University of Glasgow

School of Mathematics and Statistics

Level-5 Honours project

Machine Learning Based Surrogate Modelling in Cardiac Mechanics

Author:
Dhurim Cakiqi
2263325c

Supervisor:
Hao Gao

Declaration of Originality

I confirm that this assignment is my own work and that I have:

- *Read and understood the guidance on plagiarism in the Undergraduate Handbook, including the University of Glasgow Statement on Plagiarism*
- *Clearly referenced, in both the text and the bibliography or references, all sources used in the work*
- *Fully referenced (including page numbers) and used inverted commas for all text quoted from books, journals, web etc.*
- *Provided the sources for all tables, figures, data etc. that are not my own work.*
- *Not made use of the work of any other student(s) past or present without acknowledgement. This includes any of my own work, that has been previously, or concurrently, submitted for assessment, either at this or any other educational institution, including school.*
- *Not sought or used the services of any professional agencies to produce this work*
- *In addition, I understand that any false claim in respect of this work will result in action under the University regulations for Student Conduct*
- *I am aware of and understand the University's policy on plagiarism and I certify that this assignment is my own work, except where indicated by referencing, and that I have followed the good academic practices noted above.*

I also agree that this project can be used by the School of Mathematics and Statistics at University of Glasgow for teaching, recruitment and other aspects of its work.

Monday 1st February, 2021

Abstract

The text for your abstract goes here. Abstracts should be a SHORT but COMPLETE summary of what you have accomplished.

We have considered the stability of flow in flexible-walled channel....

Contents

1 Introduction

The heart is a muscular pump in which it has a main purpose to supply oxygen to meet tissue demands of nutrients and waste removal in all organs of the body [2]. From a 2020 study by the British Heart Foundation 7.4 million people in the UK live with heart and circulatory diseases furthermore heart and circulatory diseases are the cause 170,000 deaths per year in the UK [8].

We can go as far back as 1892 to see when mathematicians were trying to implement methods from continuum mechanics with the publication of Woods' [7] model in which he considered the heart as a spherical model. Since that point in time we have made huge progress with modelling the heart specifically with the work of Holzapfel and Ogden [6] in which their model showed to describe the current data available on shear along the walls of the myocardium. With that we have arrived at the current problem that faces computational cardiology right now and that is the lack of data present in which we are able to create mathematical models that we can be, for a reasonable amount, sure is general enough for most patients. It is incredibly difficult to gather data on the myocardium as to gather any data would require use of an echo-cardiograph which whilst providing a clear image of the heart is incredibly invasive. This is the reason why we have moved away from the finite element method (FE) of modelling the heart to trying to implement machine learning (ML) methods into our models as we can try to predict the parameters that are needed in order to effectively model the heart of a patient. Another fault of FE is the long CPU time required to model 1 second of a cardiac cycle [4]. In more recent years we have seen the rise of surrogate models which seek to emulate the FE model through classical ML training methods for a classification model as well as using packages in Python such as XGBoost [9], as well as using more sophisticated ML methods such as Gaussian Processes [10] (GP) and Deep Learning [11].

Looking back at the data from the British Heart Foundation we can see how important an accurate model of the heart could be to the lives of many millions of patients as we could create a general model. With that patient data could be used to provide a personal model for every person that simulates what would happen to their heart and give us the ability to diagnose people in advance and take precautionary measures to reduce the chance of heart disease. Another limitation we face with FE is the fact that we still have yet to understand the full mechanical underpinnings of cardiac mechanics which leaves a major problem when it comes to trying to model specific patients with the parameters given. As we use our surrogate models we gain more data that seeks to emulate the real world and with the models getting better with training it leads to our mathematical models being updated and becoming better which in turn helps us create a surrogate model. An image below shows how we can use ML and multiscale modelling in conjuncture with each other can help solve the problems we face today in modelling the heart.

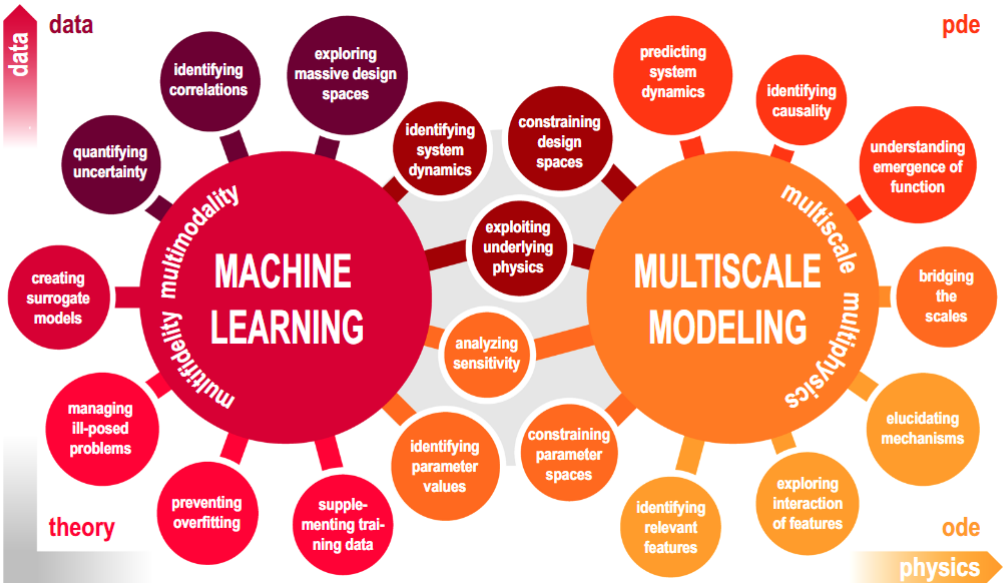


Figure 1: The relation between machine learning and multiscale modeling where they interact when exploiting the underlying physics and analyzing sensitivity. Through machine learning we can create surrogate models with the end goal being to explore massive design spaces and identify correlations. Multiscale modeling will eventually lead us to identifying relevant features for our model and explore their interaction. Adopted from [5].

In Chapter 2 we review the necessary mathematics required to be able to understand the current prevailing models that are used to model the heart, this will then lead onto us taking a look at the Holzapfel-Ogden model [6] which is seen as the most accurate model. We will go into more depth on why FE is so inefficient as a means to model the heart. In Chapter 2 we will study machine learning methods such as XGBoost, Deep learning, GP and Surrogate Modeling. In Chapter 3 we will make an effort to introduce the mathematics used in GP so the reader can better understand the "behind the scenes" of the code that allows us to create these models. Chapter 4 brings us to a very interesting point in this paper where we will look to create our own "toy" model of a surrogate model and use our own parameters to estimate the shear stress of the myocardium and compare it to models from other papers. This then naturally leads us on to the Discussion portion of the paper where we will compare and contrast the different methods used and come up with a conclusion as to which method was the best.

2 Mathematical Background for Cardiac Mechanics

2.1 Structure of he heart

The heart is a four chambered organ with 2 ventricles ,the mitral and tricuspid, along with 2 vales, the right and left as well, which control the direction of blood flow as shown in Figure 2.

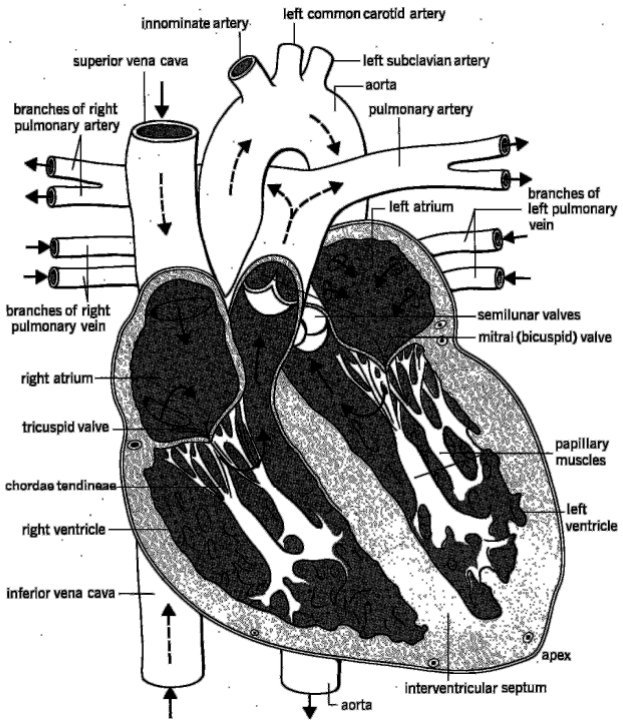


Figure 2: Diagram of the heart showing the blood flow as well as the interior sections, taken from Kogan 2010 [2].

As we can see from the diagram above the heart is split into 4 chambers and divided into two sides, a right and left side, where we have two atria and two ventricles. The function of the atria is to receive blood and to "prime" the blood to be released into the ventricles where they pump blood to the designated location, in between the ventricle and atrium of both sides lies a valve which serves the purpose of controlling the flow of blood from the respective atrium to the ventricle. The left atrium receives oxygenated blood from the lungs, the oxygenated blood then flows into the left ventricle where it is pumped to the rest of the tissues in the body. Deoxygenated blood travels through the vena cava into the right atrium where it then travels into the right ventricle where it is pumped into pulmonary circulation where it goes to the lungs to become oxygenated blood.

We will follow the precedent set by Ogden and Holzapfel [6] in assuming that the heart is a continuum composed of laminar sheets. We focus on the left ventral and from previous studies is shown to be modelled well as a thick-walled ellipsoid of revolution that is truncated at the base. There are three layers to consider that make up the wall of the heart; the inner (endocardium), the middle (myocardium) and the outer (epicardium). The endocardium is of thickness 100μm and contains epimysial collage, elastin and a layer of endothelial cells. The epicardium is of thickness 100μm as well and serves as a protective layer of the heart consists largely of epimysial collagen and some elastin [6].

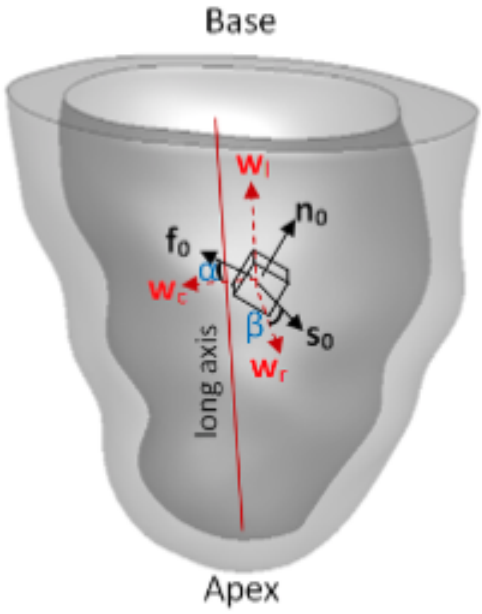


Figure 3: Mathematical drawing of the heart showing how the axis are orientated for the purposes of modelling, adapted from [13]

As we can see from figure 3 when we model the heart in three dimensions we adopt the notation $\mathbf{f}_0$,$\mathbf{n}_0$ and $\mathbf{s}_0$ which represent the fibre, normal and sheet axis respectively. These axis are orthonormal basis vectors. The muscle fibre direction rotates from 50° to 70° in the subendocardium, to 0° in the mid-wall, to approximately -50° to 70° in the subepicardium [6].

2.2 Requirement for Computational Cardiology

In this section we refer to [15] as a good source for different perspectives on the need for computational cardiology. From the viewpoint of a clinician when a patient suffers from myocardial infarction (MI) the use of imaging techniques such as Cardiac magnetic resonance (CMR) is a very useful tool for immediate assessment of risk, however this must be done within 3 days of (MI). Furthermore it is important to state that for patients at greater risk such as those who need a defibrillator is problematic [15]. The use of computational cardiology can lead to more descriptive bio-markers for the classification of patients such as the multi-scale/physics model. This model includes information such biomechanics, blood flow and electrophysiology, these bio-markers can lead to personalised medical treatments for patients.

As stated before the use of computational cardiology can usher in a new era of personalised medicine however this is dependant on us having an anatomically accurate model as well as knowledge of the passive and active material properties of the myocardium[15]. Furthermore to get to the point of real time estimation we need a method that is computationally efficient as well being accurate, this is where (ML) methods are adopted to fill in this gap.

2.3 Review of Continuum Mechanics

To be able to understand the theory of cardiac mechanics it would be wise for us to first lay down the foundations required to grasp the ideas in this widening field. I would recommend the reader to review [12] for a more in depth look at tensor theory as well as the ideas behind deformations as we are only going to review the required identities and definitions.

First we define the deformation gradient

$$\mathbf{F} = \frac{\partial \mathbf{x}}{\partial \mathbf{X}} \tag{2.1}$$

with $\mathbf{x}$ being the current configuration and $\mathbf{X}$ being the reference configuration.We then move onto defining the Jacobian of equation (2.1)

$$J = \det\mathbf{F} > 0. \tag{2.2}$$

If we require that out material be incompressible we define it as the following

$$J = 1. \tag{2.3}$$

We define the Cauchy-Green tensors as

$$\mathbf{C} = \mathbf{F}^T\mathbf{F} \quad \text{and} \quad \mathbf{B} = \mathbf{F}\mathbf{F}^T \tag{2.4}$$

where the left tensor is defined by $\mathbf{C}$ and the right tensor is defined by $\mathbf{B}$. We follow this with a definition of the principal invariants of $\mathbf{C}$ defined as

$$I_1 = \mathrm{tr}(\mathbf{C}), \quad I_2 = \frac{1}{2}[I_1^2 - \mathrm{tr}(\mathbf{C})^2] \quad \text{and} \quad I_3 = \det(\mathbf{C}) \tag{2.5}$$

To have impressibility as required when modelling the heart it is intuitive to the reader that from (2.3) that $I_1 = 1$. We follow the work of Wang et al (2013) [13] and consider the fibre, sheet and fibre-sheet invariants which are defined respectively as,

$$I_{4\mathrm{f}} = \mathbf{f}_0 \cdot (\mathbf{C}\mathbf{f}_0), \quad I_{4\mathrm{s}} = \mathbf{s}_0 \cdot (\mathbf{C}\mathbf{f}_0), \quad I_{8\mathrm{fs}} = \mathbf{f}_0 \cdot (\mathbf{C}\mathbf{s}_0). \tag{2.6}$$

To analyse the stresses that occur on heart we require two important facts from continuum mechanics that being the 2$^{\mathrm{nd}}$ Piola-Kirchoff stress and the Cauchy stress tensors given by

$$\mathbf{S} = 2\frac{\partial W}{\partial \mathbf{C}}, \quad \boldsymbol{\sigma} = \frac{1}{J}\mathbf{F}\mathbf{S}\mathbf{F}^T \tag{2.7}$$

respectively. The W in the equation above is the strain energy function that we will go on to describe in the next section. We will now specify the boundary value problem to be solved in the current configuration as given by [16],

$$\left.\begin{array}{lll} \nabla \cdot \boldsymbol{\sigma} + \mathbf{b} = 0 & \text{in} & \Omega \\ \nabla \cdot \boldsymbol{\sigma} = \mathbf{t} & \text{in} & \Gamma^N \\ \mathbf{u} = \mathbf{u}_0 & \text{in} & \Gamma^D \end{array}\right\} \tag{2.8}$$

where $\mathbf{b}$ is the body force, $\mathbf{n}$ is the normal direction of $\partial\Omega$, $\mathbf{t}$ is the traction and Γ^N, Γ^D are Neumann and Dirichlet boundaries. This BVP is solved by FE via the use of the programming package ABAQUS which simulates the LV during diastolic filling. From [16] they found that simulation via the use of this method without parallelization took about 18 minutes

2.4 The Holzapfel-Ogden Model

Many models have become before the Holzapfel-Ogden model (H-O) that have sought to describe the shear forces present in the myocardial and to model the LV properly.From Holzapfel and Ogden [6] we say that the structure-based strain function is given by

$$\begin{aligned} W = \frac{a}{2b}\exp[b(I_1 - 3)] + \frac{a_f}{2b_f}(\exp[b_f(I_{4f} - 1)^2] - 1) + \frac{a_s}{2b_s}(\exp[b_s(I_{4s} - 1)^2] - 1) \\ + \frac{a_{fs}}{2b_{fs}}(\exp[b_{fs}(I_{8fs})^2] - 1) \end{aligned} \tag{2.9}$$

where a,b,a_i,b_i for $i = f$,s,fs are eight non-negative material parameters. The stress dimensions are described the a parameters where b's are dimensionless quantities. Now we will seek to find the Cauchy stress of the given strain-energy functionz first we seperate W into four components we shall call W_m, W_f, W_s and W_{fs} each given by

$$W_m = \frac{a}{2b}\exp[b(I_1-3)], \quad W_f = \frac{a_f}{2b_f}(\exp[b_f(I_{4f}-1)^2]-1)$$
$$W_s = \frac{a_s}{2b_s}(\exp[b_s(I_{4s}-1)^2]-1), \quad W_{fs} = \frac{a_{fs}}{2b_{fs}}(\exp[b_{fs}(I_{8fs})^2]-1).$$

We will also define each component for the respective Piola-Kirchhoff stress in a similar fashion. For W_m, using (2.7)

$$\frac{\partial W_m}{\partial \mathbf{C}} = \frac{\partial W_m}{\partial I_1}\frac{\partial I_1}{\partial \mathbf{C}}, \quad \left(\text{note that} \quad \frac{\partial I_1}{\partial \mathbf{C}} = \frac{\text{tr}(\mathbf{C})}{\mathbf{C}} = \mathbf{I}\right)$$
$$\implies \frac{\partial W_m}{\partial \mathbf{C}} = \frac{a}{2}e^{b(I_1-3)}\mathbf{I} \implies \mathbf{S}_m = 2\cdot\frac{a}{2}e^{b(I_1-3)}\mathbf{I} = ae^{b(I_1-3)}\mathbf{I}.$$

For W_f and W_s, via symmetry, we have

$$\frac{\partial W_f}{\partial \mathbf{C}} = \frac{\partial W_f}{I_{4f}}\frac{I_{4f}}{\partial \mathbf{C}}, \quad \text{from (2.7) we have that } \frac{\partial I_{4f}}{\partial \mathbf{C}} = \mathbf{f}_0\otimes\mathbf{f}_0$$
$$\text{Thus} \quad \frac{\partial W_f}{\partial \mathbf{C}} = \frac{a_f}{2bf}e^{b_f(I_{4f}-1)^2}\cdot 2b_f(I_{4f}-1)\mathbf{f}_0\otimes\mathbf{f}_0.$$
$$= a_f(I_{4f}-1)e^{b_f(I_{4f}-1)^2}\mathbf{f}_0\otimes\mathbf{f}_0$$
$$\implies \mathbf{S}_f = 2\frac{\partial W_f}{\partial C} = 2a_f(I_{4f}-1)e^{b_f(I_{4f}-1)2}\mathbf{f}_0\otimes\mathbf{f}_0.$$

As said before there is a seen symmetry between the W_f and W_s where to save time we use the same method as above to determine W_s and we thus obtain

$$\mathbf{S}_s = 2a_s(I_{4s}-1)e^{b_s(I_{4s}-1)^2}\mathbf{s}_0\otimes\mathbf{s}_0.$$

We now come to the W_{fs} term which follows.

$$\text{First we use the identity} \quad I_{8fs} = \mathbf{f}_0\cdot(\mathbf{C}\mathbf{s}_0) = \frac{1}{2}(\mathbf{f}_0\cdot(\mathbf{C}\mathbf{s}_0)+\mathbf{s}_0\cdot\mathbf{C}\mathbf{f}_0)$$
$$\implies \frac{\partial I_{8fs}}{\partial \mathbf{C}} = \frac{1}{2}(\mathbf{f}_0\otimes\mathbf{s}_0+\mathbf{s}_0\otimes\mathbf{f}_0).$$
$$\text{Thus} \quad \frac{\partial W_{fs}}{\partial \mathbf{C}} = \frac{\partial W_{fs}}{\partial I_{8fs}}\frac{I_{8fs}}{\partial \mathbf{C}} = \frac{a_{fs}}{2b_{fs}}e^{b_{fs}(I_{8fs})^2}\cdot b_{fs}\cdot 2I_{8fs}\cdot\frac{1}{2}(\mathbf{f}_0\otimes\mathbf{s}_0+\mathbf{s}_0\otimes\mathbf{f}_0)$$
$$= \frac{a_{fs}}{2}e^{b_{fs}(I_{8fs})^2}\cdot I_{8fs}(\mathbf{f}_0\otimes\mathbf{s}_0+\mathbf{s}_0\otimes\mathbf{f}_0)$$
$$\implies \mathbf{S}_{fs} = 2\frac{\partial W_{fs}}{\partial \mathbf{C}} = e^{b_{fs}(I_{8fs})^2}\cdot I_{8fs}(\mathbf{f}_0\otimes\mathbf{s}_0+\mathbf{s}_0\otimes\mathbf{f}_0)$$

Now we collect all of our terms we have obtained with

$$\mathbf{S} = \mathbf{S}_m + \mathbf{S}_f + \mathbf{S}_s + \mathbf{S}_{fs} - P\mathbf{F}^{-1}\mathbf{F}^{-T}$$

where the negative P term is introduced so that we can satisfy the need for the incompressibility requirement and P is the Lagrangian multiplier to be determined from boundary conditions. From (2.7) we have that

$$\boldsymbol{\sigma} = ae^{b(I_1-3)}\mathbf{F}\mathbf{F}^T + 2a_f(I_{4f}-1)e^{b_f(I_{4f}-1)^2}\mathbf{f}\otimes\mathbf{f} + 2a_s(I_{4s}-1)e^{b_s(I_{4s}-1)^2}\mathbf{s}\otimes\mathbf{s} + I_{8fs}e^{b_{fs}(I_{8fs})^2}\ (\mathbf{f}\otimes\mathbf{s}+\mathbf{s}\otimes\mathbf{f}) - P\mathbf{I} \tag{2.10}$$

where $\mathbf{f} = \mathbf{F}\mathbf{f}_0$ and $\mathbf{s} = \mathbf{F}\mathbf{s}_0$, where $J = 1$ as we assume the material is incompressible. With this stress formulation we can now seek to find examples of the pressure in different scenarios.

2.5 Experiments-based stiffness calibration

In our first case we will think about a uni-axial stretch in the $\mathbf{f}$ direction where we say that $\mathbf{f}_0 = [1, 0, 0]$ and assume that the the fibre is compressed so we have that there is no stress contribution in the sheet and normal directions. Furthermore we will neglect the I_{8fs} term in (2.10) for this subsection to obtain

$$\boldsymbol{\sigma} = ae^{b(I_1-3)}\mathbf{F}\mathbf{F}^T + 2a_f(I_{4f}-1)e^{b_f(I_{4f}-1)^2}\mathbf{f}\otimes\mathbf{f} - P\mathbf{I}. \tag{2.11}$$

With this uni-axial stretch we have that

$$\mathbf{F} = \begin{bmatrix} \lambda & 0 & 0 \\ 0 & \frac{1}{\sqrt{\lambda}} & 0 \\ 0 & 0 & \frac{1}{\sqrt{\lambda}} \end{bmatrix}$$

where it can be easily verified that $\det(\mathbf{F}) = 1$. Furthermore we have that

$$\mathbf{F}\mathbf{F}^T = \begin{bmatrix} \lambda^2 & 0 & 0 \\ 0 & \frac{1}{\lambda} & 0 \\ 0 & 0 & \frac{1}{\lambda} \end{bmatrix}.$$

From (2.6) and (2.4) it follows that $I_{4f} = \lambda^2$. Now we substitute these values above into (2.11) to obtain

$$\boldsymbol{\sigma} = ae^{b(I_1-3)}\begin{bmatrix} \lambda^2 & 0 & 0 \\ 0 & \frac{1}{\lambda} & 0 \\ 0 & 0 & \frac{1}{\lambda} \end{bmatrix} + 2a_f(I_{4f}-1)e^{b_f(I_{4f}-1)^2}\begin{bmatrix} \lambda^2 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} - P\mathbf{I}$$

Now we separate the above into three equations

$$\begin{aligned} \sigma_{11} = \sigma_{ff} &= a\lambda^2 e^{b(I_1-3)} + 2a_f(I_{4f}-1)e^{b_f(I_{4f}-1)^2} - P \\ \sigma_{22} = \sigma_{ss} &= \frac{a}{\lambda}e^{b(I_1-3)} - P = 0 \\ \sigma_{33} = \sigma_{nn} &= \frac{a}{\lambda}e^{b(I_1-3)} - P = 0. \end{aligned}$$

So from the ss and nn stress components we see that

$$P = \frac{a}{\lambda}e^{b(I_1-3)}$$

2.5.1 Simple Shear

We look into the shear forces present on the model and use the same assumption made in (2.11) and from [6] we define the following

$$\begin{aligned} \mathbf{f}_0 &= [1, 0, 0] \\ \mathbf{s}_0 &= [0, 1, 0] \\ \mathbf{n}_0 &= [0, 0, 1]. \end{aligned} \tag{2.12}$$

2.5.2 Shear in fs plane

The deformation gradient in the $\mathbf{f}_0$ direction is defined to be

$$\mathbf{F} = \begin{bmatrix} 1 & \lambda & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}.$$

It can be verified that for shear in the $\mathbf{f}_0$ direction

$$\mathbf{F}\mathbf{F}^T = \begin{bmatrix} 1+\lambda^2 & \lambda & 0 \\ \lambda & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

where $\mathbf{f} = [1, 0, 0]$, $\mathbf{s} = [\lambda, 1, 0]$ and $\mathbf{n} = [0, 0, 1]$. From (2.11) we have the following

$$\boldsymbol{\sigma} = ae^{b(I_1-3)} \begin{bmatrix} 1+\lambda^2 & \lambda & 0 \\ \lambda & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} + 2a_f(I_{4f}-1)e^{b_f(I_{4f}-1)^2} \begin{bmatrix} 1 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} + 2a_s(I_{4s}-1)e^{b_s(I_{4s}-1)^2} \begin{bmatrix} \lambda^2 & \lambda & 0 \\ \lambda & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix} - P\mathbf{I}.$$

We separate this into three equations as before

$$\begin{aligned} \sigma_{11} = \sigma_{ff} &= a(1+\lambda^2)e^{b(I_1-3)} + 2\lambda^2 a_f(I_{4f}-1)e^{b_f(I_{4f}-1)^2} - P \\ \sigma_{22} = \sigma_{ss} &= ae^{b(I_1-3)} + 2a_s(I_{4s}-1)e^{b_s(I_{4s}-1)^2} - P \\ \sigma_{33} = \sigma_{nn} &= ae^{b(I_1-3)} - P = 0. \end{aligned}$$

It can be seen that

$$P = ae^{b(I_1-3)}.$$

In the $\mathbf{s}_0$ direction it can be verified by the reader that the pressure is the same as above with the same calculation.

2.5.3 Shear in the sn plane

The deformation gradient in the s_0 direction is

$$\mathbf{F} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & \lambda \\ 0 & 0 & 1 \end{bmatrix}.$$

We can then see that

$$\mathbf{F}\mathbf{F}^T = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1+\lambda^2 & \lambda \\ 0 & \lambda & 1 \end{bmatrix}$$

where $\mathbf{f} = [1, 0, 0]$, $\mathbf{s} = [0, 1, 0]$ and $\mathbf{n} = [0, \lambda, 1]$. Using (2.11) we obtain

$$\boldsymbol{\sigma} = ae^{b(I_1-3)} \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1+\lambda^2 & \lambda \\ 0 & \lambda & 1 \end{bmatrix} + 2a_f(I_{4f}-1)e^{b_f(I_{4f}-1)^2} \begin{bmatrix} 1 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} + 2a_s(I_{4s}-1)e^{b_s(I_{4s}-1)^2} \begin{bmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix} - P\mathbf{I}.$$

Separating into the 3 equations again

$$\sigma_{11} = \sigma_{ff} = ae^{b(I_1-3)} + a_f(I_{4f}-1)e^{b_f(I_{4f}-1)^2} - P = 0$$
$$\sigma_{22} = \sigma_{ss} = (1+\lambda^2)ae^{b(I_1-3)} + a_s(I_{4s}-1)e^{b_s(I_{4s}-1)^2} - P$$
$$\sigma_{33} = \sigma_{nn} = ae^{b(I_1-3)} - P.$$

So we can see that in this case we have that

$$P = ae^{b(I_1-3)} + a_f(I_{4f}-1)e^{b_f(I_{4f}-1)^2}$$

and once again the pressure is the same in the n_0 direction with similar analysis.

2.5.4 Shear in the fn plane

The deformation gradient in the f_0 direction is

$$\mathbf{F} = \begin{bmatrix} 1 & 0 & \lambda \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}.$$

Where it can be seen that

$$\mathbf{F}\mathbf{F}^T = \begin{bmatrix} 1+\lambda^2 & 0 & \lambda \\ 0 & 1 & 0 \\ \lambda & 0 & 1 \end{bmatrix}$$

where $\mathbf{f} = [1,0,0]$, $\mathbf{s} = [0,1,0]$ and $\mathbf{n} = [\lambda,0,1]$. Using (2.11) we obtain

$$\boldsymbol{\sigma} = ae^{b(I_1-3)} \begin{bmatrix} 1+\lambda^2 & 0 & \lambda \\ 0 & 1 & 0 \\ \lambda & 0 & 1 \end{bmatrix} + 2a_f(I_{4f}-1)e^{b_f(I_{4f}-1)^2} \begin{bmatrix} 1 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} + 2a_s(I_{4s}-1)e^{b_s(I_{4s}-1)^2} \begin{bmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix} - P\mathbf{I}.$$

Separating into the 3 equations again

$$\sigma_{11} = \sigma_{ff} = (1+\lambda^2)ae^{b(I_1-3)} + a_f(I_{4f}-1)e^{b_f(I_{4f}-1)^2} - P$$
$$\sigma_{22} = \sigma_{ss} = ae^{b(I_1-3)} + a_s(I_{4s}-1)e^{b_s(I_{4s}-1)^2} - P = 0$$
$$\sigma_{33} = \sigma_{nn} = ae^{b(I_1-3)} - P.$$

So we see that

$$P = ae^{b(I_1-3)} + a_s(I_{4s}-1)e^{b_s(I_{4s}-1)^2}$$

which is the same when looking at the $\mathbf{n}_0$ direction through the same analysis.

3 Surrogate Modeling

3.1 Motivation

In this section we enter the main idea that is encapsulated by this project and that is replacing the FE methods that have come before and replacing it with Surrogate modeling. The FE method has provided us with accurate models of the heart, however the main drawback of FE comes down to the fact that when we try to use it to give us a real time image of the heart it takes about 100 hours to model 1 second of a cardiac cycle [4]. The main reason for these lengthy CPU times is due to the large matrices that contain multiple parameters which adds to the calculation time. This is far too long if we are seeking to give real time data to patients in a clinical setting, when looking at the recent studies of modelling the left ventricle using Surrogate models we see that these CPU times come down to seconds to model the same duration of a cardiac cycle. Furthermore, a major drawback of FE is trying to find relevant data in which we can use in our models as trying to obtain said data requires invasive methods on patients. One major benefit of surrogate modelling is that it can circumvent this issue and via use of ML methods seeks to emulate the data that we currently have to then train its self to create better models. This method has shown promising results such as in [14] and [9] where they showed that the procedure of collecting in person data can be quickened once the models are trained. Using machine learning methods rather than the standard FE methods that have been used before can lead to us making new discoveries relating to the underlying physics of the heart, this is due to the fact that large scale ML finds correlations between data that we ourselves would find very difficult to see. An enlightening example of this can be seen in saber-metrics, the name given to use of statistical analysis in baseball, before advanced statistical techniques were used many of the scouts in baseball believed that the batting average was a determining factor to creating a strong team. This was an incorrect assumption, when advanced statistical tools were used there was a high correlation between the number of wins a team had and the percentage chance that players were able to score a point by "getting on base", thus changing the understanding that scouts had of baseball.

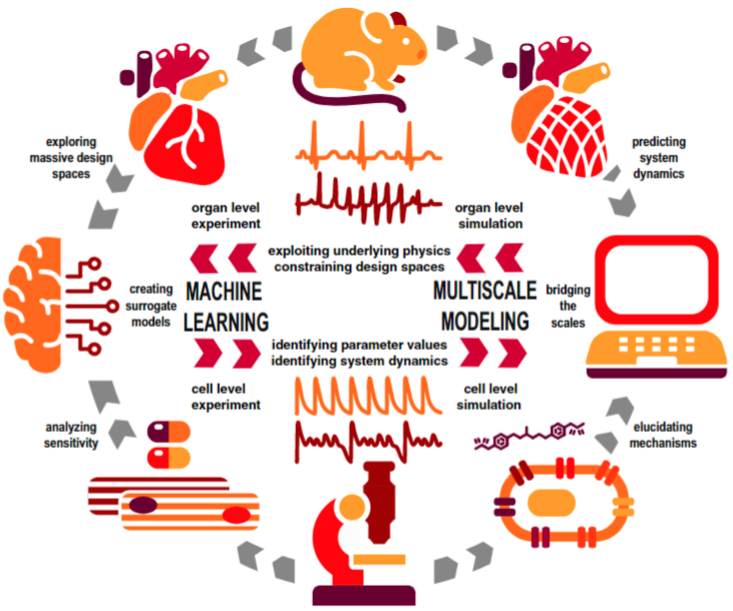


Figure 4: Diagram adopted from [5] showing us how we can use multiscale models and machine learning in tandem to create a cycle that produces models that become better through ML methods

3.2 How Surrogate Modelling works

It would be wise to give the reader a description of how this form of modelling works here we will give a conceptual idea first and then dive deeper into the mathematics. A good resource for the reader is [17] to gain some insight to other applications as well as the further programming in this topic.

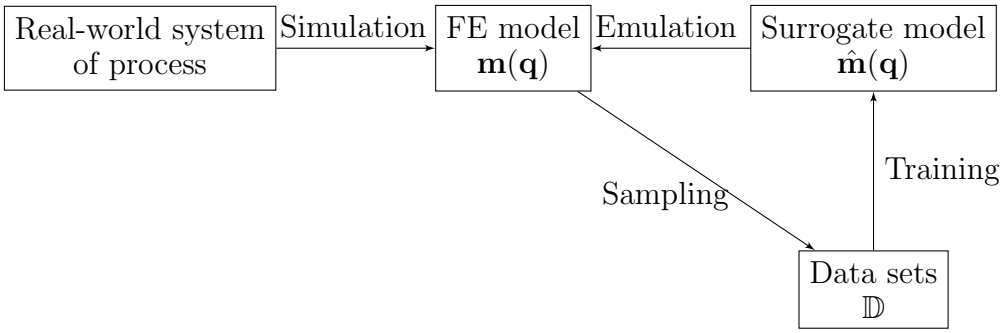


Figure 5: Diagram adapted from [9] where it pictorially describes how we train a surrogate model in order to feed into the (FE) model in order to create better samples for data sets and so on

As we can see from figure 5 which describes surrogate modeling in a general sense, to guide the reader we will relate each stage to a cardiac modelling. So as we can see in the first node, we have a real world system, this can be represented by the myocardium which we seek to model. Through our constitutive models we have we can use (FE) methods to create a working model that represents the passive response of the myocardium of the LV. From this (FE) model we use programming packages such as ABAQUS to solve this model subject to the boundary conditions specified in (2.8) to get data sets of the stress and strain etc. Now from these data sets we train our surrogate model via an ML method of choosing to then be able to inverse the problem and understand the correlation of stress and strain and the material parameters from the given constitutive law, in this case it would be (2.9). The ability to inverse will allow use to gain more data for the (FE) model to then help us train the surrogate model further.

Following from Noè (2018) [24] as a guide we now introduce the mathematics involved in surrogate modelling. First we define the idea of a loss function which is a statistical tool we use for parameter estimation, to optimize this we seek to minimize the loss function. We denote our parameters by $\mathbf{q} \in Q \subset \mathbb{R}^d$, the simulator is denoted by $\mathbf{m}$ which takes the parameters and outputs some data. Let $\mathbf{y}^{obs}$ be our observed data generated by a model we shall call $\mathbf{m(q)}$, we want to find some $\hat{\mathbf{q}}$ such that it gives us a prediction $\mathbf{m}(\hat{\mathbf{q}})$ that is as close as possible to $\mathbf{q}$. The *target loss* is then given by

$$l_m(\mathbf{q}) = d(m(\mathbf{q}, \mathbf{y}^{obs})^2 \tag{3.1}$$

where d is the metric distance function, so it follows that

$$l_m(\mathbf{q}) > 0.$$

Remember that $\hat{\mathbf{q}}$ minimizes the loss function and define it as

$$\hat{\mathbf{q}} = \arg\min_{\mathbf{q} \in Q}(l_m(\mathbf{q})). \tag{3.2}$$

If we had two observed values $\mathbf{y}_i, \mathbf{y}_j$, we define the Euclidean distance by

$$d_2(\mathbf{y}_i, \mathbf{y}_j) = \left[\sum_{t=1}^{k} (y_{it} - y_{jt})^2 \right]^{1/2} \tag{3.3}$$

When we evaluate the objective function (3.1) it involves a costly forward simulation of $m(\mathbf{q})$ [24]. To reduce the computational cost of these evaluations we introduce a surrogate model denoted by $\hat{\mathbf{m}}$ which is a statistical approximation of our original model $\mathbf{m}$ based on a set defined by

$$D = \{\mathbf{q}_i, \mathbf{y}i = m(\mathbf{q}_i)\}_{i=1}^{n}. \tag{3.4}$$

Anytime we simulate data at a point not previously done by $\mathbf{m}(\mathbf{q})$, we replace the simulation by the much faster surrogate $\hat{\mathbf{m}}(\mathbf{q})$. Different methods of replacing the original simulation with the surrogate are given in detail in [24], to summarise the point made, the problem with emulating the output as we have described is made with two points namely; (1) "the multivariate emulator $\hat{\mathbf{m}}$ requires fitting k independent emulators $\hat{\mathbf{m}}_j$". (2)"each evaluation of $l_{\hat{m}}$ involves predicting from k univariate emulator". Where $l_{\hat{m}}$ is the *surrogate-based loss* function given by

$$l_{\hat{m}} = d(\hat{m}(\mathbf{q}), \mathbf{y}^{obs})^2 \tag{3.5}$$

where d is some arbitrary metric. So we need something that is less computationally expensive than output emulation, this brings us to loss-emulation in which we emulate (3.1) and replace (3.2) with the estimate

$$\hat{\mathbf{q}} = \arg\min_{\mathbf{q}\in Q}(\hat{l}_m(\mathbf{q})). \tag{3.6}$$

The reason why this is quicker than output emulation is that we reduce the training complexity, as the vector $\mathbf{y}$,which can be an element of a very high dimensional vector space, is replaced by the scalar $l_{\hat{m}}(\mathbf{q})$. We can represent this reduction in dimension with the mapping

$$\mathbf{m}(\mathbf{q}_i) \in \mathbb{R}^k \to l_m(\mathbf{q}_i) \in \mathbb{R}. \tag{3.7}$$

However we make an important note, that there is a distinct disadvantage to using loss-emulation, that being "the emulator can only be trained after the patient has come into the clinic and the training data is made available" [16]. We can define the process of these forms of emulation more rigorously with the aid of [10].

Algorithm 1: Emulator based on the outputs algorithm. Let $\mathbf{m}$ be our simulator as defined before with parameters $\mathbf{q}_1, \cdots, \mathbf{q}_n$, let $\hat{\mathbf{m}} = (\hat{m}_1, \cdots, \hat{m}_j)$ be the emulator as defined before and we let $\mathbf{y}^{obs}$ be the the observed data from the simulator. Define l, the loss function, by equation (3.1).

Simulate from $\mathbf{m}(\mathbf{q}_1, \cdots \mathbf{q}_n)$;
for *j from* 1 *to* J **do**
  Fit J independent real-valued emulators $\hat{\mathbf{m}}$
  Given $\mathbf{y}_0$ and $\hat{\mathbf{m}}$, construct l
  Minimize l to give the estimates $\mathbf{q}_0$
end

Algorithm 2: Emulator based on the loss algorithm. Let $\mathbf{m}$ be our simulator as defined before with parameters $\mathbf{q}_1, \cdots, \mathbf{q}_n$, let $\hat{\mathbf{m}} = (\hat{m}_1, \cdots, \hat{m}_j)$ be the emulator as defined before and we let $\mathbf{y}^{obs}$ be the the observed data from the simulator. Define $\hat{l}$ by equation (3.5)

Simulate from $\mathbf{m}(\mathbf{q}_1, \cdots \mathbf{q}_n)$;
for *n from* 1 *to* N **do**
  Calculate l between each individual simulation and $\mathbf{y}^{obs}$
  Emulate the losses by using $\hat{l}$
  Estimate $\mathbf{q}_0$ by minimizing the mean of $\hat{l}$
end

3.3 Fundamentals of Machine Learning

In this subsection we seek to elucidate the ideas of machine learning, we will assume the mathematics required to understand this topic is known by the reader a great resource to getting an understanding of the mathematics would be [18] and [19] where they goes through the basic linear algebra, calculus and probability and information theory required. Here we will go through the definitions of the ML methods that we will be using as well as giving examples to further simplify the explanation.

It would be wise for us to first start with explaining what ML is in a conceptual way. In the most general sense ML is using data we have currently to learn about some function that we don't know about to then make predictions on data we don't have yet. We now define the idea of supervised learning as it is the method of ML that we will be implementing in our work.

Definition 3.1 (Supervised learning)**.** The process of observing some random vector $\mathbf{x}$ (we can see this as the input) and an associated vector $\mathbf{y}$ (which is the output), then learning to predict $\mathbf{y}$ from $\mathbf{x}$ [18].

Since the data that we are collecting is continuous it follows that we choose to use regression modelling, to explain regression modeling in simpler terms a short example is given below.

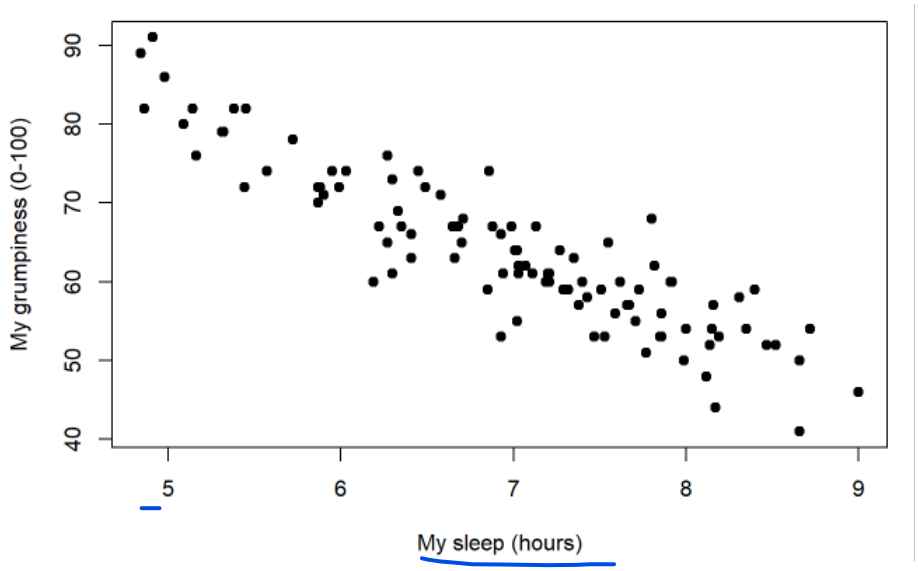


Figure 6: A plot of data graphing an arbitrary index of "grumpiness" to my sleep hours

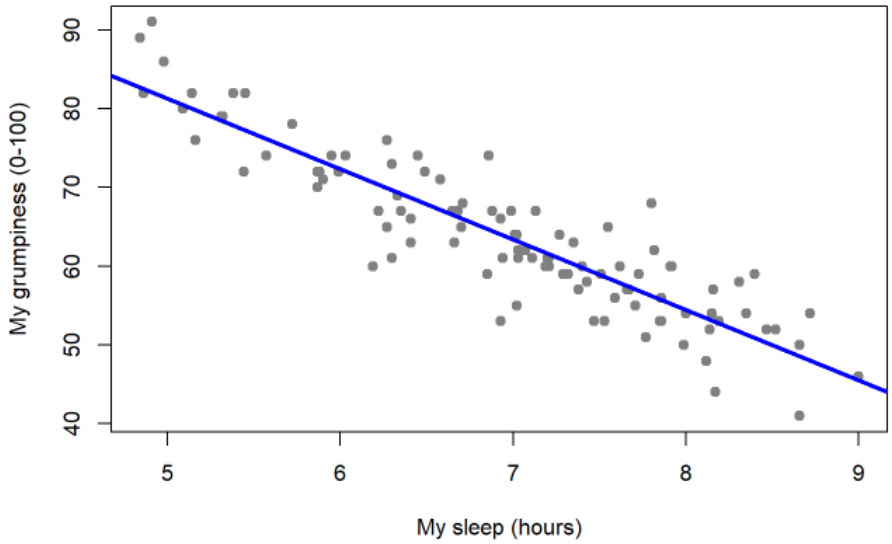


Figure 7: A best fit line plotted onto the data from figure 6, showcasing a very simplistic case of linear regression

As we can see from figures 6 and 7 the basic idea of linear regression is that we are looking for a line of best fit of sorts in order to predict data that we may not have, of course this being a simple example data we see in the real world is rarely ever this nice. Mathematically speaking our goal in linear regression is to take some vector $\mathbf{x} \in \mathbb{R}^n$ and predict some scale $y \in \mathbb{R}$. We set $\hat{y}$ to be the value that our model predicts y. Hence we now define $\hat{y}$ as

$$\hat{y} = \mathbf{w}^T\mathbf{x} + \epsilon \tag{3.8}$$

where $\mathbf{w} \in \mathbb{R}^n$ is a vector of parameters, commonly referred to as the *weight* in the literature, and ϵ is the "noise" in the data [19] assumed to be Gaussian distributed by $\epsilon \sim \mathcal{GP}(0, \sigma^2)$. We can understand parameters as a set of values that we use to control the behaviour of our model. We need parameters in order to make predictions on data we don't have. Now, we need to be able to understand how accurate our model is, this brings us onto the idea of mean squared error (MSE). We define the MSE by

$$\text{MSE} = \frac{1}{m}(\hat{\mathbf{y}}_i - \mathbf{y}_i)^2. \tag{3.9}$$

It follows from inspection that as $y_i \to \hat{y}_i$ gives $\text{MSE} = 0$, our goal when creating a machine learning model is trying to get as close to this value for the MSE. Furthermore for a value of $i = 2$ we get

$$\text{MSE} = \frac{1}{m}||\hat{\mathbf{y}}_2 - \mathbf{y}_2||^2 = \frac{1}{m}||\hat{\mathbf{y}} - \mathbf{y}||_2^2 \tag{3.10}$$

which is congruent to the Euclidean distance metric. When creating an algorithm it needs to improve the weights $\mathbf{w}$ such that it reduces the MSE when the algorithm learns from a training set we will denote by $(\mathbf{X}^{train}, \mathbf{y}^{train})$. We can minimize the MSE via simple calculus techniques, i.e. solving for $\mathbf{w}$ when $\nabla\text{MSE}_{train} = 0$, where we are taking the gradient with respect to $\mathbf{w}$,

$$\begin{aligned}
\nabla\text{MSE}_{train} = 0 &= \nabla\frac{1}{m}||\hat{\mathbf{y}}^{train} - \mathbf{y}^{train}||_2^2 \\
&= \frac{1}{m}\nabla||\mathbf{X}^{train}\mathbf{w} - \mathbf{y}^{train}||_2^2 \\
&= \nabla(\mathbf{X}^{train}\mathbf{w} - \mathbf{y}^{train})^T(\mathbf{X}^{train}\mathbf{w} - \mathbf{y}^{train}) \\
&= \nabla(\mathbf{w}^T\mathbf{X}^{(train)T}\mathbf{X}^{train}\mathbf{w} - 2\mathbf{w}^T\mathbf{X}^{(train)T}\mathbf{y}^{train} + \mathbf{y}^{(train)T}\mathbf{y}^{train}) \\
&= 2\mathbf{X}^{(train)T}\mathbf{X}^{train}\mathbf{w} - 2\mathbf{X}^{(train)T}\mathbf{y}^{train}.
\end{aligned}$$

When we rearrange for $\mathbf{w}$ we obtain

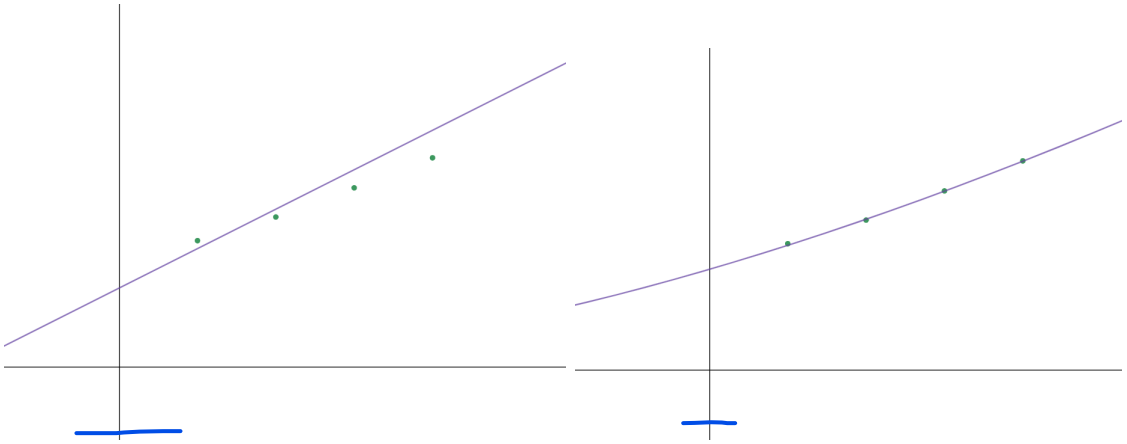
$$\mathbf{w} = (\mathbf{X}^{(train)T}\mathbf{X}^{train})^{-1}\mathbf{X}^{(train)T}\mathbf{y}^{train}, \tag{3.11}$$

(3.11) is called the *normal equations* and solving for $\mathbf{w}$ involves creating a simple learning algorithm. When creating our model we have two data sets that we focus on, that being the training set and the test set. We give two colloquial definitions of these sets as,

Definition 3.2 (Training set)**.** The set we use to "fine tune" our parameters $\mathbf{w}$

Definition 3.3 (Test set)**.** The set that follows the same probability distribution of the training set that we use to see how accurate our training model was.

Now when we train our ML model we must be careful not to overfit or underfit the model, the idea that we don't want our model to predict the noise in the data. Underfitting a model occurs when the MSE of the training set ($\text{MSE}_{(train)}$) is not small enough and overfitting occurs when the difference of the of $\text{MSE}_{(train)}$ and $\text{MSE}_{(test)}$ is too large. To limit these factors we introduce the idea of capacity, a models capacity can be seen as its ability to fit a wide variety of functions [18]. We must take care not to have a low capacity as it will not be able to fit the training set, a high capacity will memorise properties of the training set and cause the model to overfit.



(a) Underfitting (b) Correct capacity

Figure 8: Two examples of us trying to fit a hypothetical data set in which we have underfitted it in (a) and have correctly fitted the data in (b).

This now takes us to the idea of a *validation set* which we define below.

Definition 3.4 (Validation set)**.** The set created by splitting the training set into two parts one being the training set and the second being the validation, this validation set evaluates the performance of the training set and how well the proposed model fits the data.

When we make this division of training and test sets it is possible to have a test set that is too small, this causes a statistical uncertainty around the estimated average test error [18]. This is very prevalent when using a small data set such as the 1000 parameters that we will be simulating in this study, we use *cross-validation* specifically *k-fold cross-validation* as shown in algorithm 3.1 as defined from [18].

Algorithm 3: k-fold cross-validation algorithm. Let A be the learning algorithm, let the given dataset be called $\mathbb{D}$ and let the mean loss be denoted L. We define an element in $\mathbb{D}$ by $\mathbf{a}^i = (\mathbf{x}^i, y^i)$ where $\mathbf{x}^i$ is the input and y^i is the output. A then returns the error vector $\mathbf{e}$ for each element in $\mathbb{D}$. We define the number of folds for our cross-validation by k.

Define KFoldXV($\mathbb{D}, A, L, K$);
Split $\mathbb{D}$ into k mutually exclusive subsets $\mathbb{D}_i$;
for *i from 1 to k* **do**
  $f_i = A(\mathbb{D} \backslash \mathbb{D}_i)$ **for** $\boldsymbol{a}^j \in \mathbb{D}_i$ **do**
    $e_j = L(f_i, \mathbf{a}^j)$
  end
end
return $\boldsymbol{e}$

3.4 Gaussian Processes

3.4.1 An Introduction to Bayesian Statistics

Before we jump into the Gaussian process it would be wise to set up the necessary background that being Bayesian statistics. We will not go into great detail and keep to the necessary ideas required so that the set up for the Gaussian process is smoother. For an application of Bayes to machine learning a great resource is Kevin Murphy (2012) [19]. One might wonder why we want to use Bayesian and this is answered by the fact that if we consider some value we seek to estimate, we shall denote by $\boldsymbol{\theta}$. In Bayesian statistics we use Bayes' theorem,

$$P(A|B) = \frac{P(B|A)P(A)}{P(B} \quad \text{for two events } A,B,$$

to update the probability for a given hypothesis as new information is gained. In other words this method of inference gives tells us how accurate our model is given the available data. Now to represent our hypothesis we use something called a *prior distribution* $p(\boldsymbol{\theta})$, which for brevity we call simply the prior. We denote what we know about the unknown $\boldsymbol{\theta}$ by the posterior $p(\boldsymbol{\theta})$. We define the posterior more rigorously for some estimate $\boldsymbol{\theta}$ by

$$p(\boldsymbol{\theta}|data) = \frac{p(data|\boldsymbol{\theta}) \times p(\boldsymbol{\theta})}{p(data)}. \tag{3.12}$$

The reader may wonder how one goes about defining a prior that is accurate to the data, in this writing we choose to skip that part as we only focus purely on describing Bayesian statistics, a good source would be [25], [19]. The advantages of using this form of statistics compared to classical statistical methods, such as using t-tests, are the following. Classical statistics makes predictions using point estimates of $\boldsymbol{\theta}$ where as Bayesian uses the full distribution of $\boldsymbol{\theta}$. Classical methods use a confidence interval to tell us if a hypothesis is correct or not where as Bayesian uses probabilities that constantly update to tell us how correct our model is.

3.4.2 Introduction to Gaussian Processes

Definition 3.5 (Gaussian Process)**.** A collection of random variables, any finite number of which have a joint Gaussian distribution [3].

We now introduce one of the main methods of supervised learning used in surrogate modelling that being the Gaussian Process. In essence GP follows these basic steps that will be discussed in more detail;

- Come up with a prior distribution over functions
- Observe some data
- Come up with a posterior distribution over functions
- Sample from that posterior distribution to get predictions for our unobserved data.

We will elucidate these points further, but first we discuss why we may use GP, remember from before in linear regression how we had the relation $\hat{y} = \mathbf{w}^T\mathbf{x} + \epsilon$, well in that example we constantly tune our model by finding parameters that best fit our training data. In GP the non-parametric approach is used, whilst this may sound like it means there are no parameters instead it refers to the fact that we use an infinite number of parameters. Furthermore the benefit to using GP is that we get a distribution of all the possible functions that our model could be that is Gaussian/Normally distributed. A nice example would be to consider a case where we have some data as shown below.

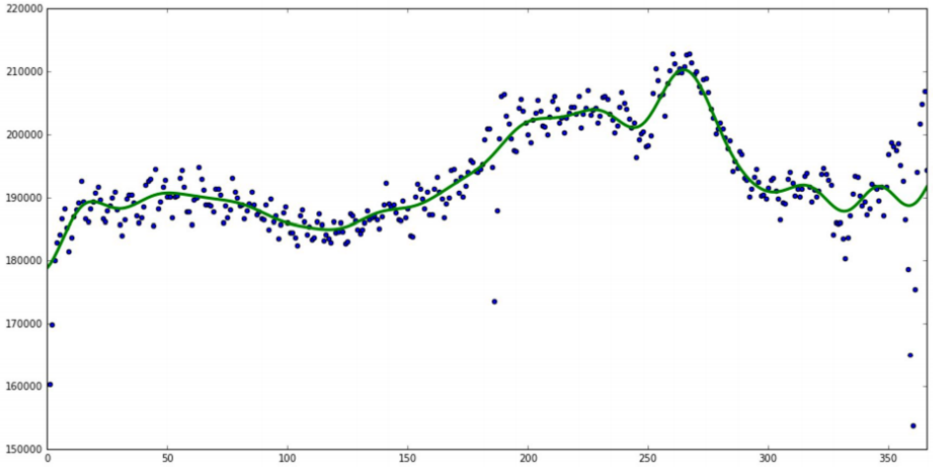


Figure 9: After using the Gaussian Process we can see that the model fits the data very well and without over/underfitting it

Now we have the model that represents our data and we can use it to predict values we don't have. We discussed before what the stages of GP are, now we discuss these points further. The idea of a prior is what we think our data will look like given some assumptions we make, the posterior is the probability distribution that represents our updated beliefs about the parameters after seeing the data. We can describe the posterior by the prior with the identity

$$\text{posterior} \propto \text{prior} \times \text{likelihood}.$$

3.4.3 Mathematical Background for Gaussian Process'

The previous subsection was used to help the reader understand the conceptual idea of what the Gaussian Process is, in this subsection we seek to introduce the statistics required to understand GP. We first define the mean and covariance functions of a function $f(\mathbf{x})$ given respectively by

$$m(\mathbf{x}) = \mathbb{E}[f(\mathbf{x})] \tag{3.13}$$

$$k(\mathbf{x}, \mathbf{x}') = \mathbb{E}[(f(\mathbf{x}) - m(\mathbf{x}))(f(\mathbf{x}') - m(\mathbf{x}'))] \tag{3.14}$$

where $\mathbb{E}$ is the expectation. The covariance function tells us how much two values (functions in this case) are linearly related to each other. We define a Gaussian Process as

$$f(\mathbf{x}) \sim \mathcal{GP}(m(\mathbf{x}), k(\mathbf{x}, \mathbf{x}')). \tag{3.15}$$

The Gaussian distribution is given by the probability density function

$$p(f(\mathbf{x})|\mathbf{m}, k) = (2\pi)^{D/2}|k|^{-1/2}\exp[-\frac{1}{2}(\mathbf{x}-\mathbf{m})^T k^{-1}(\mathbf{x}-\mathbf{m})], \tag{3.16}$$

where equation (3.5) is the shorthand that we adopt, D is the length of the mean vector $\mathbf{m}$ and k is the covariance matrix of size $D \times D$ as defined above. As is standard in the literature and for our purposes we will sample the covariance function using the squared exponential kernel, which satisfies the requirement for all kernels that being it is positive definite, defined as

$$k(x, x') = \sigma^2 \exp\left(-\frac{1}{2l}(x-x')^2\right). \tag{3.17}$$

Where σ^2 denotes the variance, in this situation it acts as the amplitude, and l defines the horizontal length scale. This kernel is used as it provides a smooth prior on functions sampled from the Gaussian Process. The kernel function in general is used in statistical processes so that we can use a linear classifier to solve non-linear problems. A simple explanation can be given if we consider some arbitrary function $f : \mathbb{R}^n \to \mathbb{R}^m$, the dot product of two arbitrary vectors $\mathbf{x}$,$\mathbf{y} \in \mathbb{R}^n$ is given as $f(\mathbf{x})^T f(\mathbf{y})$, we can then define an arbitrary kernel function such that $k(\mathbf{x}, \mathbf{y} = f(\mathbf{x})^T f(\mathbf{y})$, what we have done here is find a way to calculate the dot product in some vector space without needing to know what the original function f defines. Reverting back to our discussion, the joint distribution has the form

$$\begin{bmatrix}\mathbf{f}\\ \mathbf{f}_*\end{bmatrix} \sim \mathcal{GP}\left(\begin{bmatrix}\boldsymbol{\mu}\\ \boldsymbol{\mu_*}\end{bmatrix}, \begin{bmatrix}\mathbf{K} & \mathbf{K}_*\\ \mathbf{K}_*^T & \mathbf{K}_{**}\end{bmatrix}\right) \tag{3.18}$$

where $\mathbf{f}$ is our observation, $\mathbf{f}_*$ are the outputs we wish to predict [19]. If we define our training set as $\mathbf{X}$ and our test set defined as $\mathbf{X}_*$ then $\mathbf{K} = k(\mathbf{X}, \mathbf{X})$, $\mathbf{K}_* = k(\mathbf{X}, \mathbf{X}_*)$ and $\mathbf{K}_{**} = k(\mathbf{X}_{**}, \mathbf{X}_{**})$. From [19] the posterior is expressed as

$$p(\mathbf{f}_*|\mathbf{X}_*, \mathbf{X}, \mathbf{f}) = \mathcal{GP}(\mathbf{f}_*|\boldsymbol{\mu}_*, \boldsymbol{\sigma}_*) \tag{3.19a}$$

$$\boldsymbol{\mu}_* = \boldsymbol{\mu}(\mathbf{X}_*) + \mathbf{K}_*^T\mathbf{K}^{-1}(\mathbf{f} - \boldsymbol{\mu}(\mathbf{X})) \tag{3.19b}$$

$$\boldsymbol{\sigma}_* = \mathbf{K}_{**} - \mathbf{K}_*^T\mathbf{K}^{-1}\mathbf{K}_*. \tag{3.19c}$$

Here we have shown the use of noise-free predictions in which we have assumed that our observations have no normally distributed noise ϵ, in the real world this does not suffice and we introduce the ϵ term and our goal is be to get as close as possible to accurately predicting the data [19]. However as will be discussed in the next subsection since we are sampling the data with a probability distribution we will not need to consider including noise in our predictions, if the reader is interested in this subject we advice you to read [3] and [19].

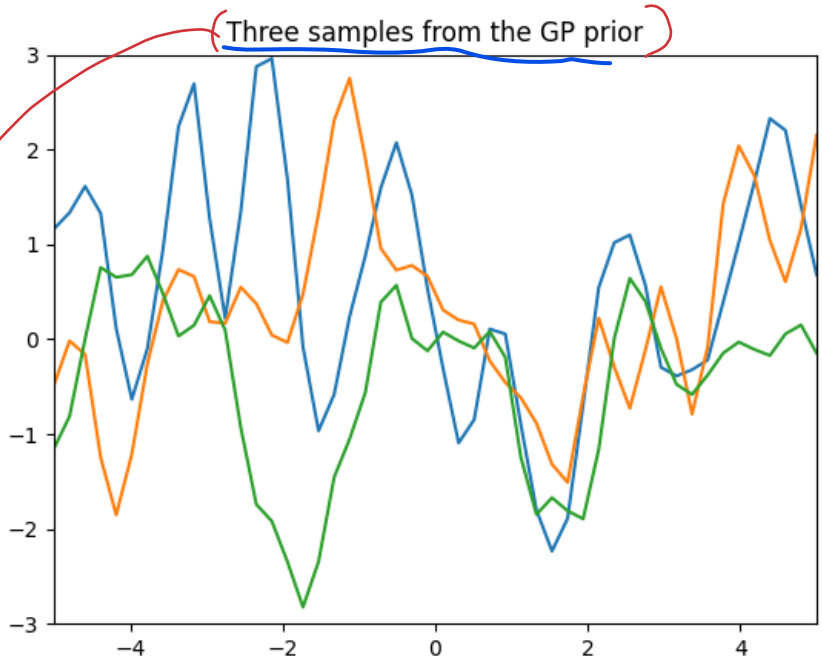
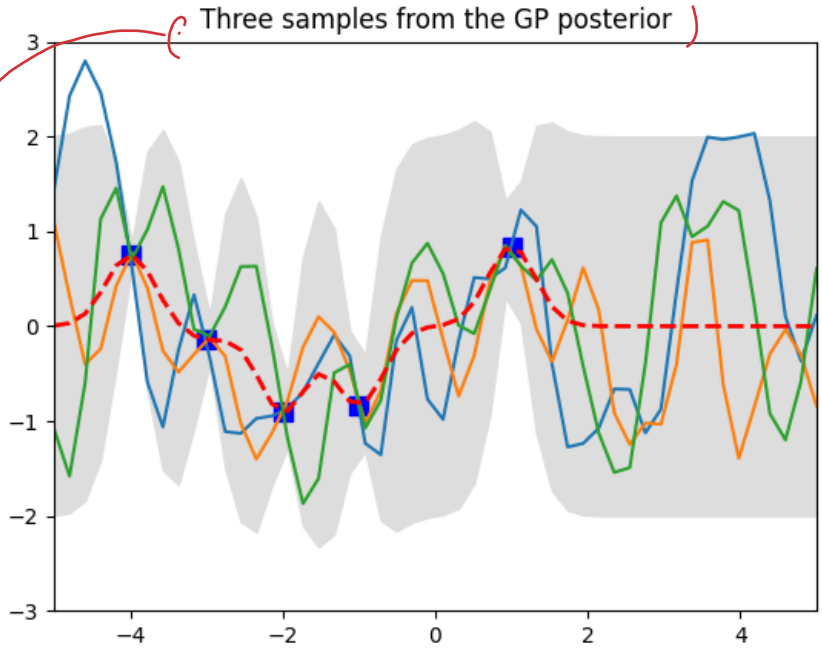


Figure 10: Here we have some random functions that we might think that our data might follow, i.e. our prior.

That is the description in the main text



Move to main text.

posterior.

Figure 11: After using the Gaussian process we obtain a confidence on how accurate our prior is were the blue dots represent our data points, and the red line denotes the real function that the data is represented by.

This can be put into a new chapter together with Section 4.: Learning shear-response using GP

3.4.4 Latin Hypercube

In this study we will be simulating the data sets as mentioned before from a study done by Hao Gao (2017) [21], the reader may think we may then rightfully use some kind of random sampling method where we would just create a table of 1000 random data sets. However this is not good enough, due to the pseudo-random nature of the random function it leads to having data points that collect to a certain area, furthermore this form of sampling does not take into account of previous values. This is where the Latin hypercube comes into play, we need a sampling method that gives us a representation of data we could hypothetically collect in a real world environment. The idea of the Latin hypercube comes from generalising the idea of the Latin square from two dimensions to n dimensions. The Latin square is an idea from combinatorics where we fill an $n \times n$ with n distinct items such that each item appears once in each row and each column [22], so as a enlightening example think of a Sudoku puzzle.

$$\begin{bmatrix} 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 & 10 \\ 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 & 10 & 1 \\ 3 & 4 & 5 & 6 & 7 & 8 & 9 & 10 & 1 & 2 \\ 4 & 5 & 6 & 7 & 8 & 9 & 10 & 1 & 2 & 3 \\ 5 & 6 & 7 & 8 & 9 & 10 & 1 & 2 & 3 & 4 \\ 6 & 7 & 8 & 9 & 10 & 1 & 2 & 3 & 4 & 5 \\ 7 & 8 & 9 & 10 & 1 & 2 & 3 & 4 & 5 & 6 \\ 8 & 9 & 10 & 1 & 2 & 3 & 4 & 5 & 6 & 7 \\ 9 & 10 & 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 \\ 10 & 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 \end{bmatrix}.$$

Figure 12: An example of a simple Latin square where the distinct elements are numbers. Note: this is a very simple case and serves as an elementary example, in actual sampling we would distribute the elements with some probability distribution i.e a uniform or normal/Gaussian.

We will not go into the mathematics to in depth in this discussion but for those who wish to learn further a great resource would be [23]. Moving to the three dimensional case for the Latin hypercube we sample a function of N variables, the range of each variable divided into M equally probable intervals, M sample points are then placed to satisfy the Latin hypercube requirements which will ensure from each placed hypercube we could travel the function space along any direction parallel with any of the axes without encountering any other placed hypercube [9]. We will create 1000 samples where we will use 70% for training 20% for validation and 10% will be used for testing. Following from the data gathered from [21] we will generate data of the material responses from figure 11 with the ranges seen in table 1 and the value of γ is taken from [6].

	a (kPa)	b	a_f (kPa)	b_f	a_s (kPa)	b_s	a_{fs} (kPa)	b_{fs}
controls	0.18 ± 0.1	2.6 ± 0.8	3.34 ± 0.94	2.73 ± 1.06	0.69 ± 0.26	1.11 ± 0.40	0.31 ± 0.18	2.58 ± 0.71
MI subjects	0.09 ± 0.04	4.05 ± 1.6	6.8 ± 3.25	5.53 ± 1.84	1.5 ± 0.66	1.93 ± 0.78	0.16 ± 0.07	4.22 ± 1.7

Figure 13: Summary of average passive parameters of patients tested in [21] where the control are healthy patients and MI are those who suffered from myocardial infarction, adopted from [21].

Table 1: Range of parameters

a	b	a_f	b_f	a_s	b_s	a_{fs}	b_{fs}	γ
[0.05,10]	[0.1,30]	[0.1,30]	[0.1,30]	[0.05,10]	[0.1,30]	[0.05,10]	[0.1,30]	[0.01,0.5]

Chapter 4: Learn shear response using GP

4.1 Brief introduce the aim of this chapter.

4.2. generating data

(a) introduce LHC

(b) what types of data are generated

constant gamma

varied gamma.

4.3 Result.

constant gamma.

varied gamma.

4.4. Brief discussion and summary

4 Results

4.1 Use of 1000 parameters

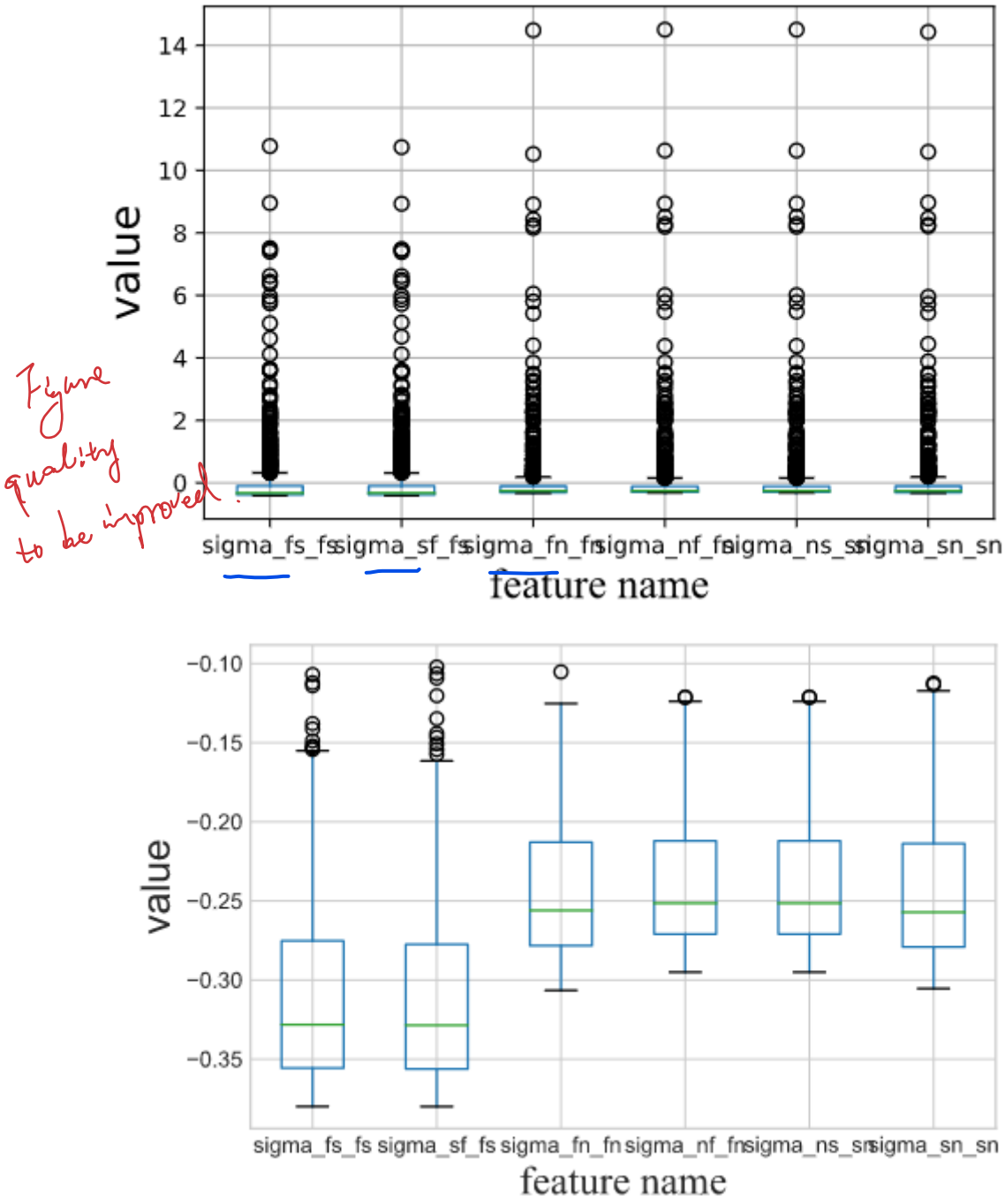


Figure 14: Distribution of 1000 parameters for each plane of stress before taking outliers out and after

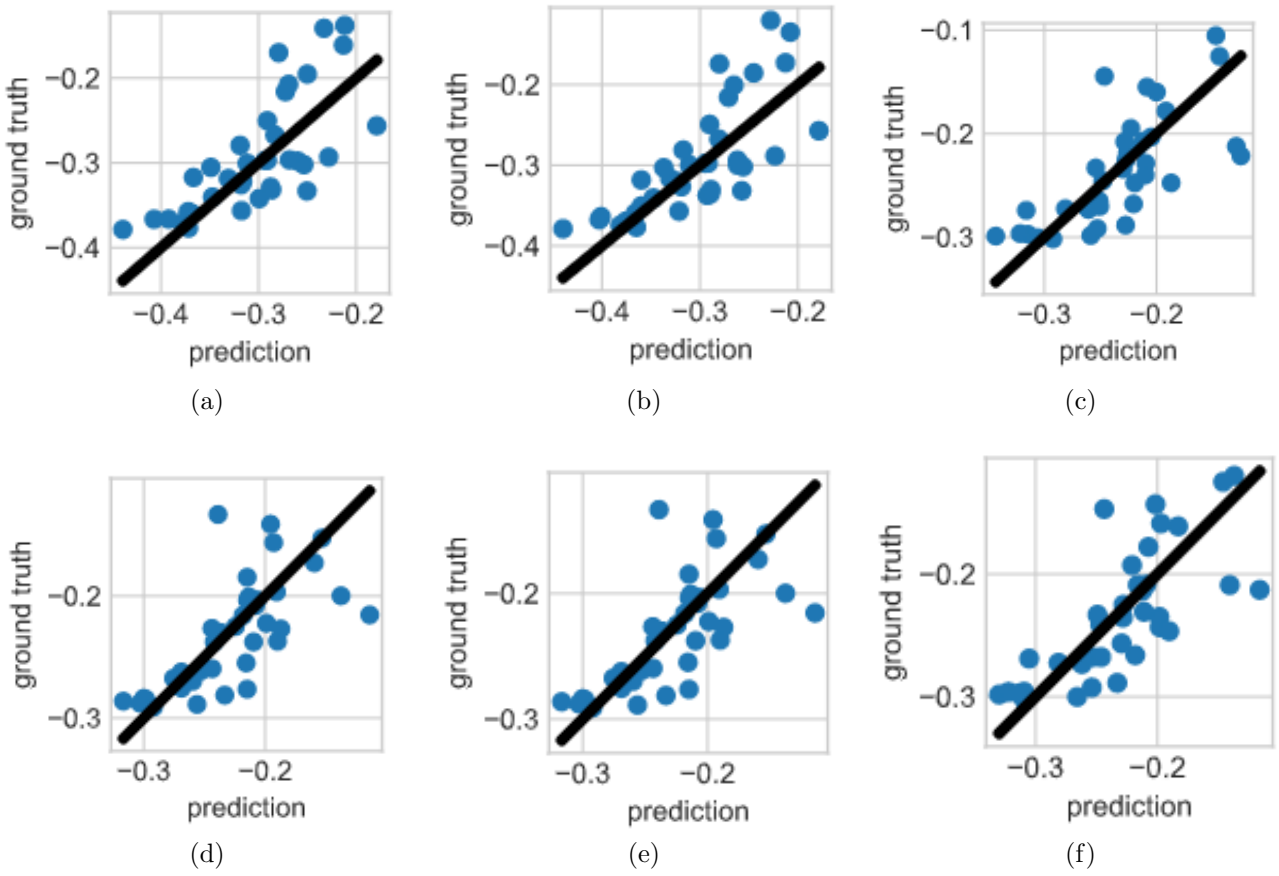
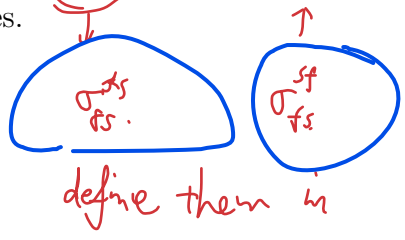


Figure 15: Predictions of the stress plane given the simulated material parameters and shear stress values relating to a plane of stress. (a) is the $fsfs$, (b) is the $sffs$, (c) is the $fnfn$, (d) is the $nffn$, (e) is the $nssn$ and (f) is the $snsn$ planes.



4.2 Use of 50,000 parameters

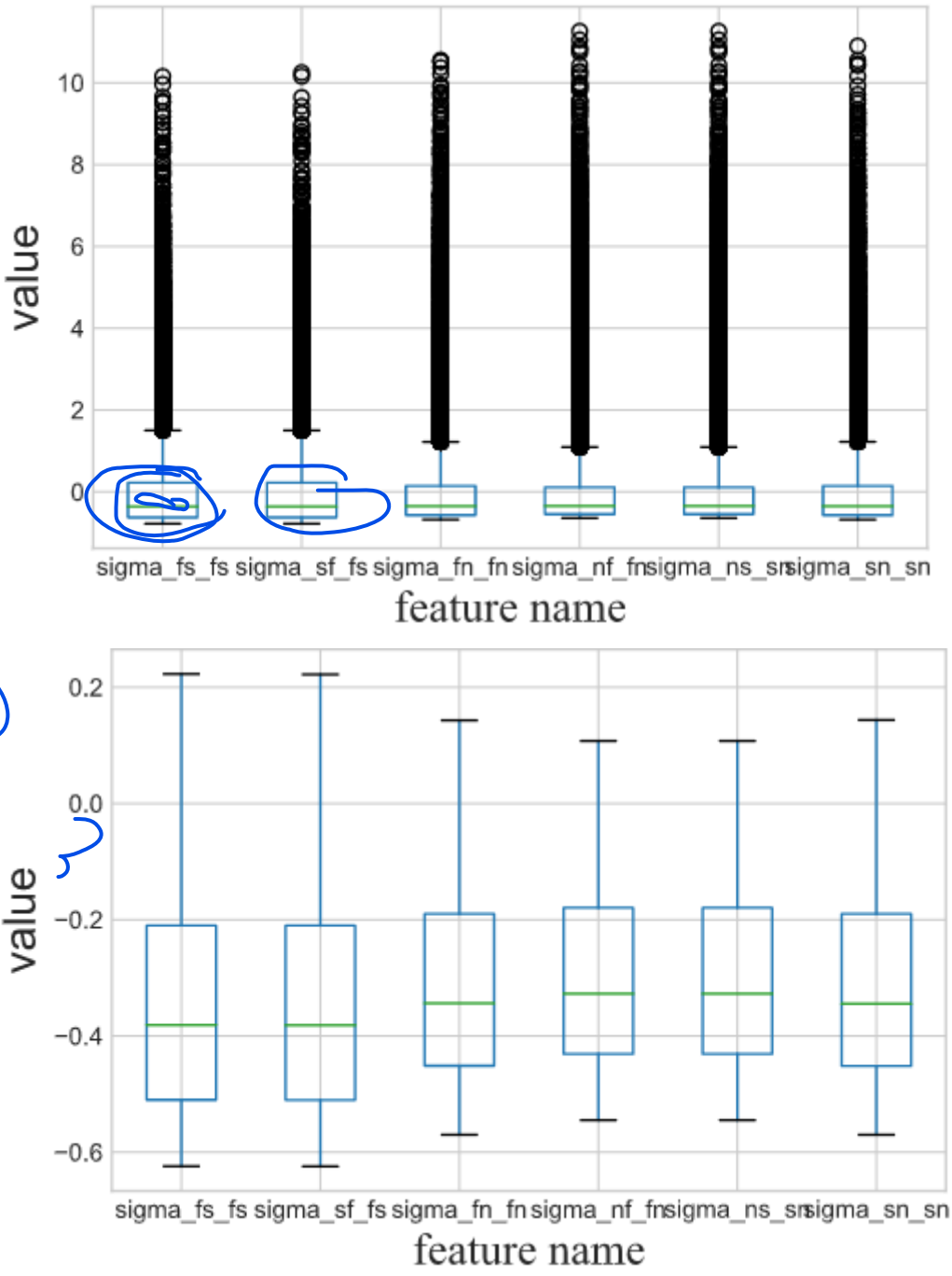


Figure 16: Distribution of 50000 parameters for each plane of stress before taking outliers out and after

Shear plane	MSE	r^2
nf-fn	0.003730	0.861779
fn-fn	0.005365	0.819387
sf-fs	0.007342	0.824481
fs-fs	0.007342	0.825867
sn-sn	0.005416	0.816724
ns-sn	0.003730	0.861779

4.3 Constant Gamma

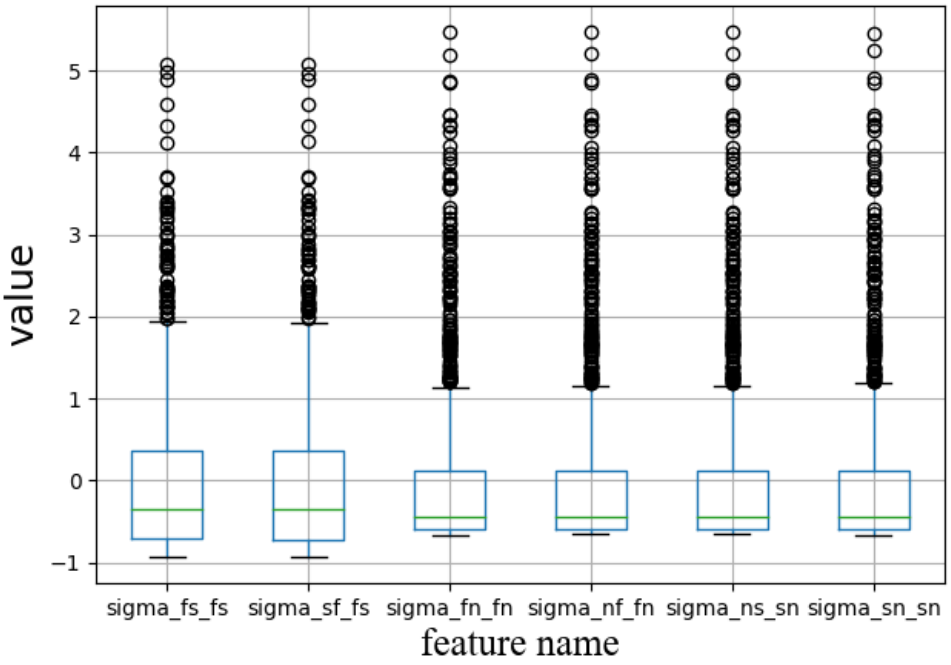


Figure 17: Distribution of the data where the shear data is constant

Shear plane	MSE	r^2
fs-fs	0.006126	0.991505
sf-fs	0.006118	0.991488
fn-fn	0.004523	0.99570
nf-fn	0.004516	0.994581
ns-sn	0.004516	0.994581
sn-sn	0.004516	0.994754

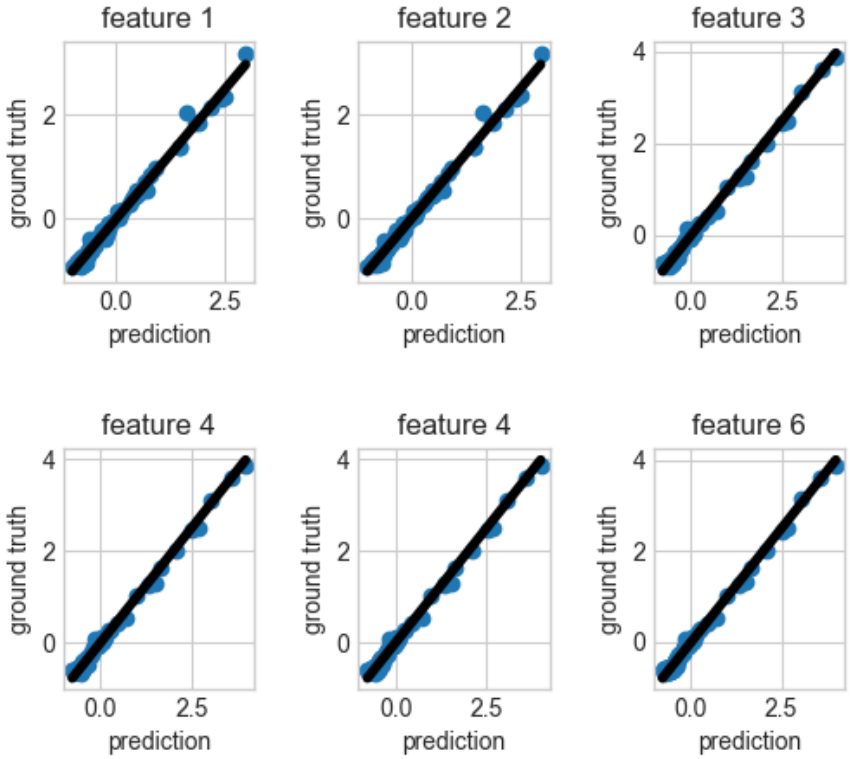


Figure 18: Training on the simulated data for each plane of shear where each feature corresponds to the the stress plane in order shown in Figure 17

4.4 Varied Gamma

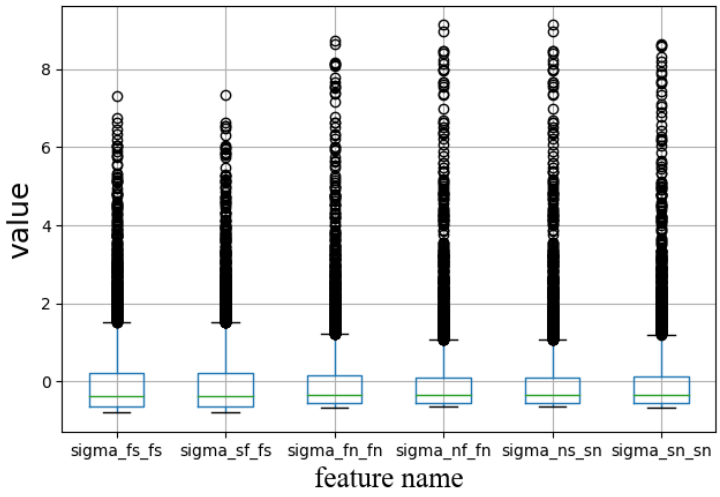


Figure 19: Distribution of data where we have a shear that varies

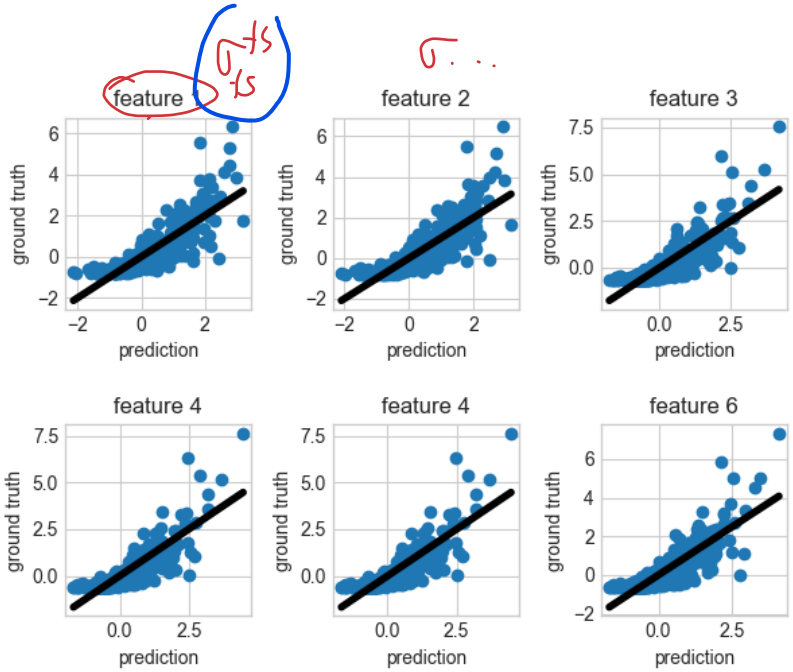


Figure 20: Training on the simulated data for each plane of shear where each feature corresponds to the the stress plane in order shown in Figure 19

Shear plane	MSE	r^2
fs-fs	0.311841	0.682367
sf-fs	0.310075	0.679548
fn-fn	0.282772	0.672920
nf-fn	0.262166	0.690056
ns-sn	0.262166	0.690056
sn-sn	0.278699	0.667150

5 Discussion

A Appendix title

References

[1] J. M. Guccione G. S. Kassab M. B. Ratcliffe, Computational Cardiovascular Mechanics, Springer, (2010)

[2] B. J. Kogan, Introduction to Computational Cardiology, Springer, (2010).

[3] C. E. Rasmussen C. .K .I .Williams, Gaussian Processes for Machine Learning, MIT Press, (2006)

[4] Dabiri, Yaghoub et al. "Prediction of Left Ventricular Mechanics Using Machine Learning." Frontiers in physics vol. 7 (2019): 117. doi:10.3389/fphy.2019.00117

[5] Alber, M., Buganza Tepole, A., Cannon, W.R. et al. Integrating machine learning and multiscale modeling—perspectives, challenges, and opportunities in the biological, biomedical, and behavioral sciences. npj Digit. Med. 2, 115 (2019).

[6] Holzapfel Gerhard A. and Ogden Ray W. (2009) Constitutive modelling of passive myocardium: a structurally based framework for material characterizationPhil. Trans. R. Soc. A.3673445–3475

[7] Woods RH. A Few Applications of a Physical Theorem to Membranes in the Human Body in a State of Tension. J Anat Physiol. 1892;26(Pt 3):362-370.

[8] British Heart Foundation UK Fact sheet 2020 July

[9] Surrogate models based on machine learning methods for parameter estimation of left ventricular myocardium. Li Cai, Lei Ren, Yonghe Wang, Yiqiang Li, Jie Jiao1, Hao Gao

[10] 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 by using statistical emulation. J. R. Stat. Soc. C, 68: 1555-1576. doi:10.1111/rssc.12374

[11] Maso Talou Gonzalo D., Babarenda Gamage Thiranja P., Sagar Mark, Nash Martyn P. Deep Learning Over Reduced Intrinsic Domains for Efficient Mechanics of the Left Ventricle. Frontiers in Physics Volume 8 (2020) pg 30. https://www.frontiersin.org/article/10.3389/fphy.2020.00030

[12] Non-Linear Elastic Deformations (1997). R.W.Ogden.

[13] Wang, H.M., Gao, H., Luo, X.Y., Berry, C., Griffith, B.E., Ogden, R.W. and Wang, T.J. (2013), Structure-based finite strain modelling of the human left ventricle in diastole. Int. J. Numer. Meth. Biomed. Engng., 29: 83-103. doi:10.1002/cnm.2497

[14] Minliang Liu, Liang Liang, Wei Sun, Estimation of in vivo constitutive parameters of the aortic wall using a machine learning approach, Computer Methods in Applied Mechanics and Engineering, Volume 347, 2019, Pages 201-217, ISSN 0045-7825, https://doi.org/10.1016/j.cma.2018.12.030.

[15] Mangion K, Gao H, Husmeier D, Luo X, Berry C. Advances in computational modelling for personalised medicine after myocardial infarction. Heart. 2018 Apr;104(7):550-557. doi: 10.1136/heartjnl-2017-311449. Epub 2017 Nov 10. PMID: 29127185.

[16] Noè Umberto, Lazarus Alan, Gao Hao, Davies Vinny, Macdonald Benn, Mangion Kenneth, Berry Colin, Luo Xiaoyu and Husmeier Dirk (2019). Gaussian process emulation to accelerate parameter estimation in a mechanical model of the left ventricle: a critical step towards clinical end-user relevance J. R. Soc. Interface.1620190114 http://doi.org/10.1098/rsif.2019.0114

[17] Surrogates - Gaussian Process Modeling, Design, and Optimization for the Applied Sciences (2020) .Robert B. Gramacy. CRC Press

[18] Ian Goodfellow, Yoshua Bengio and Aaron Courville- Deep Learning- MIT Press [2017]

[19] Kevin P. Murphy - Machine Learning- A Probabilistic Perspective- MIT Press [2012]

[20] Lui et al - A generic physics-informed neural network-based constitutive model for soft biological tissues (2020)

[21] Gao Hao, Aderhold Andrej, Mangion Kenneth, Luo Xiaoyu, Husmeier Dirk and Berry Colin 2017 Changes and classification in myocardial contractile function in the left ventricle following acute myocardial infarction J. R. Soc. Interface.1420170203

[22] Razi Sheikholeslami, Saman Razavi, Progressive Latin Hypercube Sampling: An efficient approach for robust sampling-based analysis of environmental models, Environmental Modelling Software, Volume 93, 2017, Pages 109-126, ISSN 1364-8152, https://doi.org/10.1016/j.envsoft.2017.03.010.

[23] A. Olsson, G. Sandberg, O. Dahlblom, On Latin hypercube sampling for structural reliability analysis, Structural Safety, Volume 25, Issue 1, 2003, Pages 47-68, ISSN 0167-4730, https://doi.org/10.1016/S0167-4730(02)00039-5.

[24] Noè, Umberto (2019) Bayesian nonparametric inference in mechanistic models of complex biological systems. PhD thesis.

[25] Ben Lambert - A Students Guide to Bayesian Statistics - Sage [2018]