r}) \propto \prod_{i=1}^{p}(1+r_i^2)^{-1}|\boldsymbol{\Sigma}|^{-\frac{q+1}{2}}$ ($p =$ dimension of inputs) and integrating the hyperparameters $\boldsymbol{B}$ and $\boldsymbol{\Sigma}$ out of the full posterior in (24) such that we sample from a distribution of the form:

$$\begin{aligned}\pi_{\boldsymbol{R}}(\boldsymbol{r}|\boldsymbol{Y}) \propto \pi_{\boldsymbol{R}}(\boldsymbol{r})|\boldsymbol{K}|^{-\frac{M}{2}} \\ \cdot |\boldsymbol{H}^\top \boldsymbol{K} \boldsymbol{H}|^{-\frac{M}{2}} |\boldsymbol{Y}^\top \boldsymbol{G} \boldsymbol{Y}|^{-\frac{n-m}{2}}\end{aligned} \tag{25}$$

where $\boldsymbol{G} = \boldsymbol{K}^{-1} - \boldsymbol{K}^{-1}\boldsymbol{H}(\boldsymbol{H}^\top \boldsymbol{K}^{-1}\boldsymbol{H})^{-1}\boldsymbol{H}^\top \boldsymbol{K}^{-1}$. As suggested by the authors, we adopt the median from these MCMC samples as the lengthscales in our GP (for validity of these summaries, unimodality of the posterior distributions can be seen in contour plots).

Taking point estimates of the parameters $\hat{\boldsymbol{B}}_{\mathrm{GLS}}$ and $\boldsymbol{\Sigma}_{\mathrm{GLS}}$ (Dutilleul, 1999), the resultant posterior for the emulator is given by the following multivariate Student's process:

$$f(\cdot)|\boldsymbol{r}, \boldsymbol{Y} \sim \mathrm{T}_l(\hat{\boldsymbol{f}}(\cdot), \rho(\cdot, \cdot)\hat{\boldsymbol{\Sigma}}_{\mathrm{GLS}}) \tag{26}$$

where

$$\begin{aligned}\hat{\boldsymbol{f}}(\boldsymbol{q}_1) = \hat{\boldsymbol{B}}_{\mathrm{GLS}}\boldsymbol{h}(\boldsymbol{q}_1) \\ + (\boldsymbol{Y} - \boldsymbol{H}\hat{\boldsymbol{B}}_{\mathrm{GLS}})^\top \boldsymbol{K}^{-1}\boldsymbol{t}(\boldsymbol{q}_1)\end{aligned} \tag{27}$$

and

$$\begin{aligned}\rho(\boldsymbol{q}_1, \boldsymbol{q}_2) = k^*(\boldsymbol{q}_1, \boldsymbol{q}_2) \\ + [\boldsymbol{h}(\boldsymbol{q}_1) - \boldsymbol{H}^\top \boldsymbol{K}^{-1}\boldsymbol{t}(\boldsymbol{q}_2)]^\top \\ \cdot (\boldsymbol{H}\boldsymbol{K}^{-1}\boldsymbol{H})^{-1}[\boldsymbol{h}(\boldsymbol{q}_1) - \boldsymbol{H}^\top \boldsymbol{K}^{-1}\boldsymbol{t}(\boldsymbol{q}_2)]\end{aligned} \tag{28}$$

We can take the mean of this distribution, (27), as our estimate of the simulator.

In order to permit real time decision making, as well as preventing stability issues in matrix inversions, we have to consider an approximation of the full GP approach. Unfortunately, higher computational costs in the multivariate output framework imply that the local GP approach in parameter space (see Section 4.2) is no longer viable. We therefore opt to find the *nearest neighbours in function space* and obtain the local GP based on the nearest neighbours of the vector of outputs for which parameter estimation is required. Computational costs are reduced since the local GP only needs to be fitted once, avoiding the repeated re-estimations inherent in Algorithm 1. The procedure can be summarized in the following algorithm, where predictive means and variances are provided in the context of the method of Conti and O'Hagan:

Algorithm 2. *Predicting from a local multioutput Gaussian process at* $(\boldsymbol{q}_*, \boldsymbol{y}_*)$*:*

1. Find the indices $\mathcal{N}(\boldsymbol{y}_*)$ of the points in $\boldsymbol{Y}$ having the K smallest Euclidean distances from $\boldsymbol{y}_*$;
2. Training inputs: $\boldsymbol{Q}_K(\boldsymbol{y}_*) = \{\boldsymbol{q}'_1, \ldots, \boldsymbol{q}'_K\} = \{\boldsymbol{q}_i : i \in \mathcal{N}(\boldsymbol{y}_*)\}$;
3. Training outputs: $\boldsymbol{y}_K(\boldsymbol{y}_*) = \{y'_1, \ldots, y'_K\} = \{y_i : i \in \mathcal{N}(\boldsymbol{y}_*)\}$;
4. Train a multivariate output GP using the data $\mathcal{D}_K(\boldsymbol{y}_*) = \{\boldsymbol{Q}_K(\boldsymbol{y}_*), \boldsymbol{y}_K(\boldsymbol{y}_*)\}$;
5. Predictive mean at test point $\boldsymbol{q}_*$, see (27): $\hat{\boldsymbol{f}}(\boldsymbol{q}_*) = \hat{\boldsymbol{B}}_{\mathrm{GLS}}\boldsymbol{h}(\boldsymbol{x}_*) + (\boldsymbol{Y} - \boldsymbol{H}\hat{\boldsymbol{B}}_{\mathrm{GLS}})^\top \boldsymbol{K}^{-1}\boldsymbol{t}(\boldsymbol{q}_*)$;
6. Predictive variance at test point $\boldsymbol{q}_*$, see (28): $s^2(\boldsymbol{q}_*) = \rho(\boldsymbol{q}_*, \boldsymbol{q}_*)\hat{\boldsymbol{\Sigma}}_{\mathrm{GLS}}$

In the algorithm above, $\hat{\boldsymbol{f}}(\boldsymbol{q}_*)$ denotes the posterior mean of the multivariate output GP given in (27) and the predictive variance $s(\boldsymbol{q}_*)$ is given in the covariance of (26). $\boldsymbol{K}$ denotes the autocovariance matrix from the initial Gaussian process prior outlined in (20). The model hyperparameters are sampled from the posterior distribution by Metropolis Hastings MCMC (Hastings, 1970). To reduce the computational costs for real-time decision support at the clinic, we reduce the posterior distribution of the hyperparameters to a point estimate - the posterior median - which will be plugged into the GP used for emulation.[6] Using multiple MCMC chains in parallel permits the use of MCMC convergence diagnostics via the potential scale reduction factor (Gelman et al., 2004), without incurring any additional computational costs.

[6] We confirmed empirically that this method gives very similar results to a proper Bayesian posterior average using the full posterior sample of hyperparameters. However, predictions for the latter approach require the inversion of a posterior population of covariance matrices, which is computationally too expensive for the clinic.

5 Building and applying the emulator

Building and applying the emulator is a process in three phases. In the first phase, a set of constitutive parameter vectors $\boldsymbol{q}$ is generated from a Sobol sequence. In general, these are the 8D vectors from the Holzapfel-Ogden model defined in (1), but based on the discussion at the end of Section 2, we have reduced the dimension to 4D, as defined in (7). For each parameter vector $\boldsymbol{q}$, the dynamical equations (6) from Section 2 are solved numerically with finite element discretization, ideally by massive parallelization, to obtain the corresponding data vectors $\boldsymbol{y}^{\text{obs}}$: these are 25D vectors of 24 peak circumferential stains at well-defined positions at the LV wall and the non-dimensionalized LV chamber volume at end of diastole; see the beginning of Section 3 for details. In the second phase, Gaussian process regression is applied to the training set obtained in the first phase. Depending on the mode of operation, this training set can either consist of a set of independent scalar outputs, output vectors, or loss functions. In the third phase, an iterative optimization algorithm is applied to the emulated loss function, to obtain the parameter estimate for new test data. In a clinical context, Phase 1 is carried out before using the emulator in the clinic. Phase 3 is the way the emulator is used as a decision support system in the clinic, with the test data corresponding to data obtained from a group of patients. Phase 2 depends on the mode of operation. When emulating the output, this phase can be completed before using the emulator in the clinic. When emulating the loss function, the emulator has to be created "on demand" after obtaining the patient data.

5.1 Phase 1: Parallel simulations from the model

In order to build the emulator we need a set of training runs $\mathcal{D} = \{(\boldsymbol{q}_i, \boldsymbol{y}_i)\}_{i=1}^n$, for parameter vectors $\boldsymbol{q}_i$ and associated model outputs $\boldsymbol{y}_i$. In our study, the training inputs $\boldsymbol{Q} = [\boldsymbol{q}_1, \ldots, \boldsymbol{q}_n]^\top$ represent $n = 10,000$ points from a Sobol sequence in $[0.1, 5]^4$, where the domain has been chosen to represent typical parameter ranges from the literature (Gao et al., 2017). The corresponding outputs $\boldsymbol{y}_i$ are obtained as described at the beginning of Section 3.

5.2 Phase 2: Training the emulator

As discussed above, there are three different emulation methods. However, what all these methods have in common is the construction of the $K \times K$ training covariance matrix $\boldsymbol{K} = [k(\boldsymbol{q}_i, \boldsymbol{q}_j)]_{i,j=1}^K$, the $K \times 1$ vector of covariances between the training points and the test point $\boldsymbol{k}(\boldsymbol{q}_*) = (k(\boldsymbol{q}_1, \boldsymbol{q}_*), \ldots, k(\boldsymbol{q}_K, \boldsymbol{q}_*))$, and the specification of the mean function $\boldsymbol{m} = (m(\boldsymbol{q}_1), \ldots, m(\boldsymbol{q}_K))$. The choice of K depends on the emulation strategy. When using a standard GP, $K = n$, i.e. K is the number of training points. When using a sparse GP, $K = m$, where m is the number of inducement points. For a local GP, K is the number of nearest neighbours to the query point. For the univariate-output GPs, we chose a constant mean function $m(\boldsymbol{q}) = c$. For the multi-output GP, we followed Conti et al. (2009) and Conti and O'Hagan (2010), and used a low-order polynomial $m(\boldsymbol{q}) = \boldsymbol{b}^\top \boldsymbol{h}(\boldsymbol{q})$ with $\boldsymbol{h}^\top(\boldsymbol{q}) = (1, \boldsymbol{q}^\top)$. For the kernel $k(\boldsymbol{q}_i, \boldsymbol{q}_j)$ we chose the squared exponential function from equation (15) as in our previous work (Gao et al., 2017). This kernel is widely used in the emulation of computer codes literature, see e.g. Fang et al. (2006) and Santner et al. (2003). This choice of kernel requires the selection of $2 + \dim(\boldsymbol{q})$ hyperparameters, for the vertical scale, the noise variance, and the length scale associated with each of the $\dim(\boldsymbol{q})$ parameters. For the univariate output GPs, the hyperparameters were estimated by maximizing the log marginal likelihood, using the Quasi-Newton method with multiple restarts (for avoiding local optima). For the multivariate output GPs, we used the sampling-based procedure described at the end of Section 4.3. These hyperparameters were either obtained with standard iterative optimization algorithms to maximize the marginal likelihood, or they were sampled from the posterior distribution with MCMC; see Sections 3 and 4.3 for details.

5.3 Phase 3: Using the emulator for parameter estimation

The final step is the minimization of the loss function (10). When the outputs are emulated, then the loss function (9) is approximated by the surrogate loss (17), and the task is to find a solution to the optimization problem (18). Conversely, when the loss function is emulated directly, then the task is to find a solution to the optimization problem (19), replacing the true loss (9) by the output from the emulator. In either case, we face a high-dimensional and typically multimodal optimization problem, for which a variety of algorithms have been proposed in the literature. In the present work, the surrogate-based loss and the emulated loss are optimized using the Global Search algorithm by Ugray et al. (2007), implemented in MATLAB's Global Optimization toolbox.[7]. See Section 2 in the online supplementary material for more details.

[7] `https://uk.mathworks.com/products/global-optimization.html`

6 Data

The objective of the present article is to assess the performance of the emulation strategies discussed in Section 4 in a comparative evaluation study. To this end, we use both simulated data from the LV model and real MRI data from a healthy subject. Training data are obtained by simulating the forward FE LV model with sets of chosen parameters. Note that only in the former case, the true parameters are known and a performance evaluation based on an objective gold standard is feasible, while in the latter case we have to resort to a comparison with the literature.

6.1 Simulated data

We generated a simulated test set of sample size $n_{\text{test}} = 100$ by continuing the Sobol sequence that was used for generating the training data. This ensures that the whole parameter space is uniformly covered, while guaranteeing that the parameter vectors used for testing, and hence their associated data vectors, are different and independent from the training data. The format of the test data is the same as the training data: a 25-dimensional vector $\mathbf{m}(\boldsymbol{q})$ representing 24 circumferential strains and the non-dimensionalized left ventricle volume taken at end of diastole, as described in Section 6.2. So formally, the test set is of the form

$$\mathcal{D} = \{\mathbf{m}(\boldsymbol{q}_{n+1}), \ldots, \mathbf{m}(\boldsymbol{q}_{n+n_{\text{test}}})\} \tag{29}$$

with $\{\boldsymbol{q}_{n+1}, \ldots, \boldsymbol{q}_{n+n_{\text{test}}}\}$ representing the n_{test} parameter vectors obtained by continuing the Sobol sequence after the first $n = 10,000$ points used for generating the training set. Further details are provided in Section 3 of the supplementary material.

6.2 MRI cine data

A 3D left ventricular (LV) geometry model was reconstructed from a MR imaging study of a healthy volunteer (male, 31 years). The MRI study was conducted on a Siemens MAGNETOM Avanto (Erlangen, Germany) 1.5-Tesla scanner with a 12-element phased array cardiac surface coil. Cine MR images were acquired using the steady-state precession imaging protocol at short-axial planes from the base to apex, and three long-axial views (the left ventricular outflow tract view, the four-chamber view, and the one-chamber view) as shown in Figure 1. Short-axis and long-axis cine images were then manually segmented to extract the endocardial and epicardial boundaries at early-diastole when the LV pressure was lowest (see Figure 1). The 3D LV model was reconstructed in Solidworks (Dassault Systemes SolidWorks Corp., Waltham, MA USA) using B-spline surfaces. Tetrahedron elements were generated to represent the whole LV domain with 133,000 elements and 26,000 nodes, shown in Figure 2(a).

Twenty four peak segmentally averaged circumferential strains in diastole at short-axis cine images were measured using a B-spline deformable registration approach (Gao et al., 2014a) with 4 positions of short-axis cine images from the basal to the middle ventricle. Following the clinical convention, strains were calculated with respect to the end-diastolic phase. The end-diastolic volume was also calculated from short-axis cine images at end of diastole. Ventricular pressure can only be measured invasively, and therefore (for ethical reasons) is not available for healthy volunteers. For that reason, a population-based average end-diastolic pressure was assumed, which is 8 mmHg.

7 Results

7.1 Evaluating sparse versus local GPs

We compared the performance of the sparse GP, described in Section 4.2.2, and the local GP, described in Section 4.2.3, on the simulated test data, described in Section 6.1, using the framework of separate output emulation, as described in Section 4.3.

For the sparse GP we tried 100, 500 and 1000 inducing points, using the code accompanying the paper by Titsias (2009). The computational costs for the prediction at a new query point were between 0.5 and 0.6 seconds for 100 inducing points, and in the order of several seconds for 1000 inducing points[8]. These computational costs are accumulated over all the steps of the iterative optimization routine for solving the optimization problem (17). For instance, the computational costs for an optimization routine with 1000 iterative steps are in the order of 10 minutes with 100 inducing points, and in the order of an hour with 1000 inducing points. Since the ultimate goal of emulation is fast decision support in the clinic, we restricted our analysis to 100 inducing points.

Figure 5 shows the predictive accuracy of the sparse GP model on the test data. Each panel shows the true test outputs (horizontal axis) vs the prediction of the sparse GP (vertical axis) at the test inputs. We can see that the fit of some variables, like the LV chamber volume and Strains 17, 19, 20, 21, 22 and 23 are slightly off the perfect prediction line comparing the true with the predicted parameter values. The predictive accuracy improves by increasing the number of inducing points, but at the cost of a slower prediction time, as discussed above.

For comparison, we used a local GP with the same number of nearest neighbours as the number of inducing points for the sparse GP: $K = 100$. The CPU time required to get a prediction at a new query point

[8] Dual Intel Xeon CPU E5-2699 v3, 2.30GHz, 36 cores and 128GB memory.

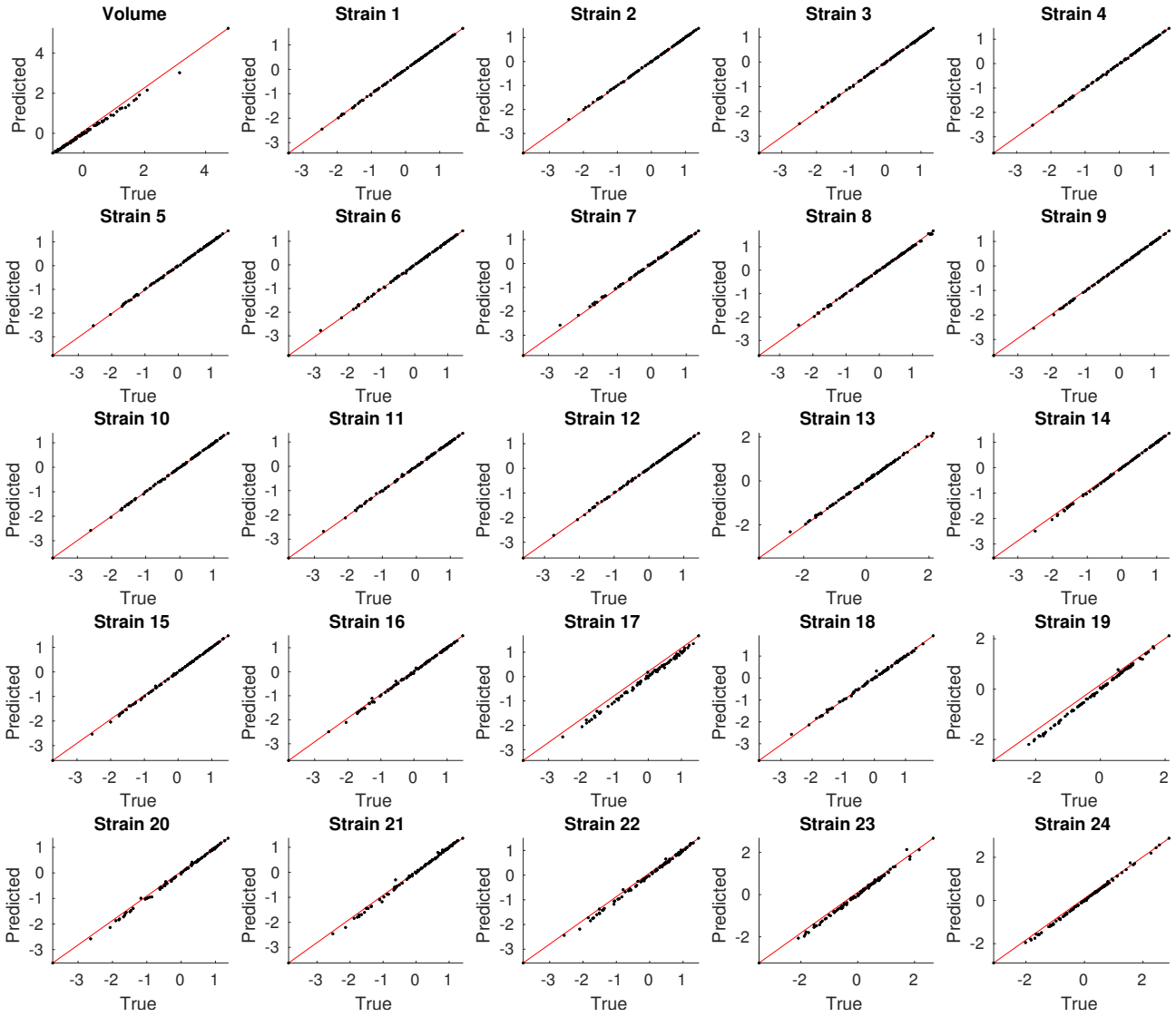


Figure 5: Results for emulation with variational sparse GPs. The figure shows for each of the 25 outputs (LV volume and 24 longitudinal circumferential strains at end of diastole) the true vs predicted test outputs using separate output emulation, as described in Section 4.3, with variational sparse GPs (described in Section 4.2.2) and $K = 100$ inducing points.

was approximately 0.18 seconds[9] We evaluated the predictive accuracy on the test data. Figure 6 shows that local GP regression using the $K = 100$ nearest-neighbours leads to accurate predictions at the test inputs, with the predicted and true test outputs all lying on or very close to the perfect prediction line comparing the true with the predicted parameter values.

This finding suggests that the local GP approach achieves improved accuracy at lower computational costs, and was hence used in the subsequent studies.

[9]Dual Intel Xeon CPU E5-2699 v3, 2.30GHz, 36 cores and 128GB memory.

7.2 Evaluating loss versus output emulation

We have extended an earlier study (Davies et al., 2019) and compared the paradigms of loss versus output emulation, as discussed in Section 4.1, using both separate univariate output GPs and multivariate output GPs, as discussed in Section 4.3. For the latter case, the method proposed in Conti and O'Hagan (2010) was used. For the loss function (11) we used the Euclidean distance. All simulations used local GPs with $K = 100$ nearest neighbours, as described in Section 4.2.3, with Algorithm 1 used for the univariate output GP, and Algorithm 2 used for the multivariate output GP. In summary, the competing methods are as follows:

M1 Output emulation using local GPs;

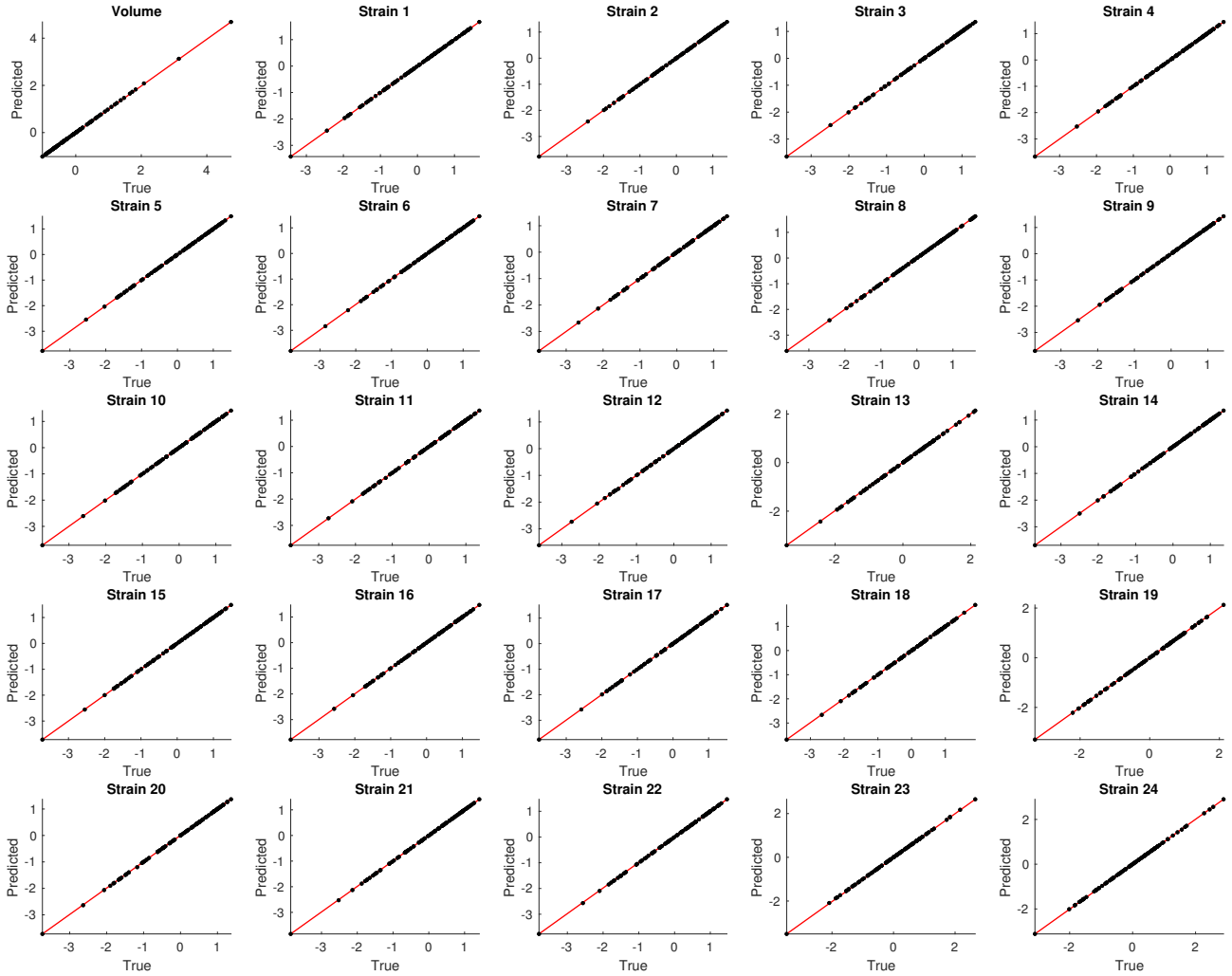


Figure 6: Results for emulation with local GPs. The figure shows for each of the 25 outputs (LV volume and 24 longitudinal circumferential strains at end of diastole) the true vs predicted test outputs using separate output emulation, as described in Section 4.3, with local GPs (described in Section 4.2.3) using the $K = 100$ nearest neighbours.

M2 Loss emulation using local GPs;

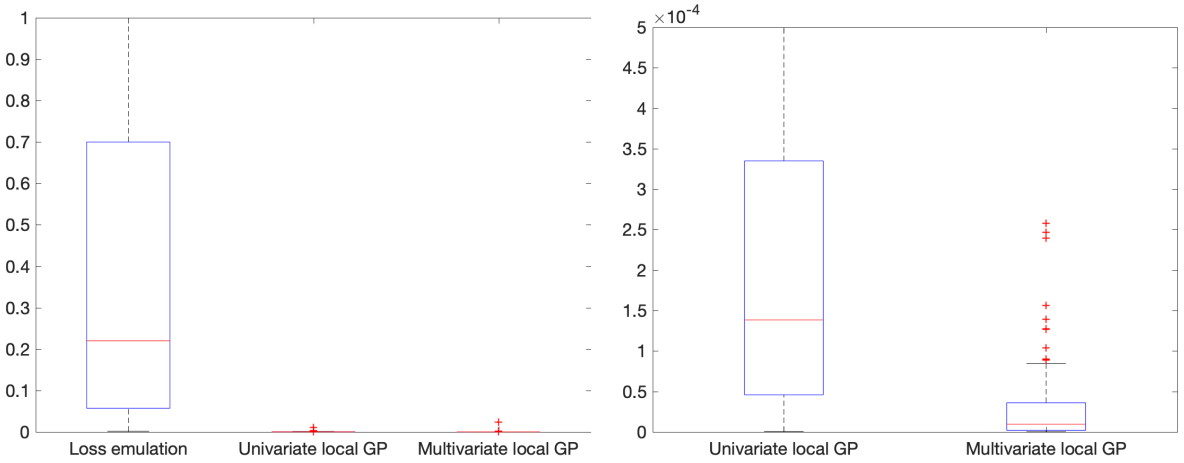
M3 Output emulation using local multivariate output GPs.

Let $\mathcal{D} = \{(\boldsymbol{q}_i, \boldsymbol{y}_i)\}_{i=1}^{n}$ denote the $n = 10,000$ training runs and $\mathcal{D}_{\text{test}} = \{(\boldsymbol{q}_t, \boldsymbol{y}_t)\}_{t=n+1}^{n+m}$ denote the $m = 100$ test data; the latter are not used to fit the GP models. For each test output $\boldsymbol{y}_t \in \mathcal{D}_{\text{test}}$ we estimate the corresponding parameter vector $\hat{\boldsymbol{q}}_t$ using the 3 methods summarized above. We now compare the estimated $\hat{\boldsymbol{q}}_t$ to the known test input $\boldsymbol{q}_t$ using the mean squared error (MSE) score:

$$\text{MSE}_t = \frac{1}{d}\sum_{k=1}^{d}(\hat{q}_{tk} - q_{tk})^2,$$

obtaining a sample of 100 MSE_t scores for each method. Table 1 reports the median out-of-sample MSE and the interquartile range for the 3 different approaches, and highlights in bold the best combination found. In Figure 7 we show the distribution of the 100 MSE_t scores for each method using boxplots. Panel (a) shows the original y-axis scale, while Panel (b) shows a reduced y-axis scale to focus on the lowest MSE scores.

Both Table 1 and Figure 7 show that emulating the output and then minimizing the surrogate-based loss (18) leads to a substantially lower error than emulating the loss function directly and then minimizing it via (19). Depending on the output GP method used, this error reduction is between 3 and 5 orders of magnitude. The explanation for this result is presumably related to the information loss inherent in mapping 25 separate outputs (1 LV chamber volume and 24 circumferential strains) into a single scalar quantity. This suggests that the higher computational costs of the more detailed emulation are rewarded by higher accuracy.



(a) **(b)**

Figure 7: Distribution of the 100 out-of-sample MSE scores for each method. (a) Boxplots of the mean squared error for all 3 methods in the original y-axis scale and (b) with a reduced y-axis scale. The methods from left to right on the left plot are as follows: Local GP emulation of the loss, local GP emulation of the outputs, and local multiple output GP emulation. The plot on the right includes only the output emulation methods.

Table 1: Comparison of the different emulation strategies. Median (1st, 3rd) quartiles of the mean squared error distribution for the out-of-sample parameter vectors. The method with the lowest median MSE is highlighted in bold: Output emulation using local multivariate output GPs.

		MSE
Statistical approximation	Emulation target	Median (1st, 3rd) quartiles
Single output GP	Output	1×10^{-4} $(4 \times 10^{-5},\ 3 \times 10^{-4})$
Single output GP	Loss	2×10^{-1} $(6 \times 10^{-2},\ 7 \times 10^{-1})$
Multivariate output GP	Output	$\mathbf{5 \times 10^{-6}}$ $\mathbf{(1 \times 10^{-6},\ 3 \times 10^{-5})}$

7.3 Evaluating single output versus multivariate output GPs

Table 1 and Figure 7 allow us to compare the performance of the multivariate output GP with that of separate univariate output GPs. The parameter estimation error achieved with the multivariate output GP is about two orders of magnitude lower than the error obtained with the separate univariate output GPs. This suggests that additional accuracy can be achieved by explicitly modelling the correlations between the various outputs. For the previous comparison, we used a fixed value of $K = 100$ nearest neighbours in Algorithm 2, to mimic the value used for the univariate output GPs in Algorithm 1, as discussed in Section 7.1. However, we also investigated the dependence of the parameter estimation accuracy (in terms of MSE) on K, the number of nearest neighbours used in the local GP algorithm, by repeating the estimation of the GP for different values of K. The results are shown in Figure 8. The figure shows that when decreasing the value of K to smaller values, $K < 100$, the performance deteriorates, as expected. Interestingly, the performance also degrades when increasing the value of K beyond $K > 200$. At first, this appears counter-intuitive. However, a possible explanation is as follows. Including additional parameters that are far away from the target parameter have a negligible direct influence on the GP output, which follows from the exponential decay implied by the functional form of the kernel; see (15). However, if the characteristic length scales, expressed by the hyperparameters λ_k in (15), are different in different areas of the parameter space, then the increase of K can lead to a suboptimal estimate of the length scale hyperparameters λ_k, and hence a deterioration of the performance overall. Varying length scales could be modelled with non-stationary kernels, as opposed to the stationary kernel that we have used (and which is commonly used). However, Figure 8 suggests that in addition to reducing the computational complexity, our local GP method also provides more robustness against deviation from stationarity.

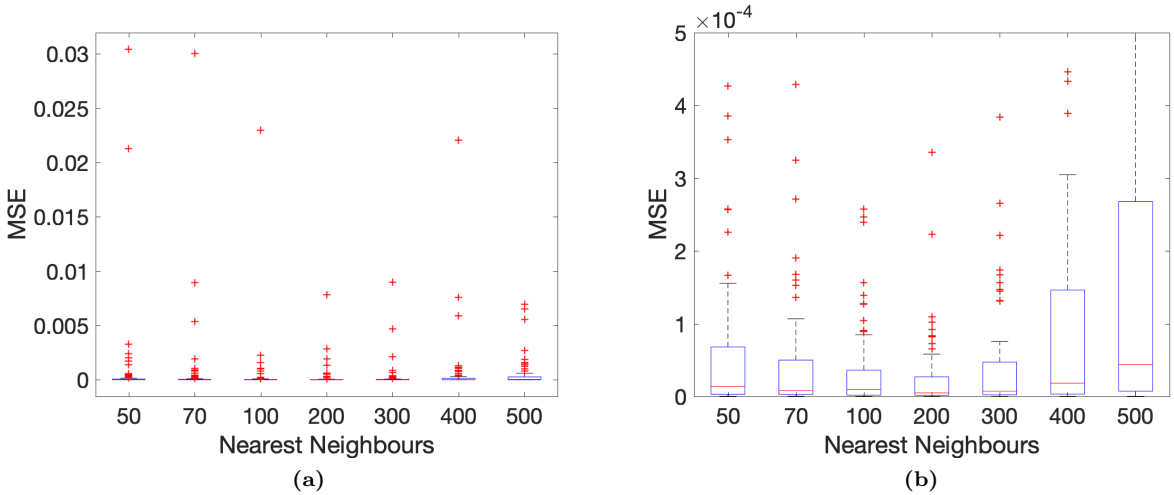


Figure 8: Validating the performance of the local multivariate output Gaussian process. The numerical values correspond to median out-of-sample MSE values in the predicted parameters using a local multivariate emulator as outlined in Section 4.3 with different numbers of nearest neighbours in function space. The plot on the right constrains the y-axis to exclude outliers, allowing us to see the minimum value of MSE occurring around $K = 100$.

Table 2: Estimates for real MRI data. The literature gold standard and the estimated parameters $\hat{\boldsymbol{q}}$ from emulation method M3. The confidence intervals (CIs) are obtained using a parametric bootstrap procedure.

	a	b	a_f	b_f	a_s	b_s	a_fs	b_fs
Literature	0.2245	1.6215	2.4267	1.8269	0.5562	0.7747	0.3905	1.6950
Point estimate (95% CI)	0.2249 (0.2246, 0.2252)	1.6247 (1.6223, 1.6264)	2.4218 (2.4128, 2.4685)	1.8232 (1.8165, 1.8283)	0.5633 (0.5586, 0.5673)	0.7845 (0.7795, 0.7901)	0.3916 (0.3885, 0.3943)	1.6994 (1.6862, 1.7114)

7.4 Performance evaluation on MRI cine data

From Table 1 we found that the best strategy is represented by method M3: Output emulation using a local multivariate output GP. In this section we apply method M3 to estimate the constitutive parameter vector $\boldsymbol{q}$ of (1) from cine MRI data of a healthy volunteer. First, we extracted the LV geometry from the MRI scans, as described in Section 6.2 and illustrated in Figures 1 (segmentation) and 2 (mesh creation). Next, we extracted from the MRI scans the LV chamber volume and 24 circumferential strains at clearly defined positions on the LV wall. These were non-dimensionalized, as described at the beginning of Section 3, to provide the data $\boldsymbol{y}^{\mathrm{obs}}$ which were to be matched by the corresponding predictions from the biomechanical model of Section 2, $\mathbf{m}(\boldsymbol{q})$. The emulation was carried out in the reduced 4D parameter space, which was mapped back into the 8D space via (7). For real data, the true parameters are unknown, so we used the parameters from the literature (Gao et al., 2017) as a benchmark. These parameters were obtained from the same cine MRI data using the iterative optimization procedure described in Section 3.1, solving the soft-tissue mechanical equations from Section 2 by (computationally expensive) brute-force numerical integration using finite element discretization. A comparison between the benchmark parameters and the parameters obtained with our emulator is shown in Table 2. The values are very similar, with a mean square error of only MSE = 0.00003. Four parameters lie outside the 95% confidence intervals obtained with the parametric boostrap procedure described below, which reflects a slight bias resulting from our dimension reduction (7).

One important aspect of the constitutive behaviour of the myocardium can be reflected in the Cauchy stress-stretch curves. Figure 9 shows the myofibre stress-stretch relationship for the healthy volunteer, obtained using the Holzapfel-Ogden law, (1), with the parameters shown in Table 2. The two panels refer to different directions: stretch along the sheet direction, panels (a,c), and along the myocyte, panels (b,d). The black dashed line shows the curve obtained from the literature gold standard method of Gao et al. (2017). The blue solid curve shows the stress-stretch relationship corresponding to the parameter vector $\hat{\boldsymbol{q}}$

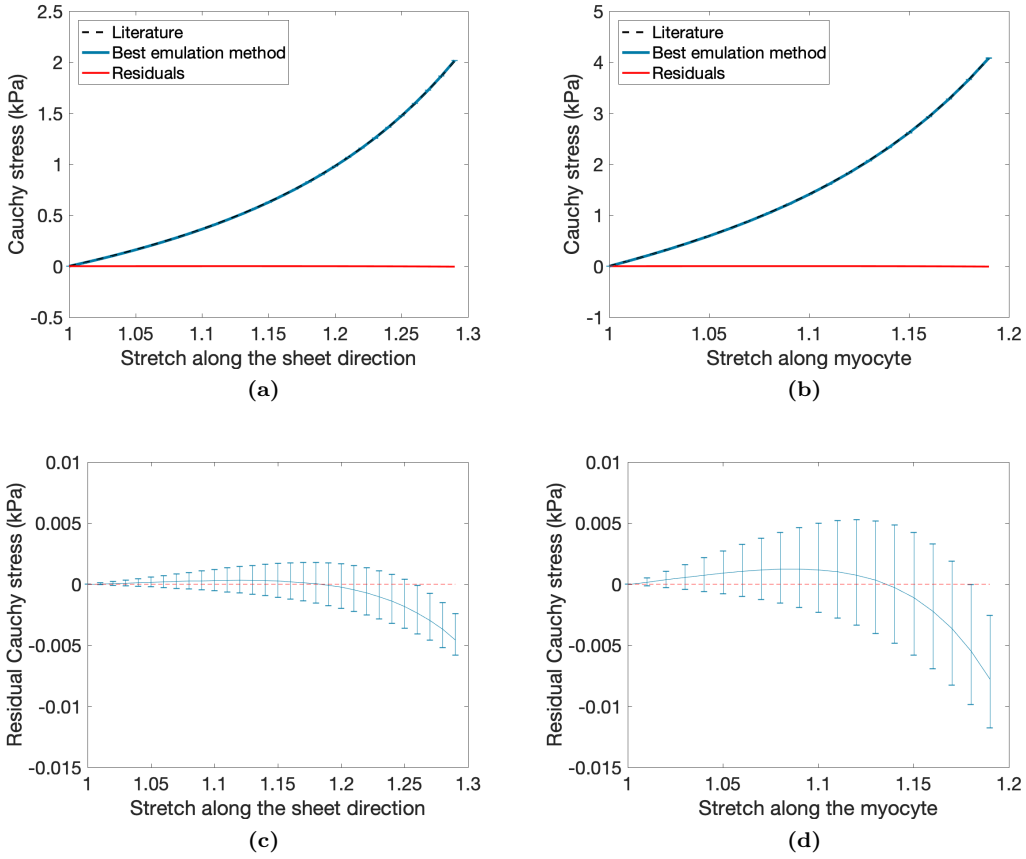


Figure 9: Plots of the Cauchy stress against the stretch along (a) the sheet direction and (b) the myocyte. The current best estimate (i.e. the literature gold standard) from Gao et al. (2017) is reported as a dashed black line. Estimates of the curves using the best emulation approach (M3) are given as a blue solid line. The error bars show a 95% confidence interval, obtained by using the bootstrap method described in Section 7.4. Each plot contains a residual curve, providing the difference between the true and estimated curves. These residual curves are plotted with 95% confidence intervals in (c) for he sheet direction and (d) along the myocyte.

estimated using our emulation method M3. In order to obtain an indication of the uncertainty of our inference, we adopt a bootstrap approach (Efron, 1981; Efron and Tibshirani, 1994). First we obtain a point estimate of the parameters, $\hat{\boldsymbol{q}}$, from the noisy data, $\boldsymbol{y}^{\text{obs}}$, and use it to compute the corresponding output in data space, $\hat{\boldsymbol{y}}$. We then obtain the residuals, $\hat{\boldsymbol{\epsilon}} = \boldsymbol{y}^{\text{obs}} - \hat{\boldsymbol{y}}$ or $\hat{\epsilon}_i = y_i^{\text{obs}} - \hat{y}_i$, for $1 \leq i \leq 25$, randomly sample with replacement from the set of residuals $\mathcal{R} = \{\hat{\epsilon}_i\}_{1 \leq i \leq 25}$, and generate surrogate data $\tilde{y}_i = \hat{y}_i + \tilde{\epsilon}_i$, where $\tilde{\epsilon}_i$ is the ith draw from $\mathcal{R}$. We then repeat the parameter estimation on the surrogate data $\tilde{\boldsymbol{y}}$ to obtain new parameter estimates $\tilde{\boldsymbol{q}}$, and repeat the procedure 100 times, to obtain a distribution of $\tilde{\boldsymbol{q}}$; this is the bootstrap distribution used for uncertainty quantification. For every sampled parameter vector $\tilde{\boldsymbol{q}}$ we compute the Cauchy stress-stretch curve. The results are shown in Figure 9. The figure shows the stress-stretch curve for the estimated parameter vector $\hat{\boldsymbol{q}}$ (in blue) and 95% confidence intervals for the estimated curves obtained from the sample of 100 Cauchy stress-stretch curves corresponding to the bootstrap sample $\{\tilde{\boldsymbol{q}}\}$. The literature curve lies mostly within this confidence interval; the slight bias for large stretches results from the dimension reduction in (7).

As already discussed at various places before, the computational complexity of the numerical procedure required to obtain the gold standard solution in Table 2 and Figure 9 is very high; in the order of over a week when following the procedure described in Gao et al. (2015). This computational complexity renders the method unfit for use in the clinical practice. By contrast, the proposed emulation method has reduced the computational complexity by three orders of magnitude, to less than 15 minutes [10]. This complexity reduction makes the inclusion of biomechanical parameter estimation in a clinical decision support system a viable prospect for future improved prognostication of the risk of myocardial infarction.

[10] Intel Xeon CPU, 2.9GHz, 32 cores and 32GB memory.

8 Discussion

The present article follows up on Gao et al. (2017), in which we have demonstrated the diagnostic power of the constitutive parameters of the Holzapfel-Ogden model of passive myocardium, equation (1), for prognostication of the risk of myocardial infarction. However, the high computational costs of the numerical procedures required for the parameter estimation currently prevent a translation of the mathematical modelling framework into medical practice and meaningful health outcomes. In the present study, we have therefore explored the approach of statistical emulation, whereby the computationally expensive mathematical model is replaced by a statistical surrogate model. We have carried out a comparative evaluation of different emulation strategies, and have demonstrated that a reduction of the computational costs by three orders of magnitude is feasible at negligible loss in accuracy.

In the present proof-of-concept study, we have applied a novel emulation approach to a representative healthy subject. To have clinical translation, we need to be able to apply this approach to a large cohort of patients. In other words, we need to develop an emulation framework that can include variations of subject-specific LV geometries, accompanied by regional heterogeneity in material properties. This could be caused by regional ischemia, regional diffuse fibrosis, and myocardial infarction. To this end we need to find a low-dimensional representation of both the LV geometry and disease features (e.g. infarct size, shape and transmurality) obtained from MRI, as well as enlarge the parameter set to include spatial parameter variations. Sensitivity analysis (Saltelli et al., 2001) and dimension reduction techniques (Murphy, 2012) can be applied to manage the total number of parameters. There are various approaches that one can pursue for geometrical changes, including the 6D parametric representation proposed in Achille et al. (2018), principal component analysis (PCA; see e.g. Chapter 12 in Murphy (2012)), to project the high dimensional LV geometry vector into a low dimensional coordinate system that contains maximum information about typical variations in the patient population, non-linear extensions of PCA, like kernel PCA (Scholkopf et al., 1998), to project the high dimensional LV geometry vector onto low-dimensional non-linear manifolds, or various non-linear dimension reduction techniques from the machine learning community, like self-organising maps or deep neural networks; see e.g. Murphy (2012) for a review. We leave a comparative evaluation of these techniques for future work. It is expected that, as more measurements become available, more patient specific parameters (such as regional variations) can be included in future model extensions. By providing a procedure for fast and computationally efficient inference, the work described in the present paper lays the foundations for their estimation in a time frame that is viable in the clinical practice.

Acknowledgement

This work was funded by the UK Engineering and Physical Sciences Research Council (EPSRC), grant number EP/N014642/1, as part of the SofT-Mech project, and the British Heart Foundation (PG/11/2/28474; PG/14/64/31043; FS/15/54/3169; RE/18/6134217). Umberto Noè was supported by a Biometrika Scholarship. Alan Lazarus is partially funded by a grant from GlaxoSmithKline plc. Benn Macdonald is supported by The Biometrika Trust, Fellowship number B0003. Dirk Husmeier is supported by a grant from the Royal Society of Edinburgh, award number 62335.

References

Achille, P. D., Harouni, A., Khamzin, S., Solovyova, O., Rice, J. J., and Gurev, V. (2018). Gaussian process regressions for inverse problems and parameter searches in models of ventricular mechanics. *Frontiers in Physiology*, 9.

Alvarez, M. A. and Lawrence, N. D. (2011). Computationally efficient convolved multiple output gaussian processes. *Journal of Machine Learning Research*, 12:1459–1500.

Bishop, C. M. (2006). *Pattern Recognition and Machine Learning (Information Science and Statistics)*. Springer-Verlag, Berlin, Heidelberg.

Casella, G. and Berger, R. L. (2002). *Statistical Inference*. Duxbury, Pacific Grove, CA.

Conti, S., Gosling, J. P., Oakley, J. E., and O'Hagan, A. (2009). Gaussian process emulation of dynamic computer codes. *Biometrika*, 96(3):663–676.

Conti, S. and O'Hagan, A. (2010). Bayesian emulation of complex multi-output and dynamic computer models. *Journal of Statistical Planning and Inference*, 140(3):640–651.

Davies, V., Noè, U., Lazarus, A., Gao, H., Macdonald, B., Berry, C., Luo, X., and Husmeier, D. (2019). Fast parameter inference in a biomechanical model of the left ventricle using statistical emulation. *Journal of the Royal Statistical Society, Series C*, Under review.

Dutilleul, P. (1999). The mle algorithm for the matrix normal distribution. *Journal of Statistical Computation and Simulation*, 64(2):105–123.

Efron, B. (1981). Nonparametric estimates of standard error: The jackknife, the bootstrap and other methods. *Biometrika*, 68(3):589–599.

Efron, B. and Tibshirani, R. J. (1994). *An Introduction to the Bootstrap*. Chapman and Hall, Boca Raton.

Fang, K., Li, R., and Sudjianto, A. (2006). *Design and modeling for computer experiments*. Chapman & Hall/CRC.

Gao, H., Aderhold, A., Mangion, K., Luo, X., Husmeier, D., and Berry, C. (2017). Changes and classification in myocardial contractile function in the left ventricle following acute myocardial infarction. *Journal of The Royal Society Interface*, 14(132):20170203.

Gao, H., Allan, A., McComb, C., Luo, X., and Berry, C. (2014a). Left ventricular strain and its pattern estimated from cine cmr and validation with dense. *Physics in Medicine & Biology*, 59(13):3637.

Gao, H., Carrick, D., Berry, C., Griffith, B. E., and Luo, X. (2014b). Dynamic finite-strain modelling of the human left ventricle in health and disease using an immersed boundary-finite element method. *The IMA Journal of Applied Mathematics*, 79(5):978–1010.

Gao, H., Li, W., Cai, L., Berry, C., and Luo, X. (2015). Parameter estimation in a holzapfel–ogden law for healthy myocardium. *Journal of engineering mathematics*, 95(1):231–248.

Gelman, A., Carlin, J., Stern, H., and Rubin, D. (2004). *Bayesian Data Analysis*. Chapman and Hall/CRC, London, 2nd edition.

Gramacy, R. B. and Apley, D. W. (2015). Local Gaussian Process Approximation for Large Computer Experiments. *Journal of Computational and Graphical Statistics*, 24(2):561–578.

Hadjicharalambous, M., Asner, L., Chabiniok, R., Sammut, E., Wong, J., Peressutti, D., Kerfoot, E., King, A., Lee, J., Razavi, R., et al. (2016). Non-invasive model-based assessment of passive left-ventricular myocardial stiffness in healthy subjects and in patients with non-ischemic dilated cardiomyopathy. *Annals of Biomedical Engineering*, pages 1–14.

Hastings, W. (1970). Monte Carlo sampling methods using Markov chains and their applications. *Biometrika*, 57(1):97–109.

Holzapfel, G. A. and Ogden, R. W. (2009). Constitutive modelling of passive myocardium: a structurally based framework for material characterization. *Philosophical Transactions of the Royal Society of London A: Mathematical, Physical and Engineering Sciences*, 367(1902):3445–3475.

Holzapfel, G. A. and Ogden, R. W. (2018). *Multiscale Soft Tissue Mechanics and Mechanobiology*. Springer Verlag.

Jones, D. R., Schonlau, M., and Welch, W. J. (1998). Efficient Global Optimization of Expensive Black-Box Functions. *Journal of Global Optimization*, 13(4):455–492.

Kennedy, M. C. and O'Hagan, A. (2001). Bayesian calibration of computer models. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 63(3):425–464.

LeGrice, I. J., Smaill, B., Chai, L., Edgar, S., Gavin, J., and Hunter, P. J. (1995). Laminar structure of the heart: ventricular myocyte arrangement and connective tissue architecture in the dog. *American Journal of Physiology-Heart and Circulatory Physiology*, 269(2):H571–H582.

Locatelli, M. and Schoen, F. (2013). *Global optimization: theory, algorithms, and applications*. Society for Industrial and Applied Mathematics (SIAM).

Loret, B. and Simoes, F. M. (2016). *Biomechanical Aspect of Soft Tissues*. CRC Press.

Melis, A., Clayton, R. H., and Marzo, A. (2017). Bayesian sensitivity analysis of a 1d vascular model with Gaussian process emulators. *International Journal for Numerical Methods in Biomechanical Engineering*, 33.

Murphy, K. (2012). *Machine Learning: A Probabilistic Perspective*. MIT Press, Cambridge, MA.

Noè, U., Chen, W., Filippone, M., Hill, N., and Husmeier, D. (2017). Inference in a partial differential equations model of pulmonary arterial and venous blood circulation using statistical emulation. In Bracciali, A., Caravagna, G., Gilbert, D., and Tagliaferri, R., editors, *Computational Intelligence Methods for Bioinformatics and Biostatistics. CIBB 2016. Lecture Notes in Computer Science*, volume 10477, pages 184–198, Cham, Switzerland. Springer.

O'Hagan, A. (2006). Bayesian analysis of computer code outputs: A tutorial. *Reliability Engineering & System Safety*, 91(10-11):1290–1300.

Overstall, A. M. and Woods, D. C. (2017). Bayesian design of experiments using approximate coordinate exchange. *Technometrics*, 59:458–470.

Rao, S. S. (2018). *Finite Element Method in Engineering*. Elsevier.

Rasmussen, C. E. and Williams, C. K. I. (2006). *Gaussian processes for machine learning*. MIT Press.

Saltelli, A., Chan, K., and Scott, E. M. (2001). *Sensitivity Analysis*. John Wiley & Sons, Chichester, UK.

Santner, T. J., Williams, B. J., and Notz, W. I. (2003). *The Design and Analysis of Computer Experiments*. Springer Series in Statistics. Springer New York, New York, NY.

Scholkopf, B., Smola, A., and Muller, K.-R. (1998). Nonlinear component analysis as a kernel eigenvalue problem. *Neural Computation*, 10:12991319.

Shahriari, B., Swersky, K., Wang, Z., Adams, R. P., and de Freitas, N. (2016). Taking the human out of the loop: A review of Bayesian optimization. *Proceedings of the IEEE*, 104:148–175.

Snelsen, E. and Ghahramani, Z. (2005). Sparse Gaussian processes using pseudo-inputs. *Proceedings of the 18th International Conference on Neural Information Processing Systems (NIPS)*, 128:1257–1263.

Titsias, M. (2009). Variational learning of inducing variables in sparse Gaussian processes. *Proceedings of Machine Learning Research*, 5:567–574.

Ugray, Z., Lasdon, L., Plummer, J., Glover, F., Kelly, J., and Martí, R. (2007). Scatter Search and Local NLP Solvers: A Multistart Framework for Global Optimization. *INFORMS Journal on Computing*, 19(3):328–340.