# Predicting Femoral Strain Using Surrogate Modelling

By

**Thomas Rundle**

2219463

*Thesis*

*Submitted to Flinders University, in conjunction with Griffith University for the degree of*

**Bachelor of Engineering (Biomedical) (Honours) /**

**Master of Engineering (Biomedical)**

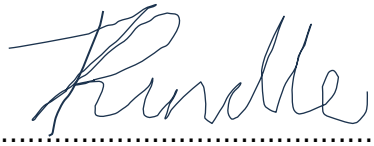
College of Science and Engineering

03/11/2023

# DECLARATION

I certify that this thesis:

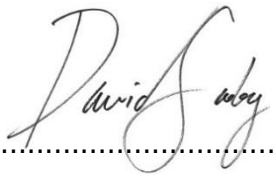
1. does not incorporate without acknowledgment any material previously submitted for a degree or diploma in any university
2. and the research within will not be submitted for any other future degree or diploma without the permission of Flinders University; and
3. to the best of my knowledge and belief, does not contain any material previously published or written by another person except where due reference is made in the text.

Signature of student.....  .....

Print name of student..... Thomas Miles Rundle

Date.....03/11/2023.....

I certify that I have read this thesis. In my opinion it is/is not (please circle) fully adequate, in scope and in quality, as a thesis for the degree of <Degree Name>. Furthermore, I confirm that I have provided feedback on this thesis and the student has implemented it minimally/partially/fully (please circle).

Signature of Principal Supervisor.....  .....

Print name of Principal Supervisor.....David John Saxby

Date.....03/11/2023.....

# ACKNOWLEDGEMENTS

I would like to acknowledge my academic supervisors, firstly from Griffith University, Dr David Saxby and Dr Claudio Pizzolato, and also from Flinders University, Professor Mark Taylor. The support, experience and guidance provided they provided has ultimately been a driving force in the completion of this study, and I am extremely grateful for their help.

I would also like to express gratitude to some PhD students at Griffith University I was lucky to work alongside throughout my study. Alireza Bavil Yahyaiee and Emmanuel Eghan-Acquah, thank you sincerely for assisting me with Finite Element related problems I encountered, and for allowing me to use your highly developed femur model in the later stages of my project. Thank you also to Claire Crossley, for providing some baseline force measurements used through the study.

Finally, a special thank you to Dermot O'Rourke, for his willingness to assist in a variety of areas. Thank you sincerely for sharing your knowledge in Finite Element studies in general, and with your assistance in constructing valid datasets.

# EXECUTIVE SUMMARY

***Background:*** Quantification of femoral strain in real-time is valuable for a range of biomedical fields as it enables rapid assessment of fracture risk. Amongst individuals living with spinal cord injury, bone fracture during rehabilitation and exercise poses a particularly high risk given diminished bone mass. Further, the lack of sensory feedback can result in injuries untreated and lead to health implications.

Currently, the finite element (FE) method is used to predict femoral strains in response to applied loads. Although the FE method has been validated for many models of bone mechanics, it is time-consuming, requires high-level training to operate, and requires extensive model development for each new application (e.g., patient). This study proposes a method which uses surrogate modelling of legacy datasets to predict femoral strain in response to novel and, in principle, arbitrary applied loading.

***Methods****:* Four techniques were investigated: multi-linear regression (MLR), cubic splining, Superposition Principle Method (SPM), and Kriging. Surrogate models were created in MATLAB (Mathworks, USA), and were used to predict femoral strains in response to various loads. Initially, a simplified linear elastic FE model of the femur was developed in Ansys (Ansys, USA) and used to train surrogates. Based on this initial analysis, the most novel and promising technique (SPM) was further explored by applying a more realistic material model: elastic non-linear, and deployed in Abaqus (Dassault Systemes, France). Validation of all techniques involved comparing surrogate predicted to FE solved strains and quantifying error between them using normalized root mean square error (nRMSE) and assessing differences in computational demand via central processing unit (CPU) time.

***Results:*** The initial linear FE models showed SPM predicted FE-modelled strains with zero error. The MLR and cubic splining techniques were both effective, with nRMSE values of <0.05% and <0.08% respectively. Kriging was inaccurate, with nRMSE >15%. All techniques were computationally tractable, but MLR was slowest taking ~14.9 seconds while splining was fastest taking ~1.50 seconds. When applied to the non-linear model, SPM was still accurate, with nRMSE of ~5% and CPU time <20 seconds.

***Conclusion:*** The SPM is the recommended surrogate modelling technique for applications requiring near-real-time femoral strain quantification. Despite being a lesser known and under-developed method, it provided exact strains in a linear model, and highly accurate

ones in a non-linear model in a timely manner. Other methods were found to be less favourable, however their lack of testing in a non-linear environment should be considered. Through code optimization, it is expected that SPM could run in real-time.

# TABLE OF CONTENTS

# LIST OF FIGURES

# LIST OF TABLES

# 1. INTRODUCTION

## 1.1 Background

Real-time or near-real-time quantification of femoral strain has shown enormous benefits in biomechanical applications (Ziaei Poor et al, 2019), including the prediction of bone fragility (Martelli et al, 2015), analysis of running and cycling gaits (Zeng et al, 2020), and improving the design of implantable devices (Singh et al, 2023). Furthermore, the ability to predict femoral fractures is an especially beneficial notion in our rapidly ageing population, as this demographic typically suffers from osteoporosis (SpinalCure, 2020). Moreover, there is an enormous economic consumption as a consequence of hip and femoral fractures, with an estimated worldwide cost of $12 billion (Burge et al, 2007).

More recently, the concept has been used in conjunction with muscle-excitation feedback to patients whilst they are exercising (Pizzolato et al, 2017), in a therapeutic activity known as lower limb Functional Electrical Stimulation (FES). Typically, this therapy is conducted on individuals living with spinal cord injury (SCI), with studies by Martin et al (2012), and BioSpine (2022), demonstrating beneficial FES application in voluntary muscle function rehabilitation. Thus far, they have seen promising results ultimately showing evidence of restoring muscle sensation. This is big news, in that SCI puts an individual's life at risk of numerous health complications (Bennett et al 2022) and are an economic burden on not only the individual but also their society. With 20,800 Australians living with SCI, the lifetime cost of their injuries has been valued at $75.4 billion, and it has been estimated that even partial reversion of 10% of the population living with this paralysis could lead to a saving of more than $3.5 billion (SpinalCure, 2020).

Specifically, quantification of femoral strains and stresses throughout the femur of people living with SCI is very valuable, in that the nature of their injuries inhibits their ability to feel muscle strains and bone breaks. This holds potential for extreme health risks during FES, since injuries may go unnoticed and hence untreated, leading to further implications. By modelling the femur of the individual, considering their bone density, we can analyse the likely behaviours of the femur at given strain levels, in theory providing information on what expected stimulations the individual can handle during FES treatment. Ultimately, real-time strain prediction provides added safety for the individual during their therapy.

A series of Finite element (FE) models has been the leading tool for studying and quantifying femoral strains (Taylor and Prendergast, 2015), however, the process requires complex procedures to set up and is computationally very demanding (Panagiotopoulou et al, 2011; Taylor et al, 2017; Ziaei Poor et al, 2019b), which prevents near-real time results which are particularly useful in the clinic. The lack of prompt solutions can limit its application going forward in time-sensitive applications (Liang et al, 2018).

## 1.2 An Improved Method

This study proposes an improved method of finding femoral strains, utilising surrogate modelling techniques, to reduce the time currently used in obtaining FE results of multiple models or simulations. This time save is possible by running fewer FE simulations and using advanced and accurate interpolation methodologies on legacy datasets to predict the femur's response to future, unknown loadings. The overarching premise of this study is to investigate the multiple known surrogate modelling techniques and analyse their applicability in femoral strain predictions, in attempt to clarify which surrogate modelling technique specifically is best suited to predicting strains in a way which is both very accurate, and faster with respect to time and power consumption than current systems.

# 2. LITERATURE REVIEW

## 2.1 Precursor

This literature review compares a variety of surrogate modelling techniques with regards to their application in the prediction of femoral strains in a timelier manner than current processes, such as Finite Element Analysis (FEA). Literature shows that Kriging (Gaussian process modelling) is especially valuable when working with non-linear systems (Eskinazi and Fregly, 2015, Tu, 1996), and is well known for its complex processing leading to high accuracy at the expense of high computational requirements (Zhang, 2016, Pizzolato et al, 2017, Ziaei Poor et al, 2019a, Ziaei Poor et al, 2019b). Simpler mathematical procedures were also investigated, such as multi-linear regression and cubic splining strategies, which revealed a much faster and simpler implementation (O'Rourke et al, 2016, Goldman, 2003) but are not capable of processing such accurate predictions and are mainly limited to strictly linear systems (Pizzolato et al, 2020, Sartori et al, 2012). Lastly, a newly developed Superposition Principle Method (SPM) was explored, which exhibited high accuracy characteristics whilst still being extremely simple to implement (O'Rourke et al, 2019). Unlike the other methods, the SPM is not a machine learning (ML) strategy, but rather simple linear interpolation, and hence the assumption of a perfectly linear system is critical (Taylor, 2023). In the scope of femoral predictions, this assumption is often made. In further studies, it is likely that all these techniques will be used or experimented with to predict femoral strains, however it is hypothesised that Kriging and the SPM will be most suitable.

## 2.2 Review of Kriging (Gaussian Process Regression)

Kriging is a statistical technique used in interpolation, utilising Gaussian processes to model predicted values based on prior covariances (Kumar er al, 2020). Historically it is best suited to forecasting values of a geographical area, for example in the fields of soil and mining (Matheron, 1973), however it has also shown success in structural reliability analysis (Gaspar, 2014), and has recently been included in some biomechanical applications (Taylor et al, 2016), due to its ability to solve non-linear problems (Eskinazi and Fregly, 2015). Kriging is a complex process which considers the magnitude and location of known data points in its prediction models. Uniquely, Kriging is a stochastic approach, meaning it uses statistical procedures rather than mathematical operations to formulate its calculations (Rebholz and Almekkawy, 2020). This is useful in that the degree of error can also be

predicted, indicating the uncertainties of the model at different points. As expected, model error estimates are lower where the distribution field is dense (Bagheri et al, 2017). This principle is demonstrated in Fig. 1, which appears in a paper written by Wang (2022), explaining how there are an infinite number of possible functions which can pass through a given set of observed data points.

Figure removed due to copyright restriction

**Figure 1 - Demonstration of error prediction: (a) Observed data points, (b) Five sample functions that fit the observed data points (Wang, 2022).**

Taylor et al (2016) performed a study comparing Kriging to multivariate linear regression techniques, with results concluding that the kriging model was more accurately able to predict 95th percentile strains through the hemipelvis, with an $R^2$ value of 99% when given 30 data training sets. Comparatively, the multi-linear regression model saw an $R^2$ of only 87% at best, concluding that Kriging is a profoundly accurate surrogate modelling technique. This evidence is also by supported in studies done by Gaspar et al (2014), proving superiority over polynomial regression strategies, and Haeri and Fadae (2016), demonstrating accurate analysis of laminated composite models. Kriging has also shown excellent potential when used as a surrogate for FE models containing contact forces (O'Rourke, 2023), however this may not be a requirement in the project at hand, considering an isolated femur model will be used.

The Kriging model is well known for its high computational energy consumption amongst the other surrogate modelling techniques, supported by papers written by Ziaepoor et al (2019), Bagheri et al (2017) and Haeri and Fadaee (2016), which is a limitation of its application. Despite this, in comparison with the full-factorial analyses completed within finite modelling software, the computational intensity is still significantly lower, and hence this limitation's effect is not as detrimental. Another limitation of this method is that it requires a large sample

of input points, otherwise the resulting predictions and errors exhibit large deviations (Chu et al, 2020).

Kriging is an all-around versatile surrogate modelling technique, commonly known as one of the leading methodologies in terms of accuracy (Taylor et al, 2016, Freier et al, 2017, Gaspar et al, 2014). Its ability to create a smooth prediction surface whilst considering the degree of confidence is highly beneficial (Bagheri et al, 2017).

## 2.3 Review of Multi-linear Regression (MLR)

Multivariate linear regression is a mathematical technique used to model a relationship between several independent variables and a singular dependent outcome (Marill, 2004), by fitting a linear equation to the observed data. Stockemer (2018) explains this in his book, Quantitative Methods for the Social Sciences, using the analogy of a student's exam results being dependent on not only their study habits, but also their health, their mood, and their sleep. Using multi-linear regression all variables can be considered to gauge their influence absolutely and comparatively on the outcome.

Multi-linear regression is an extension of simple linear regression (Hayes, 2023), in that it assumes independence of observations, meaning each independent variable is linearly correlated to the dependent variable. It also assumes that the various independent variables are not correlated to one another (Slinker and Glantz, 1985). These assumptions can be validated by the trend in residuals, which should have a normal distribution with mean zero and constant standard deviations (Alexopolous, 2010). It has had successful application in geographic disciplines such as weather forecasting (Hay and Viger, 1999, Chung et al, 1995) as well as medical uses including diagnostic research (Marill, 2004) and fracture risk assessment (O'Rourke et al, 2017, Awal and Faisal, 2021), where outcomes may be dependent on multiple inputs.

Accuracy-wise, multi-linear regression is not the optimal surrogate modelling technique, as shown by the study mentioned above including the hemipelvis, utilizing the kriging method (O'Rourke et al, 2016). This evidence is also supported by Bekesiene et al, 2021, in a study attempting to predict ozone concentration changes using multi-linear regression and artificial neural networks. Results concluded that the multi-linear regression model lacked capacity for precise measures, where other methods offered more accurate outcomes. Despite this, a point of interest for some researchers is that amongst other methodologies it is fast and

simple, computing only linear models as seen in Eq. 1 below, taken from Bevans, 2020. This property is highly desired for real-time applications.

$$y = \beta_0 + \beta_1 X_1 + \beta_2 X_2 \ldots + \beta_n X_n + \epsilon \qquad [1]$$

Where:

- $y$ is the predicted dependent value.
- $\beta_0$ is the y-intercept.
- *n* is the number of independent variables being considered.
- $\beta_n$ is the regression coefficient of the *n*th independent variable.
- $\epsilon$ is the model's error term, known as residuals.

## 2.4 Review of Spline Interpolation

Regular cubic splines, like linear regression techniques, utilise mathematical operations to carry out the interpolation. However, rather than linear principles, this method links data points, known as 'knots' using a series of unique cubic polynomials, creating a smooth, piecewise curve between known values (Biran, 2019, McClarren, 2018, Mostoufi and Constantinides, 2023, Phillips and Talor, 1996).

To ensure each piecewise function fits smoothly, the first and second derivatives of adjacent functions are equated to evaluate the polynomial coefficients (Phillips and Taylor, 1996). Each of *k* functions can be expressed using Eq 2., and are plotted together as shown in Fig 2, both pulled from Wolberg's review of cubic spline interpolation (1988).

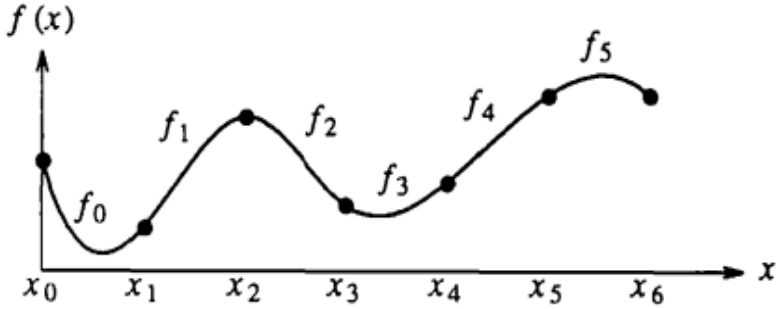



**Figure 2 – Cubic spline implementation**

$$f_k(x) = A_3(x - x_k)^3 + A_2(x - x_k)^2 + A_1(x - x_k) + A_0 \qquad [2]$$

Like MLR, splining typically has the most success in linear systems, since its calculating processes are mathematically simple. A popular variation of this is cubic B-spline interpolation. In contrast, cubic B-spline interpolation also involves constructing piecewise functions to model the data, however the chosen knots do not need to coincide with known data points (Farin, 2002), and can be flexibly placed within the data set, which in most cases results in a closer approximation of the true data trends (Goldman, 2003). This typically provides a more flexible model and is more robust when exposed to non-linear systems.

## 2.5 Review of Superposition Principle Method (SPM)

This Superposition Principle Method is by far the least documented and has been developed only in recent years. It differs from other techniques in that it is not necessarily a machine learning technique, in that its principles do not revolve around the prediction of future variables. Instead, the fundamental theory of the SPM is that the strain field of a system in response to a given input scheme can be found using simply using the known strain tensors found in response to other inputs.

A paper written by Ziaei Poor et al (2019) is the first and only documentation of this method, and conveniently this study focuses on finding femoral strains. It explains that a muscle's contribution to strain can be described by calculating the resultant strain tensor in response to the three force components at the relevant attachment points. Hence, by applying arbitrary loadings at each muscle attachment or contact force location, and solving for the nodal strain components, it can be said that every solution in the model can be expressed as a linear combination of scaled solutions found previously (Ziaei Poor et al, 2019). Fig. 3 shows this in greater detail.

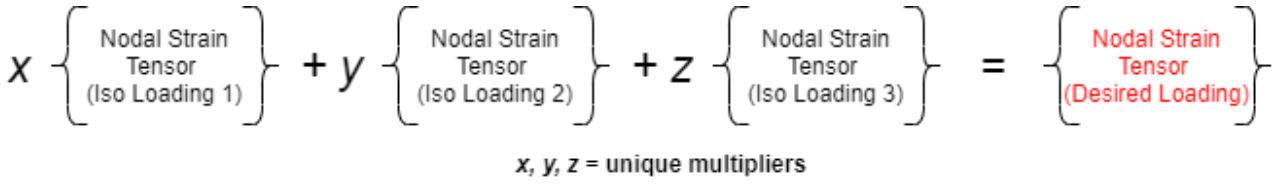



**Figure 3 - Demonstration of superposition of strain tensors to predict combined loading outputs.**

A meeting with one of the Australia's leading surrogate modelling experts, Prof. Mark Taylor (2023) revealed that this method has showed excellent promise in its few implementations thus far, for a variety of reasons. Firstly, an advantageous characteristic of the SPM is that it calculates exact solutions, rather than predictions when utilised on a completely linear system (Taylor, 2023). Not only does this result in a higher accuracy, but it also saves time in that it removes the need to calculate error. Furthermore, SPM interpolation requires a far smaller sample size to be done effectively than comparative methods such as Kriging (Ziaei Poor et al, 2019).

A limitation to this method is that thus far it has only been used reliably within strictly linear systems (O'Rourke, 2023, Ziaei Poor et al, 2019, Taylor, 2023). In this project, however, the SPM will be further analysed to not only confirm its applicability to linear systems, but also in how well it responds to non-linear ones.

Ziaei Poor et al, 2019 showed that in comparison to linear regression, adaptive spline techniques, and Gaussian process methods, the SPM showed the smallest error without requiring any training when predicting femoral strains. It also exhibited the fastest model generation time and the second fastest prediction time per activity, behind multi-linear regression. These results strongly support the application of the SPM in biomechanical strain predictions.

## 2.6 Conclusion of Literary Review

In terms of surrogate modelling, all of the considered techniques are valid in the correct applications. Kriging is clearly more suited to complex problems, as it can provide solutions to non-linear solutions (Eskinazi and Fregly, 2015, Tu, 1996), at the expense of more computational power requirements (Zhang, 2016). In general, Gaussian techniques also offer more accuracy when applied to simpler problems as well, however their solutions are still somewhat comparable to the likes of multi-linear regression and splining methods, which are substantially faster and simpler to implement (Ziaei Poor et al, 2019, Taylor et al, 2019). In a slightly different stream, the SPM was also considered, and looks very promising in that it can effectively provide very high accuracy without training, when linearity is assumed (Ziaei Poor et al, 2019).

Considering all techniques, the SPM is a standout in terms of its high accuracy, low power consumption, and simplicity. Despite this, other methods will also be experimented with, for credibility of research. Through MATLAB, all these techniques have accompanying

packages and toolkits to assist with the construction of a surrogate model (MathWorks, USA).

In summary, all these techniques are substantially faster than standard finite element processing and can all provide a sufficient level of accuracy to justify their use, however some are more applicable to certain scenarios than others. In the case of measuring femoral strains in real time, accuracy is paramount so to not risk harm, and therefore it is likely that higher consideration will be given to the methods which can provide better precision.

# 3. METHODOLOGY

## 3.1 Overview of Project Methodologies

The aim of this study is to find a method which obtains femoral strain predictions in a manner which is faster than, but as accurate as continuously building FE models. To do so, the theory of surrogate modelling and data interpolation indicated that previous FE simulations could be used to predict future ones. Throughout the project, four surrogate modelling techniques were explored:

1. Multi-linear Regression (MLR)
2. Cubic Splining
3. Kriging
4. Superposition Principle Method (SPM)

Prior to implementing the surrogate models, a base dataset needed to be acquired. Using this, mathematical and statistical operations could be performed on the data to make calculated predictions of strain responses to loadings which were not explicitly tested during the FE simulation. Hence the methods used before implementation of the different surrogate techniques were widely the same. The basis of the project is described in Fig 4., which breaks down the different components of the study. The components will be explored further in following sections.

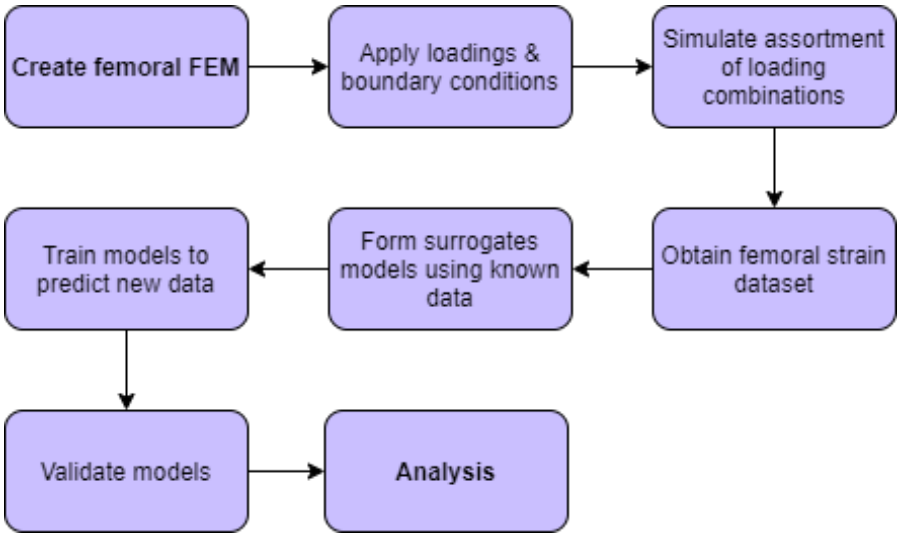



**Figure 4 - Project breakdown**

## 3.2 FE Construction

The FE part of the study was broken into two sub-components, named accordingly:

1. Simplified FE
2. Detailed FE

This was done to simplify the project into steps which appeared more achievable to someone who was new at using surrogate models. The implementation of surrogate models can be complicated, particularly when datasets become large and complex, including multiple predictor variables. Hence, removing some of the predictor variables, whilst certainly compromising the overall accuracy of the analysis, allowed the acquisition of a dataset which appeared more reasonable to operate on using surrogate models. The surrogates were still built with the intention of handling many predictor variables, however only considering a few in this initial stage made the process smoother and more intuitive. In theory, once the initial surrogate models had been made, a more detailed FE model could be made, which considered a larger number of muscle and contact forces, with more anatomically accurate data. Whilst this was the case, unfortunately time constraints prohibited all surrogate modelling techniques to be reviewed using the detailed FE model. This is spoken about further in the limitations section within the discussion.

### 3.2.1 Simplified FE

The simplified FE model was constructed in Ansys (Ansys, USA), using a femoral model obtained from Griffith University (Griffith University, QLD, 2023). The model was cut in half such that only the proximal end of the femur remained, since the literature had proven that it was in the femoral neck where majority of femoral fractures occurred (Florschutz et al, 2015; Merloz, 2018). The bone material was assumed to be linearly elastic throughout this model, in accordance with literature stating that this assumption was valid in most contexts.

The model's mesh specifications were triangular elements, 2.5mm in size. These specifications were not obtained from a convergence study since accuracy was not the focus of this initial FEA. Instead, the principle of correctly executing the surrogate models on the dataset was the priority, and the accuracy of the data would be further considered in the more detailed FEA. The model can be seen in Fig. 5 and 6.

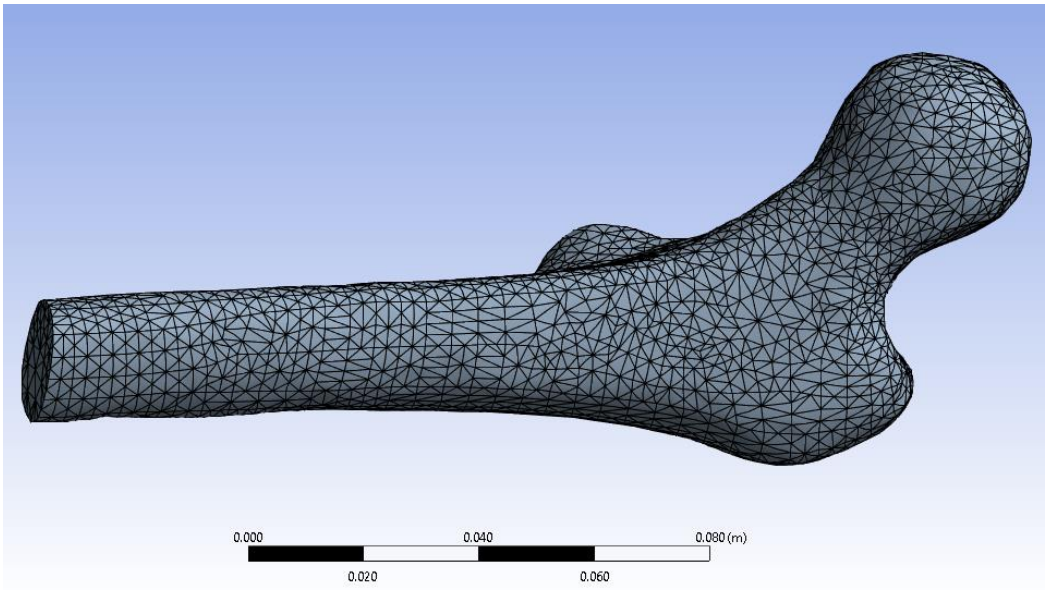



**Figure 6 - Simplified FE model, lateral view**

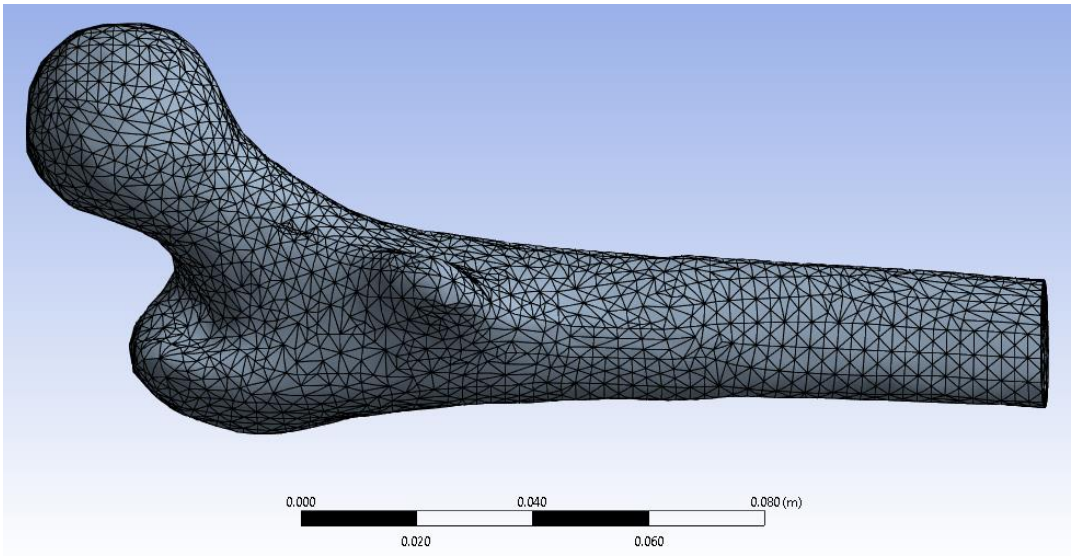



**Figure 5 - Simplified FE model, medial view**

This model was loaded with three forces, defined by the most commonly occurring study points throughout the literature review (Bitsakos et al, 2005; Duda et al, 1998; Kenedi et al, 2014).

1. Hip contact force
2. Adductor muscle force
3. Glute muscle force

It was acknowledged that the knee force is a primary contributor to loadings through the femur, however it was omitted from this investigation since the distal part of the femur had

been ignored. A constraint was also placed on the cut face of the femur as a boundary condition, which would have also led to issues should the knee contact force been considered.

#### *3.2.1.1 Assumptions – Simplified FE*

Force location and direction were fundamental assumptions of this model. Force locations were given to groups of elements which were perceived as most applicable to the given force, via muscle attachment positions viewed in literature (Carriero et al, 2010; Yadav et al, 2017), and directions were simplified as much as possible to the most dominant direction of that muscle or contact force. All forces were considered to be acting along the distal-proximal axis of the femur, with the hip contact force acting in the opposing direction to the glute and adductor. Fig. 7 and 8 demonstrate this. In these figures, A represents a constrained face in all three directions, whilst B, C and D represent the hip contact force, and the glute and adductor forces respectively.

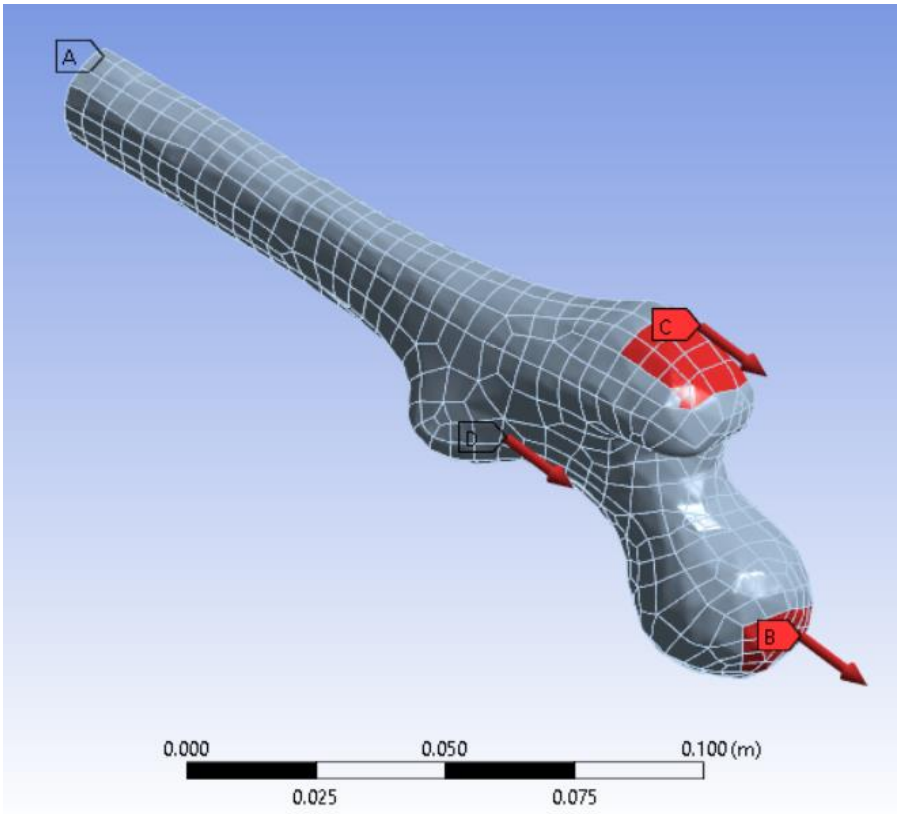



**Figure 7 - View 1 of force locations on half-femur**

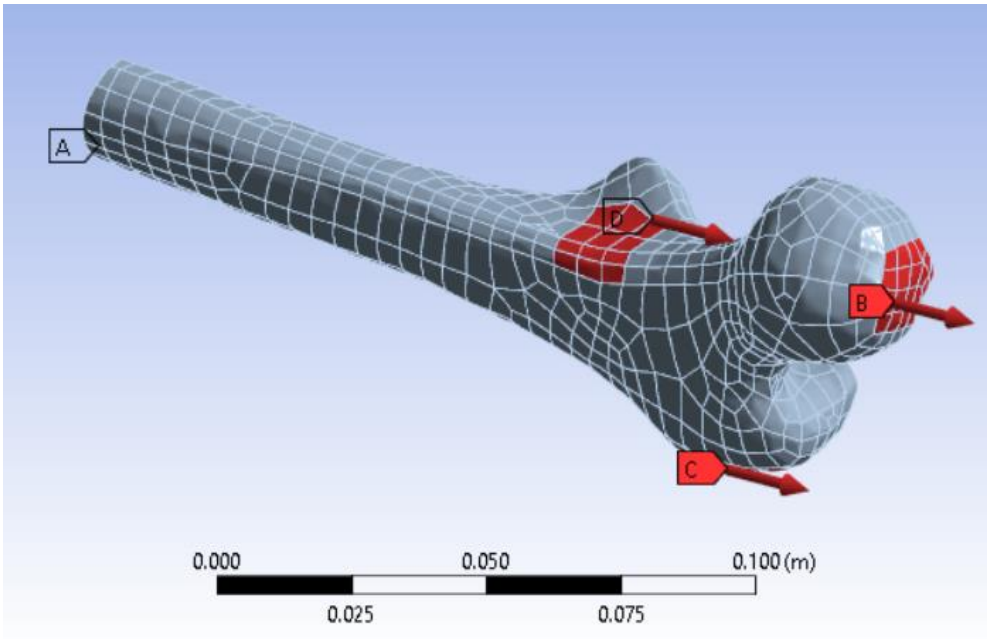



**Figure 8 - View 2 of force locations on half-femur model**

Accuracy was not a priority in this stage of the study. Instead, the main focus was obtaining a dataset which could be effectively interpolated through the use of surrogate modelling. Hence, a mesh convergence study was deemed unnecessary. Instead, an automatically generated mesh was considered adequate with element sizing at 2.5mm. For these reasons, the force locations, directions, and magnitudes were also not meticulously managed, as the consistent theme of this component of the study was to analyse the surrogate modelling methodologies. Thus, any dataset could be used to achieve this. Data that was somewhat relevant to the study was all that was desired.

Another assumption of this model was that it was completely linear. The material was made linearly with an elastic modulus of 1GPa and a Poisson's Ratio of 0.3. Ignoring non-linear behaviours of bone would speed up the simulating process. Although realistically the hip and femoral environment is not strictly elastically linear, it is often considered throughout related literature and anatomical science that the femur's properties can be considered linear when static (Carriero et al, 2010; Yadav et al, 2017; Taylor, 2023). Non-linear scenarios will be explored more in the detailed model.

#### *3.2.1.2 Loading and Simulation – Simplified FE*

The magnitude of the forces was also found through literature (Layton et al, 2022), with a median value being investigated, and then 2 equal increments each side of it to replicate 5 different loading types per force, as shown in Table 1.

**Table 1 - Summary of hip, adductor, and glute force increments for linear FE simulations**

| | Force Value (N) | | | | |
|---|---|---|---|---|---|
| **Muscle** | **Increment 1** | **Increment 2** | **Increment 3** | **Increment 4** | **Increment 5** |
| **Hip** | 1000 | 1200 | 1400 | 1600 | 1800 |
| **Adductor** | 400 | 500 | 600 | 700 | 800 |
| **Glute** | 700 | 800 | 900 | 1000 | 1100 |

With three forces, each having 5 potential load values, it was determined that all possible loading combinations would be simulated, with the desired output of maximal equivalent strain throughout the entire femur. The location of this maximum strain was not considered.

The simulations took approximately 1 minute each to run, eventually constructing a dataset of 125 loading combinations and their corresponding maximal strain output. To summarise, there were three predictor variables and one singular output.

For the special case of the SPM surrogate, only one simulation was required per force being considered to complete the dataset. In this FE model, only three forces were being explored, each unidirectional. Hence only three simulations were required to complete the SPM dataset. During these simulations, the loadings were isolated, specified as 1000N, whilst other loads were held constant at 0N. Trial simulations were then conducted using random loading combinations of the three forces. After simulating, the nodal strain tensors could be extracted and assembled as shown in Eq. 3. The 3 shear strains were easily found; however, the 3 principal strains were not capable of being extracted on a nodal level, and instead the normal strains were obtained. This is an unfortunate limitation of Ansys, and it did prohibit an equal comparison with the other methods since different strain variations were being measured. Despite this, matching nodal tensors of the predicted strains to the tensors of the measured strains obtained from the trial simulations still gave an indication of accuracy. In further applications these strain tensors could be used to find maximal principal strains which may be more useful.

$$\varepsilon_{node} = \begin{bmatrix} \varepsilon_{xx} & \varepsilon_{xy} & \varepsilon_{xz} \\ \varepsilon_{yx} & \varepsilon_{yy} & \varepsilon_{yz} \\ \varepsilon_{zx} & \varepsilon_{yx} & \varepsilon_{zz} \end{bmatrix} \quad [3]$$

### 3.2.2 Detailed FE

Following the construction of the simplified FE model, a more detailed model was desired. Since the surrogate models were already created and developed to handle large numbers of predictor variables and data, in theory any newfound datasets could also be interpolated using the same models.

Hence to increase the relevance of the results, it was hoped that a realistic dataset could be acquired. Thankfully, a more detailed FE model was easily accessible, with the help of two Griffith University PhD students, Alireza Yahyaiee Bavil, and Emmanuel Eghan-Acquah, who were completing a paper titled 'Effect of Different Constraining Boundary Conditions on Simulated Femoral Stresses and Strains During Gait'. By nature of their study an extremely detail FE model was required and adequately constructed with the help of OpenSim software (Simbios, USA) to accurately locate the position of muscle and contact forces throughout the femur. This model considered bone density throughout the femur, and hence introduced some non-linearities in that the material properties were changing.

#### *3.2.2.1 Loading and Simulation – Detailed FEA*

The detailed FE model considered 22 muscle and contact forces, each of which were decomposed into x, y and z components. Hence, 66 total force components could be considered with this model. It was decided that analysing the influence of all forces was outside the scope of the project, and that only three of the most dominant forces would be considered. Again, the hip contact force and adductor muscle force were chosen, along with the knee contact force, which was omitted in the first model, as all distal structures were ignored.

This new model's mesh size was reduced to just 2mm, used quadrilateral elements, and considered the entire femur. It was constrained via external points in space, imitating a femur under zero force when untouched. Constructed with data obtained from segmented CT scans and the aid of OpenSim, the model was considered extremely anatomically accurate.

A small limitation of this new model, however, was that it was built in Abaqus (Dassault Systemes, France), an unfamiliar software. There were challenges in learning how to effectively use this software. Thankfully, with the model built, it was purely the simulating process that needed to be learned. Fig. 9, 10 and 11 outline further outline the detail of this model.


**Figure 9 - Detailed FEM full femur view**

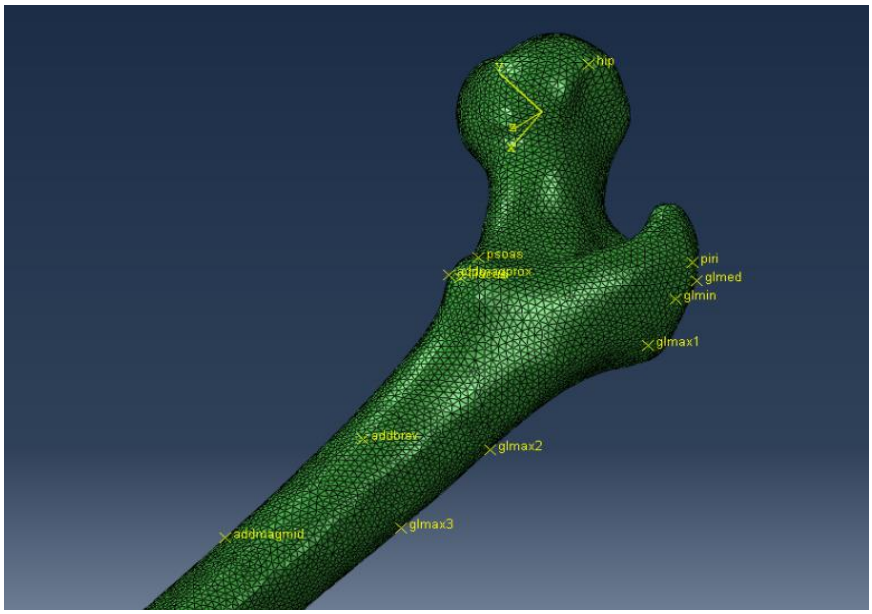



**Figure 10 Detailed FEM proximal view**

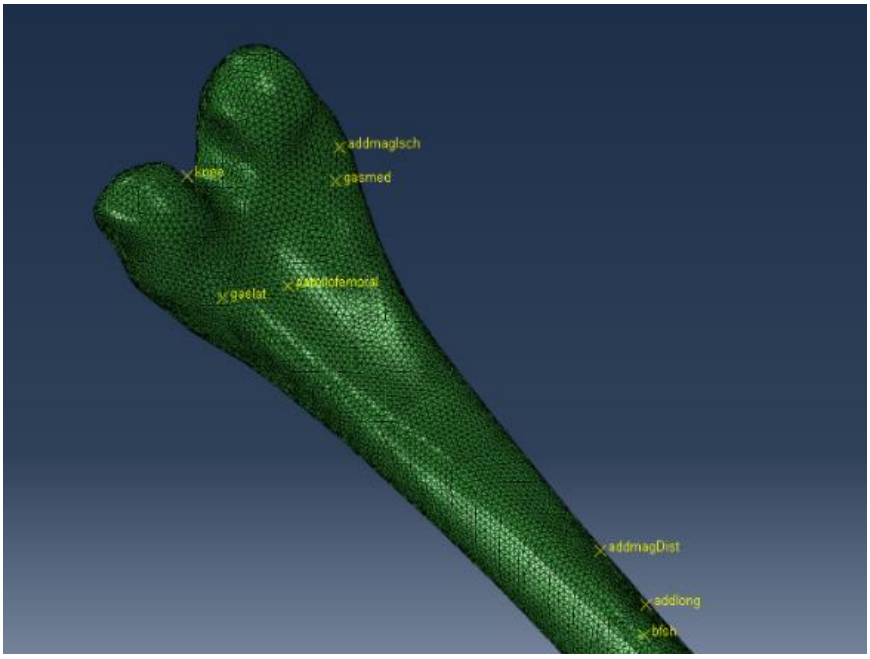



**Figure 11 - Detailed FEM distal view**

The SPM was the only surrogate modelling method that was investigated using the detailed FEM, for reasons explained in the discussion section of this paper. Hence only one simulation was required for each force considered to complete the dataset. In this instance, there were three forces, each with an x, y and z component, thus making 9 total forces. Each force component had x, y, and z normal nodal strains extracted for consistency with the first trial. Simulations were done in alignment with the initial FE model, wherein isolated forces were given 1000N loadings, whilst other forces were held constant at 0. To save time, shear strains were not collected, since it was thought that validating adequate normal strain prediction would indicate the shear strain prediction, since the simple addition of strain tensor components applies to both normal and shear strains. In summary, measuring accuracy in normal strain prediction infers the same accuracy in shear.

## 3.3 Surrogate Construction

All surrogate models were constructed in MATLAB 2023 (MathWorks, USA). Except for the SPM, all techniques had an available toolbox to assist in the execution of complex mathematical operations involved in some techniques. For the first three models, to access the dataset, it was exported from Ansys into Excel, where it was then indexed through MATLAB. This was a simple table which defined the force combinations and corresponding output from each trial. For the SPM, each simulation had nodal measurements extracted

into an Excel file before it could be accessed within MATLAB. Hence, this required the indexing of multiple Excel spreadsheets.

### 3.3.1 Multi-Linear Regression

The MLR surrogate was among the simplest to make. Utilising the '*fitlm*' function in MATLAB (MathWorks, 2013), which assigns a linear equation to datasets with more than one predictor variable.

### 3.3.2 Cubic Splines

Cubic splining was also easily implemented on the dataset using the multivariate splining tool '*interpn'* and specifying *'spline'* as the designated interpolation tool (MathWorks, 2021). This function applies splines constructed with cubic polynomials to the dataset.

### 3.3.3 Kriging

The kriging surrogate was built using the *'fitgrp'* function, which returns a Gaussian Process Regression model trained using the dataset it is provided (MathWorks, 2015). Defined as ordinary kriging, this function could also allow a kriging variance to be calculated for each data point prediction.

### 3.3.4 Superposition Principle Method

Unlike the other methods, the SPM was constructed from scratch. Once the nodal measurements had been taken, they were accessed via Excel and assembled into nodal strain tensors. Simple addition was used to superimpose nodal strain tensors, using multipliers specified by the loading condition being investigated. For example, for the loading combination of -600N, 150N, 780N for the hip, glute, and adductor respectively, the nodal strain tensors of each isolated simulation would be multiplied by -0.6, 0.15, and 0.78, considering the isolated simulations were done using 1000N loads. These results were then compared to known nodal strain solutions obtained by conducting a simulation of the given loading combination in Ansys.

## 3.4 Surrogate Testing and Validation

MLR, splining and kriging surrogates were tested by plotting the predicted femoral strain values against the known ones obtained in the initial data collection phase. To ensure datapoints fed into the model were not being copied as outputs, strain values were predicted between the known data points, but still within the bounds of the dataset.

Since the SPM surrogate considered a different dataset and analysed nodal strains tensors rather than maximal equivalent strains, their outputs were visually validated in a different way. This time, the strain tensor components of a known loading obtained via Ansys were plotted against the ones predicted using the SPM. The linearity of the plot would demonstrate its accuracy.

Root mean square error (RMSE) and normalised root mean square error (nRMSE) were both key validation tools used to verify the accuracy of each surrogate's strain predictions. RMSE is a technique which simply finds the error between the actual strain values and the predicted ones throughout the entire dataset. Since strain values were generally in the domain of $10^{-3}$ to $10^{-5}$, and hence the error values were so tiny they could've been considered negligible. A more realistic quantitative measure was the nRMSE, which normalises the RMSE relative to the raw data, effectively allowing a percentage of accuracy to be measured, which was far more comparable. This made it easier to draw conclusions and make generalisations. These methods were simply mathematical and were easily implemented through MATLAB.

The time taken to for the computer to build, train and use the surrogate to predict strains was also considered. This was done by starting a timer in the initialisation phase of the code, and simply requesting the time at the code's ending. It's recognised that this is not a perfect measure of each technique's CPU consumption, but it does provide a quantifiable measure which can be used to compare the complexity of the different techniques.

# 4. RESULTS

## 4.1 Preliminary Testing

The preliminary result of this study refers to the analyses of the surrogate models when predicting strain values on a simplified FE model; that is, a model created with little consideration for anatomically accurate results, and more regard for an eligible and valid dataset to that could be used to train, test, and validate the surrogate models.

The MLR, splining, and Kriging techniques all considered the same dataset. Hence analysing these techniques relative to each other was easy by simply comparing their nRMSE and CPU runtime. Visually they could also be examined by plotting the actual resultant maximal equivalent strain received using the FE software to the values predicted using the surrogates. Planes were formed to create surfaces of both actual and predicted strains, to indicate where the surrogate was most effective. To enable visual assessment, only two predictor variables could be considered simultaneously, creating a 3-dimensional plot with maximal equivalent strain as the output. Figures 12, 13 and 14 show a comparison of the three initial surrogate modelling techniques when applied to the simple FEM's dataset. These plots consider only the hip and adductor, omitting the glute force to portray results in 3 dimensions.

The SPM method was plotted differently, since this surrogate used a different dataset and measured different strain quantities, with nodal shear and normal strains being obtained rather than the maximal equivalent ones. Hence, validating this model and comparing it with the previous ones was challenging. An nRMSE value and a CPU runtime was still obtainable to quantify the accuracy and computational demand, however visually representing them in the same way as previous methods was not. Instead, a simple visual measure of the SPM's accuracy was to plot the expected nodal strain components to the actual ones. The linearity of the solutions in this case would indicate its accuracy. Fig. 15 shows these results.

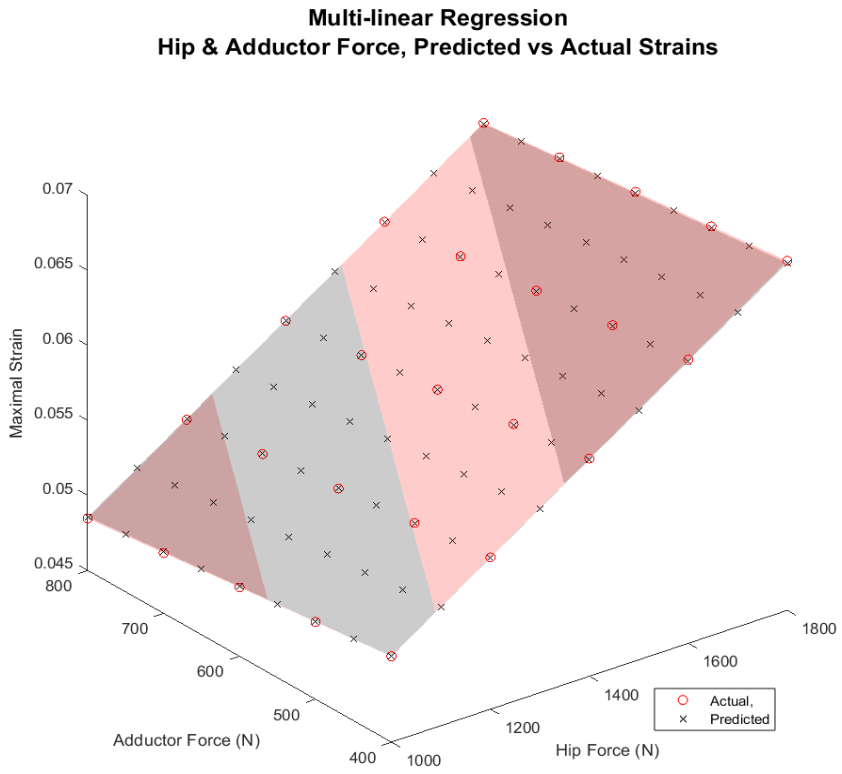



**Figure 13 - MLR surrogate maximum equivalent strain predictions throughout simplified half femur model, plotted against known strains obtained through Ansys.**

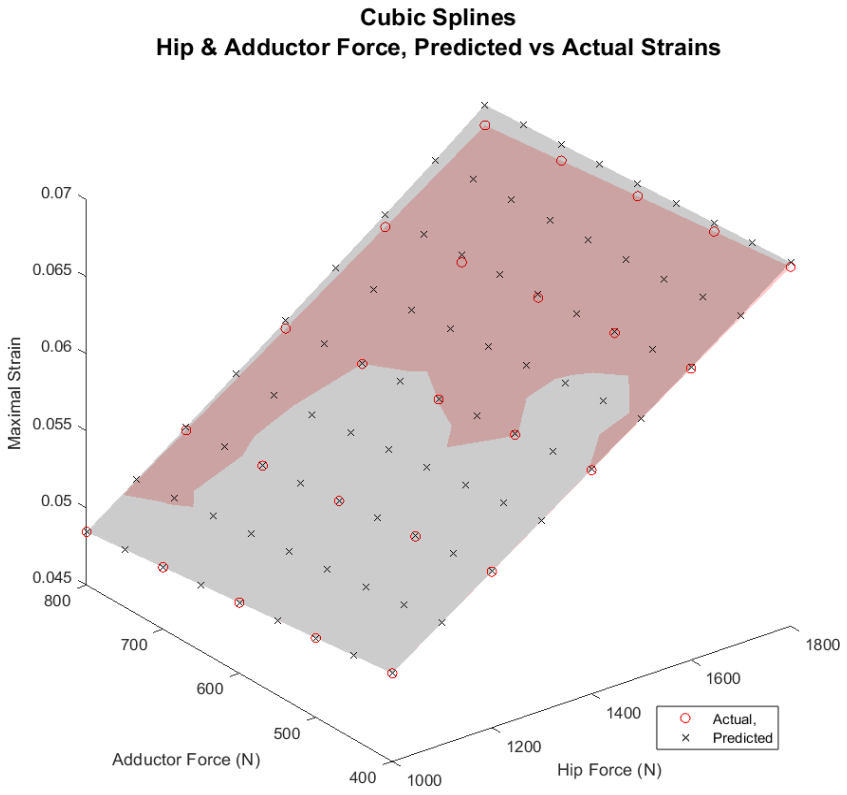



**Figure 12 – Cubic splining surrogate maximum equivalent strain predictions throughout simplified half femur model, plotted against known strains obtained through Ansys.**

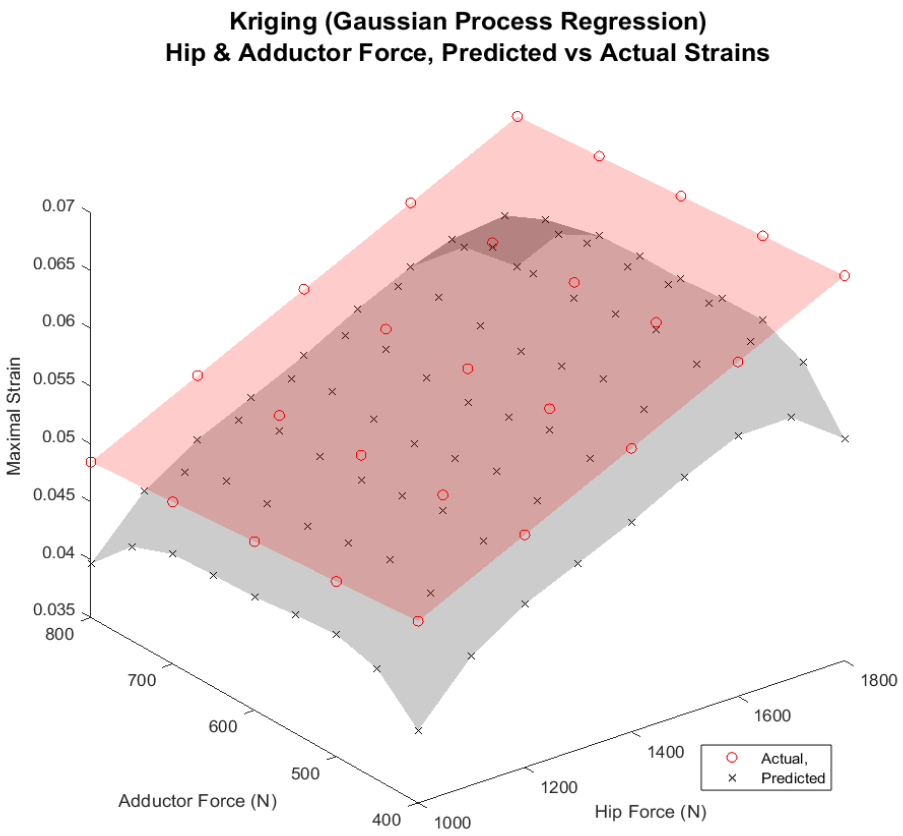



**Figure 14 - Kriging surrogate maximum equivalent strain predictions throughout simplified half femur model, plotted against known strains obtained through Ansys.**

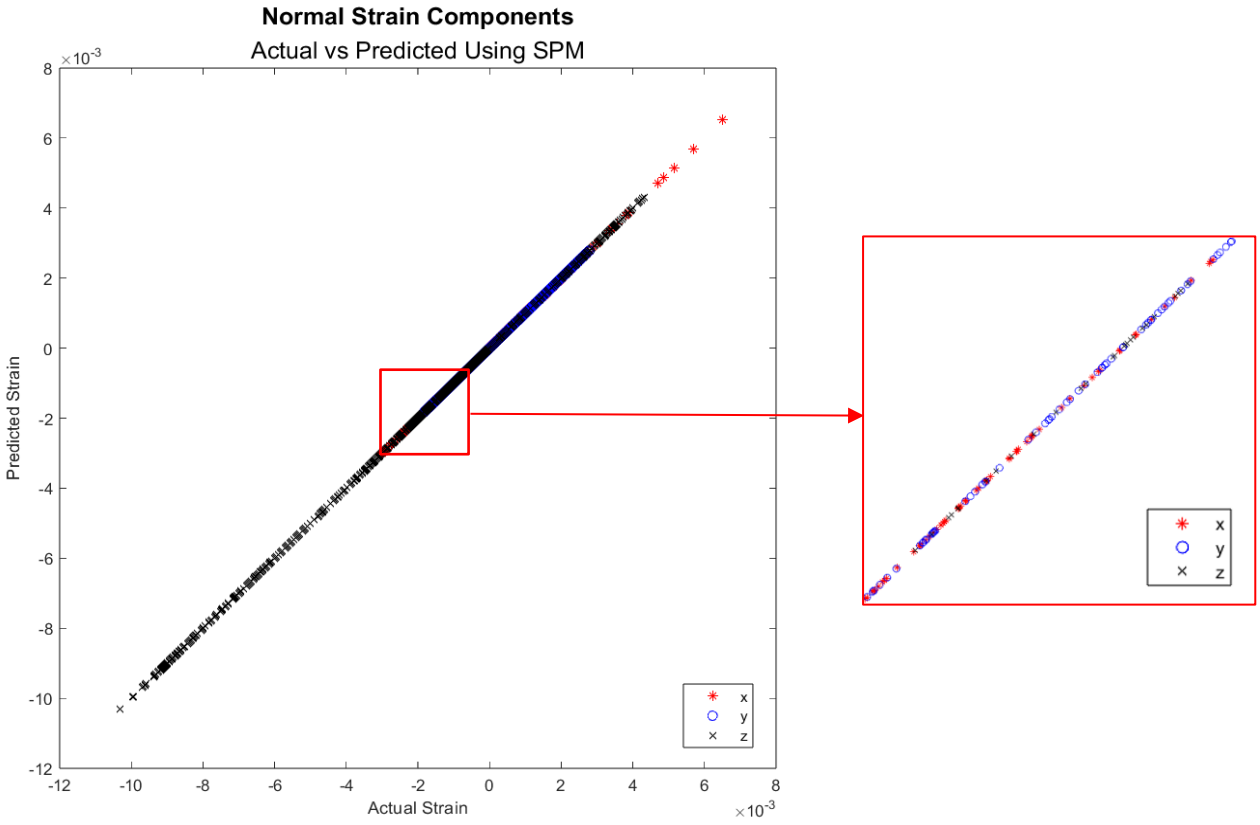



**Figure 15 - Visual representation of the SPM's accuracy in a strictly linear system. Narrowed view for clarity. Predicted normal nodal strains are plotted against known ones.**

Visually, the preliminary results indicated some key findings. However, to reinforce this, some measurable quantities were also found. The nRMSE, calculated using the relative error between the known and predicited strains, and the CPU time, found simply by getting MATLAB to output the time difference between start and finish, were both recorded. Table 2 shows these results:

**Table 2 - Quantification of surrogate accuracy and time consumption**

| | **nRMSE** | **CPU Time (sec)** |
|---|---|---|
| **MLR** | 0.00413 | 14.9 |
| **Cubic Splines** | 0.00782 | **1.50** |
| **Kriging** | 0.157 | 7.42 |
| **SPM** | **0** | 4.25 |

As seen both visually and measurably, the SPM surrogate exhibited the most accuracy, exactly predicting the nodal normal strain components in all directions. It was also required the dataset that was smallest and simplest to collect, totalling just three simulations, in comparison to the 125 needed for the other techniques. Comparably, the SPM surrogate was also fast to train and execute, taking just 4.25 seconds.

The cubic splining technique resulted in very accurate predictions, in the timeliest manner, taking just 1.50 seconds to train and execute. With a normalised root mean square error of just 0.00782 (<0.08%), this method's compatibility with linear systems was also clearly indicated by these results.

The MLR surrogate was the most accurate of the techniques used in the 125-simulation dataset. Aligning with the assumption of a strictly elastic material being used in this FE model, the favourable linear characteristics of this technique were undoubtedly on display here, since these strain predictions were incredibly accurate, exhibiting an nRMSE of just 0.00413 (<0.05%). Interestingly, the method unexpectedly had the longest CPU time of 14.9 seconds. This may be indicative of a limitation in the coding of this surrogate.

Finally, kriging demonstrated poorer accuracy than hypothesised, with a significantly higher nRMSE of 0.157 (>15%). As expected, the variance exhibited by this technique increased as the strain predictions reached the bounds of the known dataset. This can be seen by the dipping corners of the plane in Fig. 14. This occurs due to the decrease in density of known data point distribution, which is a critical part of the mathematics behind this technique. The accuracy of this model was somewhat surprising and did not necessarily align with the theory of kriging being suitable for both linear and non-linear systems. These results posed a question regarding how suitable the 125-simulation dataset was for the given techniques. In retrospect, providing a grid-style set of data gathered at equal intervals, within a strictly linear model would certainly have benefited the techniques based around linearity (MLR and cubic splines) more. In hindsight, a different loading dataset, potentially one which used a hypercube to introduce more variability, may have led to more reliable results. Kriging is also most effective when trained on large, dense datasets (Haeri and Fadaee, 2016; Chu et al, 2020). This was not necessarily available in this study, and hence may have further impacted the accuracy of the resultant strain predictions.

## 4.2 Secondary Testing

### 4.2.1 Justifications

Initially, it was intended that all four surrogate modelling techniques would be trained using both the simplified, linear FE model, followed by the more anatomically detailed, non-linear FE model to compare results. This would allow clear conclusions to be drawn for each technique regarding their strengths and weaknesses. Unfortunately, time constraints led to only limited time towards the end of this study to implement the more detailed FE model.

The decision was made to test only the SPM surrogate model as this had shown the most promising results in the preliminary testing and was the most novel of the techniques. Another advantage of this decision was that an extensive dataset was not required, but rather just one simulation for each isolated loading, totalling just 9. This model was made in Abaqus, an unfamiliar software, and so simplifying this process would be beneficial to the completion of this study.

### 4.2.2 Results

Following the completion of the 9 require simulations for the SPM dataset, some further simulations were conducted to gather the nodal solutions to a known loading combination. This allowed the comparison of the predicted strain components found via the surrogate to

be compared with some known values found via the software. For the results shown, the known loading used for reference solutions is shown in Table 2.

In this trial, only normal strains were gathered to simplify the process. The principles of superposition are the same with shear strains, and so the same results could be expected if they were investigated too.

**Table 3 - Known loading combination parameters**

| **Force Component** | ***Hip*** | | | ***Knee*** | | | ***Adductor*** | | |
|---|---|---|---|---|---|---|---|---|---|
| | X | Y | Z | X | Y | Z | X | Y | Z |
| **Magnitude** (N) | -500 | 250 | -300 | -120 | -400 | 20 | 550 | 210 | 10 |

The predicted normal nodal strain components were all plotted on the same axis, against their actual values obtained by solving the system with specifications shown in Table 2. This can be seen in Fig. 16. For simplicity the individual components are also plotted against themselves, as shown in Fig. 17, 18 and 19.

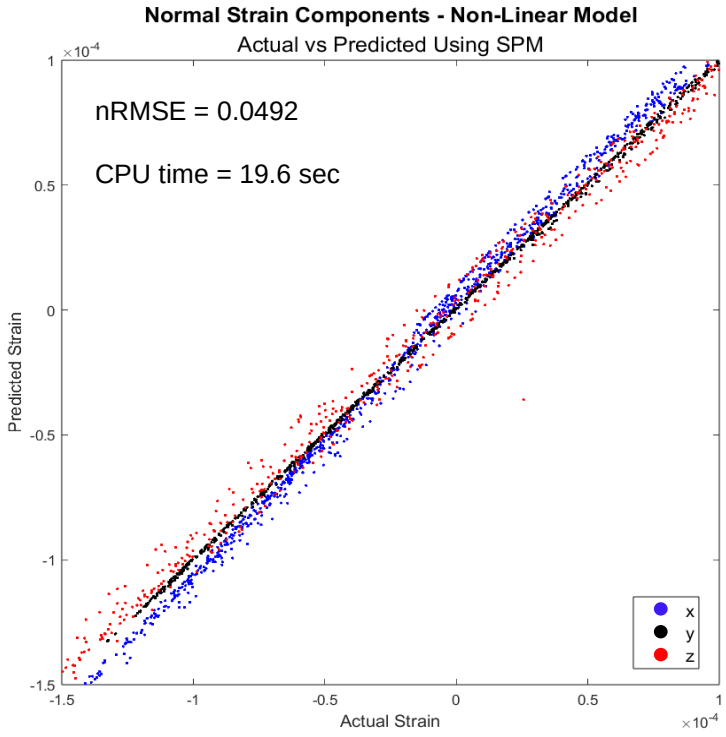



**Figure 16 - Visual representation of SPM's accuracy in a system that is non-linear. Nodal strain components are colour coded for clarity.**

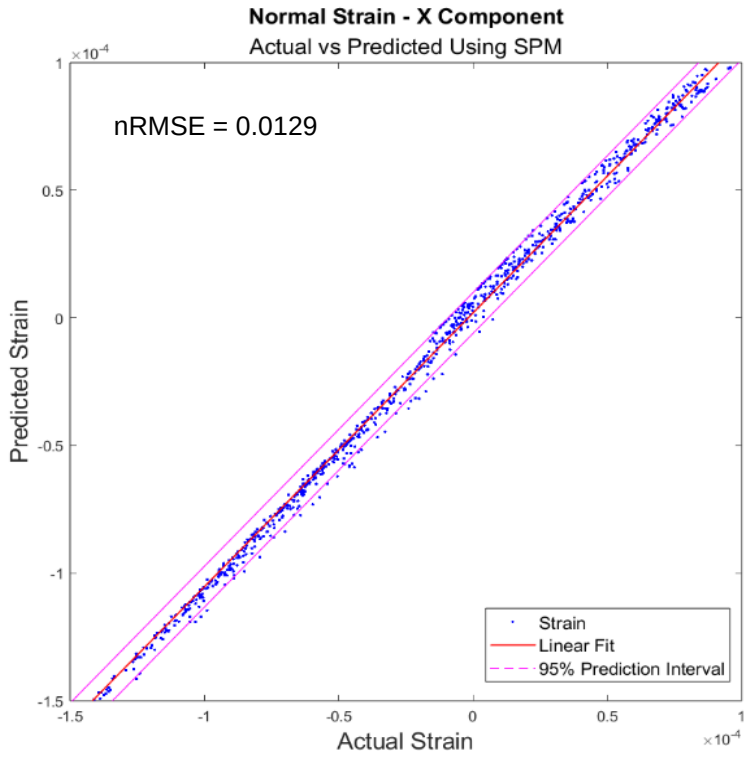



**Figure 18 - X component of nodal strain predictions, plotted against actual values. Also plotted is a 95% confidence interval of the data.**

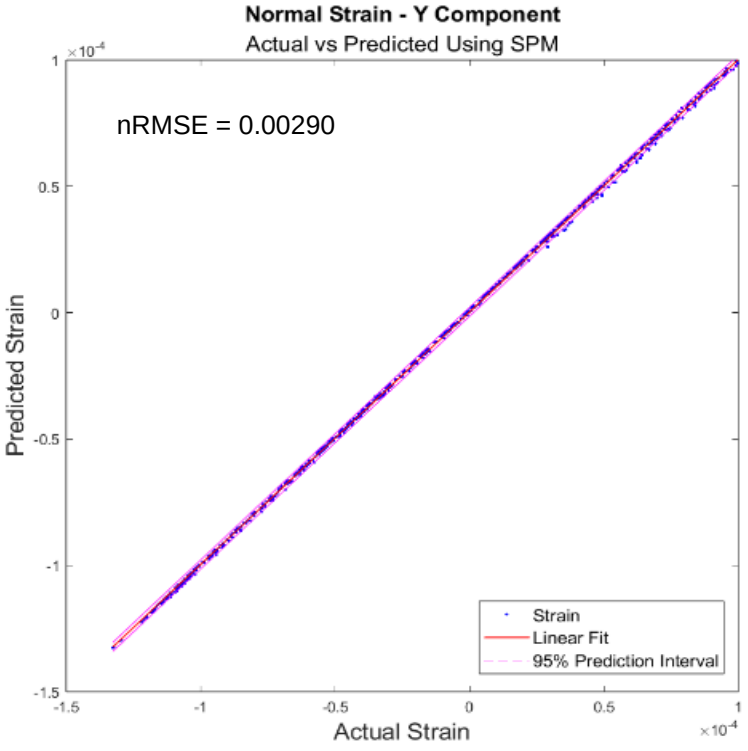



**Figure 17 - Y component of nodal strain predictions, plotted against actual values. Also plotted is a 95% confidence interval of the data.**

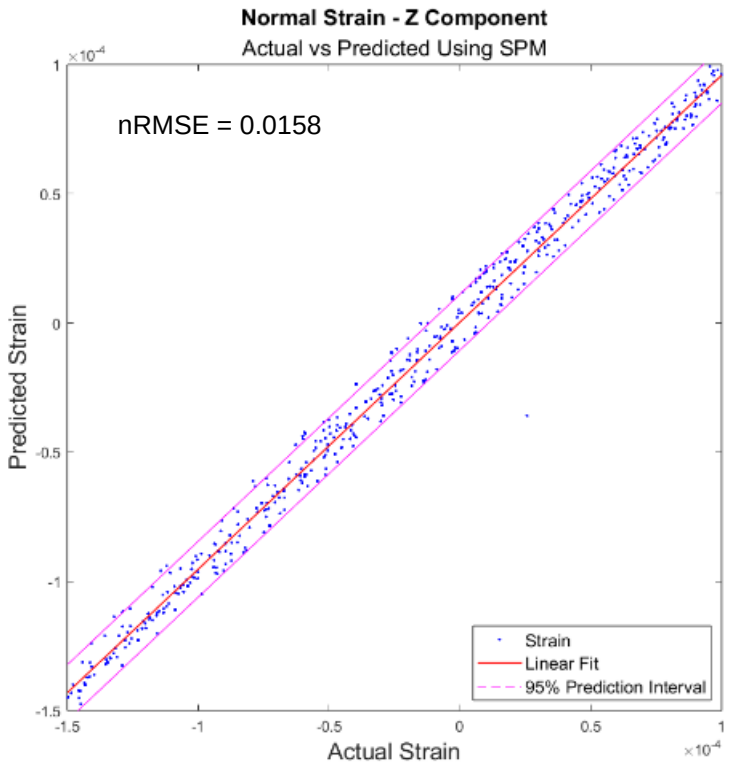



**Figure 19 - Z component of nodal strain predictions, plotted against actual values. Also plotted is a 95% confidence interval of the data.**

In applying the SPM to a more detailed model, it was found that the non-linear properties of the material were not a major limitation, which was the biggest concern going into this analysis. Although nodal strain predictions were no longer exact, there was still a very accurate correlation between the predicted solutions and the actual ones. Overall, this technique exhibited an nRMSE of 0.0492 (4.92%), whilst the sub-components were predicted with an nRMSE of 0.0129 (1.29%), 0.00290 (0.29%) and 0.0158 (1.58%) respectively in x, y, and z directions. The surrogate model was built, trained, and executed in a time of approximately 19.6 seconds, making it slower than all methods in the preliminary testing phase, however this was to be expected given the model was not exactly elastically linear, which typically poses longer simulating times in FE studies. There were also significantly more nodes in this model (97,000 compared to 27,000), however this was not much of a limitation, since only the femoral neck nodes were selected, and hence in both SPM trials the node count was approximately 2000, and the resultant time loss was considered negligible.

# 5. DISCUSSION

## 5.1 Summary of Results

The preliminary results indicated a few key findings. Firstly, it was obvious that in the linearly elastic model, with the gridded dataset, mathematically linear surrogate modelling techniques were most applicable. Aside from the SPM, the MLR and splining surrogates showed extremely high accuracy. Kriging did not present particularly valuable predictions, with significantly higher error appearing systematically across the distribution. This was contradictory to the existing literature, which suggested that kriging would provide accurate results when applied to both linear and non-linear systems (Taylor et al, 2016; Gaspar et al, 2014; Haeri and Fadae, 2016). Whilst splining was the fastest of the preliminary testing techniques, all techniques took less than 15 seconds using the initial dataset. This is significantly faster than average FE simulation, which is generally in the range of 120-180 seconds (Basafa et al, 2013), and so these measures were not considered vitally important.

Preliminary SPM testing revealed that it could provide exact strain predictions, given the system is completely linear, matching the little literature that was available (Ziaei Poor et al, 2019). Taking just 4.25 seconds to run and execute, this technique was by far the most promising of the four, which provoked a second look at how effective this method might be by applying a dataset from a non-elastically linear FE model. Results from this ‘secondary’ study concluded again that the SPM would be a commendable surrogate modelling technique, providing accurate, yet no longer exact solutions in 19.6 seconds.

The results from both the preliminary and secondary study both indicate that the SPM would be the recommended surrogate modelling technique when choosing between the four investigated. That said, there are some limitations and potential sources of error encountered throughout the study which should be considered.

## 5.2 Discussion of Errors and Limitations

The first limitation of this study were the time constraints. Ideally, all four surrogate modelling techniques would be subject to testing on both the linear and non-linear model. Not only does this increase the sample size of the study, improving reliability, but it would also expose some of the techniques strong with linear systems to a non-linear dataset. For example, it would be hypothesised that the MLR and splining methods would instead be less accurate

than kriging when applied to this model. Unfortunately, time constraints and lack of familiarity with the Abaqus software resulted in cutting these project objectives out. Despite this, the findings shown within the SPM are still very valuable, particularly since it such an under-researched technique.

Secondly, with the nature of the SPM surrogate requiring nodal solutions, whilst the others considered equivalent maximal strains, comparing the SPM to other methods was challenging. In other surrogates, many equivalent strain solutions of known loadings were found, which could easily be interpolated using their respective method. Acquiring nodal solutions for such a large number of known loading combinations was simply too time consuming and would have also contradicted the principles of superposition which state that in a linear system, once isolated solutions had been acquired, any combined loading's nodal strain components could be predicted (Ziaei Poor et al, 2019). The results perfectly demonstrated this. Although this output is useful, though, it is difficult to compare to other techniques. Quantifiable measures such as nRMSE and CPU time were used to provide some detail on how accurate the SPM was in reference to the other techniques, however there is also some common sense that applies when trying to identify which method is best suited.

The predicted strains found via Kriging were uncharacteristically inaccurate. Although exhibiting some of the common themes of gaussian process regression, such as deviation at the bounds of known data distribution, there appeared to be a systematic difference between the predicted and actual equivalent strain values. Literature based around this technique indicates that it should be adequate in both linear and non-linear applications (Eskinazi and Fregly, 2015), however it requires a dataset larger than the one used in this study to be considered accurate. Since the principles behind Kriging allow it to consider both the magnitude and location of data points in the training set, it's also thought that the equidistant training data points used in this study were a hindrance and would have instead benefited the linear mathematical operatives (MLR and splining).

Obtaining realistic loading conditions for the femur proved to be a challenging task. In the simplified model, literature was used to find an approximate value of each muscle's force contribution to the femur itself (Layton et al, 2022). This was not considered vitally important in the first model since the focus here was more on the successful execution of the surrogate models themselves, and hence the dataset's origin was not relevant. However, when attempting a more realistic FE model, anatomically accurate force values were desirable to

increase the reliability of the results in the given context. Finding such values proved challenging. Another Griffith PhD student, Claire Crossley, was able to assist in providing some force measurements obtained through her study in reclined cycling. These force measurements were used as a baseline to replicate realistic loadings on the femur, and provide a better context to the study, regarding femoral fracture prediction.

## 5.3 Application

This study was aligned with the BioSpine clinical trial currently being held at Griffith University (BioSpine, 2023). In this trial, participants with SCI take part in FES therapy, combined with neurological technology to stimulate the subject's leg muscles when they voluntarily choose to. This effectively replicates a voluntary movement.

A fundamental component of this trial is calculating how much strain a subject's femur may be able to handle under excitation, to avoid injury due to overstimulation. Lower-limb injuries in this demographic can be particularly detrimental due to their lack of sensory feedback. Whilst FE models can accurately predict the femoral strains following CT segmentation, it is an incredibly time-intensive process. Creating a series of FE models and using a trial-and-error process is the best currently available tool used within the BioSpine clinic to allow a successful FES stimulation parameter set to be found without risking harm for the patient.

The application of findings in this study may help to reduce the time spent continuously running FE simulations, by creating an initial dataset and then using surrogate interpolation to find strain responses to unknown loadings. This removes the need for the trial-and-error process, whilst maintaining accurate results, ultimately saving time for the clinicians, and improving the experience of the participant.

## 5.4 Future Work and Project Improvement

The key limitation to this study was the inability to test all four surrogate models in both a linear and a non-linear capacity. Although it is said with high confidence that the SPM is truly the most suitable of the techniques considered in this study, in future it would be beneficial to implement a non-linear dataset in all surrogates for accuracy and reliability.

# 6. CONCLUSION

This study provided an analysis of four leading surrogate modelling techniques, MLR, splining, Kriging, and SPM, with respect to their effectiveness in predicting femoral strains when unknown loading combinations are applied. The surrogates were validated by comparing predicted results to ones obtained by solving the loading combinations using Abaqus or Ansys software. Normalised root mean square error and the total CPU time taken to construct, train and execute the surrogate model were considered in the analysis.

Initially, the techniques were trained using a dataset obtained from an elastically linear model. This revealed that the SPM would be most suitable for the task, providing exact predictions in just 4.25 seconds. Splining and MLR also exhibited highly accurate answers, although they were not exact. Kriging posed the highest nRMSE value of >15% indicating a surprising lack of accuracy, which did not necessarily align with the known literature. It's believed that a small, highly linear dataset contributed to this contradictory result.

The most promising and novel of these techniques, the SPM, was then further investigated by applying a dataset obtained from a more realistic, elastically non-linear model. Results from this analysis showed that the non-linear nature of the data was not a significant limitation, although exactly accurate answers were no longer attainable. The predictions exhibited an nRMSE of 4.92%, which suggests a reasonably high level of accuracy, particularly considering the non-linear nature of the data. Unfortunately, time constraints prevented the testing of other techniques using this model, which may have better indicated their application in this context.

From the results, it can be said with confidence that the SPM is the recommended surrogate modelling technique to use when predicting femoral strains. Its ability to predict exact strains in a linear system, which is commonly inferred in many femoral applications, and CPU time consumption very similar to the other techniques, makes it the obvious choice of the four methods investigated. It is hoped that these results can be applied to some of the works being completed at BioSpine, to ensure patient safety whilst reducing the loss of time in the clinic.

# BIBLIOGRAPHY


Abaqus, Dassault Systemes, 2023, https://www.3ds.com/edu/education/students/solutions/abaqus-le/

Alexopoulous, E., *Introduction to Multivariate Regression Analysis*, December 2010.

Ansys Inc, USA, 2023, https://www.ansys.com/

Bartlett, R., *Artificial Intelligence in Sports Biomechanics: New Dawn or False Hope?*, Journal of Sports Science & Medicine, Pages 474-479, December 2006

Bekesiene, S., Meidute-Kavaliuskiene, I., Vasiliauskiene, V., *Accurate Prediction of Concentration Changes in Ozone as an Air Pollutant by Multiple Linear Regression and Artificial Neural Networks*, https://doi.org/10.3390/math9040356, 2021

Bennet, J., Das, J., Emmady, P., *Spinal Cord Injuries*, StatPearls Publishing Co., May 2011

Bessho, M., Ohnishi, I., Matsuyama, K., Matsumoto, T., Imaim K., Nakamura, K., *Prediction of strength and strain of the proximal femur by a CT-based finite element method*, Journal of Biomechanics, 40(8), Pg 1745-1753, 2007, https://doi.org/10.1016/j.jbiomech.2006.08.003

BioSpine Clinical Trial, Griffith University, *Digitally-enabled Rehabilitation For Spinal Cord Injury,* 2023, https://www.griffith.edu.au/menzies-health-institute-queensland/research-trials/biospine-study

Biran, A., *Chapter 7 – Cubic Splines*, Geometry for Naval Architects, Pages 305-324, 2019

Bitsakos, C., Kerner, J., Fisher, I., Amis, A., *The effect of muscle loading on the simulation of bone remodelling in the proximal femur*, Journal of Biomechanics, 38(1), Pg 133-139, January 2005, https://doi.org/10.1016/j.jbiomech.2004.03.005

Burge, R., Dawson-Hughes, B., Solomon, D., Wong., J., King., A., Tosteson, A., *Incidence and Economic Burden of Osteoporosis-Related Fractures in the United States 2005-2025*, Journal of Bone and Minearl Research Vol 22, November 2007

Carriero, A., Jonkers, I., Shefelbine, S., *Mechanobiological prediction of proximal femoral deformities in children with cerebral palsy*, March 2010, DOI:10.1080/10255841003682505,

Chu, L., Shi, J., Souza de Cursi, *Efficiency improvement of Kriging surrogate model by subset simulation in implicit expression problems*. Comp. Appl. Math. 39, 119 (2020). https://doi.org/10.1007/s40314-020-01147-1, July 2018

Chung, CJ.F., Fabbri, A.G., Van Westen, C.J. (1995). Multivariate Regression Analysis for Landslide Hazard Zonation. In: Carrara, A., Guzzetti, F. (eds) Geographical Information Systems in Assessing Natural Hazards. Advances in Natural and Technological Hazards Research, vol 5. Springer, Dordrecht. https://doi.org/10.1007/978-94-015-8404-3_7

Duda, G., Heller, M., Albinger, K., Schulz, O., Schneider, E., Claes, L., *Influence of muscle forces on femoral strain distribution*, Journal of Biomechanics, 31(9), Pg 841-846, September 1998, https://doi.org/10.1016/S0021-9290(98)00080-3

Eskinazi, I., Fregly, B., *Surrogate modeling of deformable joint contact using artificial neural networks*, Medical Engineering & Physics, Vol. 37 Issue 9, Pages 885-891, September 2015

Farin, G., *Chapter 8 – B-Spline Curves,* Curves and Surfaces for CAGF (Fifth Edition), Morgan Kaufmann, Pages 119-146, 2002

Florshutz, A., Langford, J., Haidukewych, G., Koval, K., *Femoral neck fractures: current management*, March 2015, DOI: 10.1097/BOT.0000000000000291

Gaspar, B., Teixeira, A., Soares, C., *Assessment of the efficiency of Kriging surrogate models for structural reliability analysis,* Probabilistic Engineering Mechanics Vol. 37, Pages 24-34, July 2014

Goldman, R., *Chapter 7 – B-Spline Approximation and the de Boor Algorithm*, Pyramid Algorithms, Morgan Kaufmann, Pages 347-443, https://doi.org/10.1016/B978-155860354-7/50008-8, 2003

Habermann, C., Kindermann, F., *Multidimensional Spline Interpolation: Theory and Applications*, DOI 10.1007/s10614-007-9092-4, May 2007

Harmening, C., *Spatio-temporal deformation analysis using enhanced B-Spline models of laser scanning point clouds*, Vienna, Austria, 2020

Hay, L., and Viger, R., *Precipitation interpolation in mountainous regions using multiple linear regression*, Denver Federal Centre, 1998

Hayes, A., *Multiple Linear Regression (MLR) Definition, Formula and Example,* April 2023.

Imai, K., *Recent Methods for Assessing Osteoporosis and Fracture Risk,* Vol. 8 Issue 1, Pages 48-59, 2014

Jenkins, S., Harrington, M., Zavatsky, A., O'Connor, J., Theologis T., *Femoral muscle attachment locations in children and adults, and their prediction from clinical measurement*, Gait & Posture, 18(1), Pg 13-22, August 2003, https://doi.org/10.1016/S0966-6362(02)00137-6

Kenedi, P., Riagusoff, I., *Stress development at human femur by muscle forces*, Journal of the Brazilian Society of Mechanical Sciences and Engineering, 37, Pg 31-43, 2015, https://doi.org/10.1007/s40430-014-0164-9

Kosinka, J., Sabin, M., Dodgson, N., *Creases and Boundary Conditions for Subdivision Curves*, DOI: 10.1016/j.gmod.2014.03.004, September 2014

Kumar, V., Sharma, A., Cerda, A., *Heavy Metals in the Environment,* Published 2020.

Layton, R., Messenger, N., Stewart, T., *Characteristics of hip joint reaction forces during a range of activities*, Medical Engineering & Physics, 108, October 2022, https://doi.org/10.1016/j.medengphy.2022.103894

Liang, L., Liu, M., Martin, C., Sun, W., *A deep learning approach to estimate stress distribution: a fast and accurate surrogate of finite-element analysis*, https://doi.org/10.1098/rsif.2017.0844, January 2018

Martelli, S., Kersh, M., Pandy, M., *Sensitivity of femoral strain calculations to anatomical scaling errors in musculoskeletal models of movement*, Journal of Biomechanics, Vol. 48 Issue 13, October 2015

Martin, R., Sadowsky, C., Obst, K., Meyer, B., McDonald, K., *Functional Eletrical Stimulation In Spinal Cord Injury*, DOI: 10.1310/sci1801-28, 2012

Matheron, G., *The Intrinsic Random Functions and their Applications,* 1973

MATLAB, MathWorks, Massachusetts, USA, 2023, https://au.mathworks.com/products/matlab.html

MATLAB, Mathworks, *Fit a Gaussian process regression (GPR) model*, *interpn*, 2015, https://au.mathworks.com/help/stats/fitrgp.html#mw_24df21fa-3e4a-463b-97aa-3e5cbe1bea4b

MATLAB, Mathworks, *Fit Linear Regression Model*, *fitlm*, 2013 https://au.mathworks.com/help/stats/fitlm.html

MATLAB, Mathworks, *Interpolation for 1-D, 2-D, 3-D, and N-D gridded data in ndgrid format*, *interpn*, 2021, https://au.mathworks.com/help/matlab/ref/interpn.html#mw_6208770a-3b10-4532-aa2f-f90ad8830d7e

MATLAB, *Surrogate Optimisation*, 2023, https://au.mathworks.com/discovery/surrogate-optimization.html

McClaren, R., *Chapter 10 – Interpolation*, Computational Nuclear Engineering and Radiological Science Using Python, Pages 173-192, Academic Press, https://doi.org/10.1016/B978-0-12-812253-2.00012-1, 2018

Merloz, P., *Optimization of perioperative management of proximal femoral fracture in the elderly*, Orthopaedics & Traumatology: Surgery & Research, Vol 104, Pg 25-30, 2018, https://doi.org/10.1016/j.otsr.2017.04.020

Mostoufi, N., Constantinides, A., *Chapter 3 – Finite difference methods and interpolation,* Applied Numerical Methods for Chemical Engineers, Pages 137-178, Academic Press, https://doi.org/10.1016/B978-0-12-822961-3.00003-0, 2023

OpenSim, Simbios, USA, 2023, https://simtk.org/projects/opensim

O'Rourke, D., Martelli, S., Bottema, M., Taylor, M., *A Comopoutational Efficient Method to Assess the Sensitivity of Finite-Element Models: An Illustration With the Hemipelvis*, Journal of Biomechanical Engineering, DOI:10.1115/1.4034831, September 2016

Phillips, G. and Taylor, P., *Chapter 6 – Splines and Other Approximations*, Theory and Applications of Numerical Analysis (Second Edition), Academic Press, Pages 131-159, https://doi.org/10.1016/B978-012553560-1/50007-0, 1996

Pizzolato, C., Reggiani, M., Saxby, D., Ceseracciu, E., Modenese, L., Lloyd, D., *Biofeedback for Gait Retraining Based on Real-Time Estimation of Tibiofemoral Joint Contact Forces*, IEEE Transactions on Neural Systems and Rehabilitation Engineering, Vol 25, September 2017

Pizzolato, C., Shim, V., Lloyd, D., Devaprakash, D., Obst, S., Newsham-West, R., Graham, D., Besier, T., Zheng, M., Barrett, R., *Targeted Achilles Tendon Training and Rehabilitation Using Personalized and Real-Time Multiscale Models of the Neuromusculoskeletal System*, DOI:10.3389/fbioe.2020.00878, August 2020

Rebholz. B., and Almekkawy, M., *Efficacy Of Kriging Interpolation In Ultrasound Imaging; Subsample Displacement Estimation*, DOI: 10.1109/EMBC44109.2020.9175457, July 2020

Sartori, M., Reggiani, M, van den Bogert, A, Lloyd, D., *Estimation of musculotendon kinematics in large musculoskeletal models using multidimensional B-splines*, Journal of Biomechanics Vol 45 Issue 3, Pages 595-601, https://doi.org/10.1016/j.jbiomech.2011.10.040, February 2012

Schöllhorn, W., *Applications of artificial neural nets in clinical biomechanics,* Clinical Biomechanics Vol. 19 Issue 9, Pages 876-898, November 2004.

Singh, A., Rana, M., Pal, B., Datta, P., Majumder, S., Roychowdhury, A., *Patient-specific femoral implant design using metamaterials for improving load transfer at proximal-lateral region of femur*, Medical Engineering & Physics Vol. 113, March 2023.

SpinalCure, *Spinal Cord Injury In Australia,* AlphaBeta, December 2020

Stockemer, D., *Multivariate Regression Analysis,* Quantitative Methods for the Social Sciences, Pages 163-174, November 2018

Taylor, M., Prendergast, P., *Four decades of finite element analysis of orthopaedic devices: Where are we now and what are the opportunities?,* Journal of Biomechanics Vol. 48 Issue 5, Pages 767-778, March 2015

Wang, J., *An Intuitive tutorial to Gaussian Processes Regression*, Ingenuity Labs Research Institute, 2022

Wolberg, G., *Cubic Spline Interpolation: A Review,* Columbia University, September 1988

Yadav, P., Shefelbine, S., Ponten, E., Gutierrez-Farewik, E., *Influence of muscle groups' activation on proximal femoral growth tendency*, June 2017, doi: 10.1007/s10237-017-0925-3

Zeng, w., Liu, Y., Hou, X., *Biomechanical evaluation of internal fixation implants for femoral neck fractures: A comparative finite element analysis*, August 2020, Computer Methods and programs in Biomedicine, Vol 196, https://doi.org/10.1016/j.cmpb.2020.105714

Ziaei Poor, H. (a), Taylor, M., Pandy, M., Martelli, S., A novel training-free method for real-time prediction of femoral strain, Journal of Biomechanics, Vol. 86, Pages 110-116, March 2019

Ziaei Poor, H. (b), *Calculation of femoral strain during normal activities using efficient computational methods*, Flinders University, College of Science and Engineering, 2019.

# APPENDICES

## Appendix A – Multi-linear Regression Surrogate Model: MATLAB Script

```
% code was created by Thomas Rundle, Flinders University
clc;
clear all;
load HalfFemurParanalysis.mat
tStart = cputime;

%% Collecting and assigning simulated data
hipRaw = -HalfFemurParanalysis.HipCF;            % Input 1
adductRaw = HalfFemurParanalysis.AdductorMF;   % Input 2
gluteRaw = HalfFemurParanalysis.GluteMF;         % Input 3
strainsRaw = HalfFemurParanalysis.StrainMax;     % Output

%% Simplifying into a single matrix
X = [hipRaw, adductRaw, gluteRaw];

%% Fitting a linear regression model
model = fitlm(X, strainsRaw);      % Finds the expected coefficients of y = b0 + b1x1 +
b2x2...


%% Plotting the model
plot(model)                        % y = predicted strains, x = b0 + b1x1 + b2x2.....
title('Multi-linear Regression Plot of Glute MF, Adductor MF, and Hip CF on Femoral
Neck Strain')
ylabel('Maximum Strain (mm^-1)')
xlabel('b0 + b2*hipCF + b3*adductorMF + b4*gluteMF')

%% useful for comparing unique results to other methods

hipCF_load = 1200;               % Setting predictor variables
adductorMF_load = 500;
gluteMF_load = 900;


predictedStrain = table2array(model.Coefficients(1,1)) + ... +
    table2array(model.Coefficients(2,1))*hipCF_load + ...
    table2array(model.Coefficients(3,1))*adductorMF_load + ... +
    table2array(model.Coefficients(4,1))*gluteMF_load;

%% storing the predicted strains so they can be plotted
vecLocation = 1;
hipCount = 1000;
adductCount = 400;
gluteCount = 700;
predictedStrainVector = zeros(1,length(strainsRaw));
queryMatrix = zeros(length(strainsRaw), 4);

% loop for interpolating more data points, and assigning to 'queryMatrix'
for i = 0:8
    for j = 0:8
        for k = 0:8
            gluteValue = gluteCount+k*50;
            adductValue = adductCount + j*50;
            hipValue = hipCount + i*100;
            k = k+1;
            strainValue = table2array(model.Coefficients(1,1)) + ... +
```

```
                table2array(model.Coefficients(2,1))*hipValue + ...
                table2array(model.Coefficients(3,1))*adductValue + ... +
                table2array(model.Coefficients(4,1))*gluteValue;
            predictedStrainVector(vecLocation) = strainValue;
            queryMatrix(vecLocation, 1) = hipValue;
            queryMatrix(vecLocation, 2) = adductValue;
            queryMatrix(vecLocation, 3) = gluteValue;
            queryMatrix(vecLocation, 4) = strainValue;
            vecLocation = vecLocation + 1;
        end
        j = j+1;
    end
    i = i+1;
end

%% plotting a singular surface for validation
% in this instance, we will isolating the hip and adductor, holding the
% glute force constant

surfVector = zeros(length(strainsRaw)/5,4);
surfPredictedVec = zeros(length(queryMatrix)/9,3);
surfVecLocation = 1;
surfPredictedVecLocation = 1;

% looping through points to find suitable locations where glute is constant
for l = 1:5:length(strainsRaw)
    surfVector(surfVecLocation, 1) = hipRaw(l);
    surfVector(surfVecLocation, 2) = adductRaw(l);
    surfVector(surfVecLocation, 3) = strainsRaw(l);
    surfVector(surfVecLocation, 4) = predictedStrainVector(l);
    surfVecLocation = surfVecLocation + 1;
end

% setting up a surface for this plot
X_raw = surfVector(:, 1);
Y_raw = surfVector(:, 2);
Z_raw = surfVector(:, 3);

% reshape the data for plotting as a surface
num_X = numel(unique(X_raw));
num_Y = numel(unique(Y_raw));
X_raw = reshape(X_raw, num_Y, num_X);
Y_raw = reshape(Y_raw, num_Y, num_X);
Z_raw = reshape(Z_raw, num_Y, num_X);

% doing the same for the much larger queryMatrix, which has far more values
for m = 1:9:length(queryMatrix)
    surfPredictedVec(surfPredictedVecLocation, 1) = queryMatrix(m, 1);
    surfPredictedVec(surfPredictedVecLocation, 2) = queryMatrix(m, 2);
    surfPredictedVec(surfPredictedVecLocation, 3) = queryMatrix(m, 4);
    surfPredictedVecLocation = surfPredictedVecLocation + 1;
end

% creating a surface for queryMatrix
X_interp = surfPredictedVec(:, 1);
Y_interp = surfPredictedVec(:, 2);
Z_interp = surfPredictedVec(:, 3);

% reshape the data to plot as surface
num_X_interp = numel(unique(X_interp));
num_Y_interp = numel(unique(Y_interp));
X_interp = reshape(X_interp, num_Y_interp, num_X_interp);
```

```
Y_interp = reshape(Y_interp, num_Y_interp, num_X_interp);
Z_interp = reshape(Z_interp, num_Y_interp, num_X_interp);


% plotting the two datasets, surfVector and surfPredictedVector, which
% represents the raw data plotted against the interpolated data, and also
% plotting the two surfaces
figure
plot3(surfVector(:,1), surfVector(:,2), surfVector(:,3), 'ro')
hold on
plot3(surfPredictedVec(:,1), surfPredictedVec(:,2), surfPredictedVec(:,3), 'kx')
hold on
surf(X_raw, Y_raw, Z_raw, 'FaceColor', 'r', 'EdgeColor', 'none');
alpha(0.2);
hold on
surf(X_interp, Y_interp, Z_interp, 'FaceColor', 'k', 'EdgeColor', 'none');
alpha(0.2);
xlabel('Hip Force (N)');
ylabel('Adductor Force (N)');
zlabel('Maximal Strain');
title({'Multi-linear Regression',...
    'Hip & Adductor Force, Predicted vs Actual Strains'})
legend('Actual,','Predicted', 'Location', 'southeast')
ax = gca;
ax.TitleFontSizeMultiplier = 1.5;


%% Calculating RMSE

rmseVector = zeros(length(queryMatrix), 1);

for p = 1:length(queryMatrix);
    if mod(queryMatrix(p,3), 100) == 50
        continue
    end
    if mod(queryMatrix(p,2), 100) == 50
        continue
    end
    if mod(queryMatrix(p,1), 200) == 100
        continue
    end
    rmseVector(p) = queryMatrix(p, 4);
end

rmseVector = nonzeros(rmseVector);

% Compute squared errors
squaredErrors = (rmseVector - strainsRaw).^2;

% Calculate RMSE & nRMSE using difference between predicted and actual
% strains
rmse = sqrt(mean(squaredErrors));
nrmse = rmse/(max(strainsRaw)-min(strainsRaw));
disp(['Root Mean Square Error (RMSE): ', num2str(rmse)]);
disp(['Normal Root Mean Square Error (nRMSE): ', num2str(nrmse)]);

% find CPU time expired during construction, training and execution of model
tEnd = cputime - tStart
```

## Appendix B – Cubic Splines Surrogate Model: MATLAB Script

```
% code was created by Thomas Rundle, Flinders University
clc;
clear all;
load HalfFemurParanalysis.mat
tStart = cputime;
%% specifying data from FEM simulation, sorting it into individual predictor inputs
hipRaw = -HalfFemurParanalysis.HipCF.';            % Input 1
adductRaw = HalfFemurParanalysis.AdductorMF.';   % Input 2
gluteRaw = HalfFemurParanalysis.GluteMF.';         % Input 3
strainsRaw = HalfFemurParanalysis.StrainMax.';     % Output

%% reshaping the data so that it can be interpolated using 'interpn' and 'spline'
% this requires the data to not be interpreted as one whole dataseries, but
% rather a 5x5x5 block combination of all possible values.
hip = [1000:200:1800];
adduct = [400:100:800];
glute = [700:100:1100];
strainValues = HalfFemurParanalysis.StrainMax;

gridsize = [5,5,5]; % reshaping the data
strains = reshape(strainValues, gridsize);

[X1, X2, X3] = ndgrid(hip, adduct, glute);

%% for predicting single strain responses, useful for comparison of known solutions to
other methods
hipTest = 1000;
adductTest = 800;
gluteTest = 900;
testValue = interpn(X1, X2, X3, strains, hipTest, adductTest, gluteTest, 'spline');
disp('Predicted strain')
disp(testValue)

%% storing the predicted strains so they can be plotted
vecLocation = 1;
hipCount = 1000;
adductCount = 400;
gluteCount = 700;
predictedStrainVector = zeros(1,length(strainsRaw));
queryMatrix = zeros(length(strainsRaw), 4);

% loop to interpolate more data points than the ones given, and assigning to
'queryMatrix'
% creates more points to plot the surface from
for i = 0:8
    for j = 0:8
        for k = 0:8
            gluteValue = gluteCount+k*50;
            adductValue = adductCount + j*50;
            hipValue = hipCount + i*100;
            k = k+1;
            strainValue = interpn(X1, X2, X3, strains, hipValue, adductValue,
gluteValue, 'spline');
            % interpn is used as the interpolation function, specified to
            % use splines
            predictedStrainVector(vecLocation) = strainValue;
            queryMatrix(vecLocation, 1) = hipValue;
            queryMatrix(vecLocation, 2) = adductValue;
            queryMatrix(vecLocation, 3) = gluteValue;
```

```
            queryMatrix(vecLocation, 4) = strainValue;
            vecLocation = vecLocation + 1;
        end
        j = j+1;
    end
    i = i+1;
end

%% plotting a singular surface for validation
% in this instance, we will isolating the hip and adductor, holding the
% glute force constant

surfVector = zeros(length(strainsRaw)/5,4);
surfPredictedVec = zeros(length(queryMatrix)/9,3);
surfVecLocation = 1;
surfPredictedVecLocation = 1;

% looping through points to find suitable locations where glute is constant
for l = 1:5:length(strainsRaw)
    surfVector(surfVecLocation, 1) = hipRaw(l);
    surfVector(surfVecLocation, 2) = adductRaw(l);
    surfVector(surfVecLocation, 3) = strainsRaw(l);
    surfVector(surfVecLocation, 4) = predictedStrainVector(l);
    surfVecLocation = surfVecLocation + 1;
end

% setting up a surface for this plot
X_raw = surfVector(:, 1);
Y_raw = surfVector(:, 2);
Z_raw = surfVector(:, 3);

% reshape the data for plotting as a surface
num_X = numel(unique(X_raw));
num_Y = numel(unique(Y_raw));
X_raw = reshape(X_raw, num_Y, num_X);
Y_raw = reshape(Y_raw, num_Y, num_X);
Z_raw = reshape(Z_raw, num_Y, num_X);

% doing the same for the much larger queryMatrix, which has far more values
for m = 1:9:length(queryMatrix)
    surfPredictedVec(surfPredictedVecLocation, 1) = queryMatrix(m, 1);
    surfPredictedVec(surfPredictedVecLocation, 2) = queryMatrix(m, 2);
    surfPredictedVec(surfPredictedVecLocation, 3) = queryMatrix(m, 4);
    surfPredictedVecLocation = surfPredictedVecLocation + 1;
end

% creating a surface for queryMatrix
X_interp = surfPredictedVec(:, 1);
Y_interp = surfPredictedVec(:, 2);
Z_interp = surfPredictedVec(:, 3);

% reshape the data to plot as surface
num_X_interp = numel(unique(X_interp));
num_Y_interp = numel(unique(Y_interp));
X_interp = reshape(X_interp, num_Y_interp, num_X_interp);
Y_interp = reshape(Y_interp, num_Y_interp, num_X_interp);
Z_interp = reshape(Z_interp, num_Y_interp, num_X_interp);


% plotting the two datasets, surfVector and surfPredictedVector, which
% represents the raw data plotted against the interpolated data, and also
% plotting the two surfaces
```

```
figure
plot3(surfVector(:,1), surfVector(:,2), surfVector(:,3), 'ro')
hold on
plot3(surfPredictedVec(:,1), surfPredictedVec(:,2), surfPredictedVec(:,3), 'kx')
hold on
surf(X_raw, Y_raw, Z_raw, 'FaceColor', 'r', 'EdgeColor', 'none');
alpha(0.2);
hold on
surf(X_interp, Y_interp, Z_interp, 'FaceColor', 'k', 'EdgeColor', 'none');
alpha(0.2);
xlabel('Hip Force (N)');
ylabel('Adductor Force (N)');
zlabel('Maximal Strain');
title({'Cubic Splines',...
    'Hip & Adductor Force, Predicted vs Actual Strains'})
legend('Actual,','Predicted', 'Location', 'southeast')
ax = gca;
ax.TitleFontSizeMultiplier = 1.5;

%% Calculating RMSE and nRMSE
rmseVector = zeros(length(queryMatrix), 1);

for p = 1:length(queryMatrix);
    if mod(queryMatrix(p,3), 100) == 50
        continue
    end
    if mod(queryMatrix(p,2), 100) == 50
        continue
    end
    if mod(queryMatrix(p,1), 200) == 100
        continue
    end
    rmseVector(p) = queryMatrix(p, 4);
end
rmseVector = nonzeros(rmseVector);

% Compute squared errors
squaredErrors = (rmseVector - strainsRaw').^2;

% Calculate RMSE & nRMSE
rmse = sqrt(mean(squaredErrors));
nrmse = rmse/(max(strainsRaw)-min(strainsRaw));
disp(['Root Mean Square Error (RMSE): ', num2str(rmse)]);
disp(['Normal Root Mean Square Error (nRMSE): ', num2str(nrmse)]);

% find CPU time expired during construction, training and execution of model
tEnd = cputime – tStart
```

## Appendix C – Kriging Surrogate Model: MATLAB Script

```
% code was created by Thomas Rundle, Flinders University
clc;
clear all;
load HalfFemurParanalysis.mat
tStart = cputime;
%% specifying data from FEM simulation, sorting it into individual predictor inputs
hipRaw = -HalfFemurParanalysis.HipCF.';            % Input 1
adductRaw = HalfFemurParanalysis.AdductorMF.';   % Input 2
gluteRaw = HalfFemurParanalysis.GluteMF.';         % Input 3
strainsRaw = HalfFemurParanalysis.StrainMax.';     % Output

% Using the provided data
hip = [1000:200:1800];
adduct = [400:100:800];
glute = [700:100:1100];

gridsize = [5,5,5]; % reshaping the data
strains = reshape(strainsRaw, gridsize);

[X1, X2, X3] = ndgrid(hip, adduct, glute);

% Create a matrix for input variables and output variables
X = [X1(:), X2(:), X3(:)];
Y = strains(:);

% Assuming you have new locations stored in variables X_new1, X_new2, X_new3
X_new = [1000, 800, 900];

% Preallocate matrices for interpolated output and kriging variance
Y_pred_test = zeros(size(X_new, 1), size(Y, 2));
sigma = zeros(size(X_new, 1), size(Y, 2));

% Perform kriging interpolation for each output variable
for t = 1:size(Y, 2)
    % Create the kriging model for the current output variable
    krigingModel = fitrgp(X, Y(:, t), 'FitMethod', 'none', 'PredictMethod', 'exact',
'KernelFunction', 'ardsquaredexponential');

    % Perform kriging interpolation at new locations
    [Y_pred_test(:, t), sigma(:, t)] = predict(krigingModel, X_new);
end

% displaying results
text = sprintf('Hip joint force = %dN, adductor force = %dN, glute force = %dN',
X_new(1), X_new(2), X_new(3));
disp(text);

% Y_pred contains the interpolated output values at the new locations for each variable
disp('Interpolated output values:');
disp(Y_pred_test);

% sigma contains the kriging variance at the new locations for each variable
disp('Kriging variance:');
disp(sigma);

queryMatrix = zeros(length(strainsRaw), 5);
vecLocation = 1;
hipCount = 1000;
adductCount = 400;
```

```
gluteCount = 700;

% looping through data to create plottable data points. Points will be
% joined using a surface. 125 points used to train --> 729 points
% interpolated
for h = 0:8
    for j = 0:8
        for k = 0:8
            gluteValue = gluteCount+k*50;
            adductValue = adductCount + j*50;
            hipValue = hipCount + h*100;
            X_query = [hipValue, adductValue, gluteValue];

            % Perform kriging interpolation for each output variable
            for i = 1:size(Y, 2)
                % Create the kriging model for the current output variable
                krigingModel = fitrgp(X, Y(:, i), 'FitMethod', 'none', 'PredictMethod',
'exact', 'KernelFunction', 'ardsquaredexponential');

                % Perform kriging interpolation at new locations
                [Y_pred(:, i), sigma_interp(:, i)] = predict(krigingModel, X_query);
            end

            k = k+1;
            queryMatrix(vecLocation, 1) = hipValue;
            queryMatrix(vecLocation, 2) = adductValue;
            queryMatrix(vecLocation, 3) = gluteValue;
            queryMatrix(vecLocation, 4) = Y_pred;
            queryMatrix(vecLocation, 5) = sigma_interp;
            vecLocation = vecLocation + 1;
        end
        j = j+1;
    end
    h = i+1;
end


%% plotting a singular surface for validation
% in this instance, we will isolating the hip and adductor, holding the
% glute force constant

surfVector = zeros(length(strainsRaw)/5,4);
surfPredictedVec = zeros(length(queryMatrix)/9,3);
surfVecLocation = 1;
surfPredictedVecLocation = 1;

% looping through points to find suitable locations where glute is constant
for l = 1:5:length(strainsRaw)
    surfVector(surfVecLocation, 1) = hipRaw(l);
    surfVector(surfVecLocation, 2) = adductRaw(l);
    surfVector(surfVecLocation, 3) = strainsRaw(l);
    surfVector(surfVecLocation, 4) = queryMatrix(l,4);
    surfVecLocation = surfVecLocation + 1;
end

% setting up a surface for this plot
X_raw = surfVector(:, 1);
Y_raw = surfVector(:, 2);
Z_raw = surfVector(:, 3);

% reshape the data for plotting as a surface
num_X = numel(unique(X_raw));
```

```
num_Y = numel(unique(Y_raw));
X_raw = reshape(X_raw, num_Y, num_X);
Y_raw = reshape(Y_raw, num_Y, num_X);
Z_raw = reshape(Z_raw, num_Y, num_X);

% doing the same for the much larger queryMatrix, which has far more values
for m = 1:9:length(queryMatrix)
    surfPredictedVec(surfPredictedVecLocation, 1) = queryMatrix(m, 1);
    surfPredictedVec(surfPredictedVecLocation, 2) = queryMatrix(m, 2);
    surfPredictedVec(surfPredictedVecLocation, 3) = queryMatrix(m, 4);
    surfPredictedVecLocation = surfPredictedVecLocation + 1;
end

% creating a surface for queryMatrix
X_interp = surfPredictedVec(:, 1);
Y_interp = surfPredictedVec(:, 2);
Z_interp = surfPredictedVec(:, 3);

% reshape the data to plot as surface
num_X_interp = numel(unique(X_interp));
num_Y_interp = numel(unique(Y_interp));
X_interp = reshape(X_interp, num_Y_interp, num_X_interp);
Y_interp = reshape(Y_interp, num_Y_interp, num_X_interp);
Z_interp = reshape(Z_interp, num_Y_interp, num_X_interp);


% plotting the two datasets, surfVector and surfPredictedVector, which
% represents the raw data plotted against the interpolated data, and also
% plotting the two surfaces
figure
plot3(surfVector(:,1), surfVector(:,2), surfVector(:,3), 'ro')
hold on
plot3(surfPredictedVec(:,1), surfPredictedVec(:,2), surfPredictedVec(:,3), 'kx')
hold on
surf(X_raw, Y_raw, Z_raw, 'FaceColor', 'r', 'EdgeColor', 'none');
alpha(0.2);
hold on
surf(X_interp, Y_interp, Z_interp, 'FaceColor', 'k', 'EdgeColor', 'none');
alpha(0.2);
xlabel('Hip Force (N)');
ylabel('Adductor Force (N)');
zlabel('Maximal Strain');
title({'Kriging (Gaussian Process Regression)',...
    'Hip & Adductor Force, Predicted vs Actual Strains'})
legend('Actual,','Predicted', 'Location', 'southeast')
ax = gca;
ax.TitleFontSizeMultiplier = 1.5;

%% Calculating RMSE & nRMSE

rmseVector = zeros(length(queryMatrix), 1);

for p = 1:length(queryMatrix);
    if mod(queryMatrix(p,3), 100) == 50
        continue
    end
    if mod(queryMatrix(p,2), 100) == 50
        continue
    end
    if mod(queryMatrix(p,1), 200) == 100
        continue
    end
```

```
    rmseVector(p) = queryMatrix(p, 4);
end

rmseVector = nonzeros(rmseVector);

% Compute squared errors
squaredErrors = (rmseVector - strainsRaw').^2;

% Calculate RMSE & nRMSE
rmse = sqrt(mean(squaredErrors));
nrmse = rmse/(max(strainsRaw)-min(strainsRaw));
disp(['Root Mean Square Error (RMSE): ', num2str(rmse)]);
disp(['Normal Root Mean Square Error (nRMSE): ', num2str(nrmse)]);

% find CPU time expired during construction, training and execution of model
tEnd = cputime - tStart
```

## Appendix D – Superposition Principle Method Surrogate Model (Linear Model): MATLAB Script

```
% code was created by Thomas Rundle, Flinders University
clc;
clear all;
load ComponentsSPM.mat
load HalfFemurTrial1.mat
tStart = cputime;

%% initialise data from obtained FE dataset (excel)
nodes = ComponentsSPM.NodeNumber;
hipXY = ComponentsSPM.Hip_XY_Shear;
hipYZ = ComponentsSPM.Hip_YZ_Shear;
hipXZ = ComponentsSPM.Hip_XZ_Shear;
hipNormX = ComponentsSPM.Hip_X_Normal;
hipNormY = ComponentsSPM.Hip_Y_Normal;
hipNormZ = ComponentsSPM.Hip_Z_Normal;
gluteXY = ComponentsSPM.Glute_XY_Shear;
gluteYZ = ComponentsSPM.Glute_YZ_Shear;
gluteXZ = ComponentsSPM.Glute_XZ_Shear;
gluteNormX = ComponentsSPM.Glute_X_Normal;
gluteNormY = ComponentsSPM.Glute_Y_Normal;
gluteNormZ = ComponentsSPM.Glute_Z_Normal;
adductXY = ComponentsSPM.Adduct_XY_shear;
adductYZ = ComponentsSPM.Adduct_YZ_shear;
adductXZ = ComponentsSPM.Adduct_XZ_shear;
adductNormX = ComponentsSPM.Adduct_X_Normal;
adductNormY = ComponentsSPM.Adduct_Y_Normal;
adductNormZ = ComponentsSPM.Adduct_Z_Normal;

% identify number of nodes
numNodes = length(nodes);

% set multiplier used for superposition of each tensor
hipMulti = 1;
gluteMulti = 0.4;
adductMulti = 0.7;

%% creating strain tensors for each node in response to HIP
hipTensors = zeros(numNodes, 7);

for i = 1:numNodes
   hipTensors(i, 1) = nodes(i);
   hipTensors(i, 2) = hipXY(i);
   hipTensors(i, 3) = hipYZ(i);
   hipTensors(i, 4) = hipXZ(i);
   hipTensors(i, 5) = hipNormX(i);
   hipTensors(i, 6) = hipNormY(i);
   hipTensors(i, 7) = hipNormZ(i);
end

%% creating strain tensors for each node in response to GLUTE
gluteTensors = zeros(numNodes, 7);

for i = 1:numNodes
   gluteTensors(i, 1) = nodes(i);
   gluteTensors(i, 2) = gluteXY(i);
   gluteTensors(i, 3) = gluteYZ(i);
   gluteTensors(i, 4) = gluteXZ(i);
   gluteTensors(i, 5) = gluteNormX(i);
```

```
    gluteTensors(i, 6) = gluteNormY(i);
    gluteTensors(i, 7) = gluteNormZ(i);
end

%% creating strain tensors for each node in response to ADDUCTOR
adductTensors = zeros(numNodes, 7);

for i = 1:numNodes
    adductTensors(i, 1) = nodes(i);
    adductTensors(i, 2) = adductXY(i);
    adductTensors(i, 3) = adductYZ(i);
    adductTensors(i, 4) = adductXZ(i);
    adductTensors(i, 5) = adductNormX(i);
    adductTensors(i, 6) = adductNormY(i);
    adductTensors(i, 7) = adductNormZ(i);
end

%% addition of tensors to find final tensors via superposition
finalTensors = hipMulti*hipTensors + gluteMulti*gluteTensors +
adductMulti*adductTensors;
finalTensors(:,1) = nodes;

%% creating strain tensors for each node in response to TRIAL 1, where loadings were
determined:
% Hip = 1000N
% Glute = 400N
% Adductor = 700N

trialTensors = zeros(numNodes, 7);
for i = 1:numNodes
    trialTensors(i, 1) = table2array(HalfFemurTrial1(i,1));
    trialTensors(i, 2) = table2array(HalfFemurTrial1(i,2));
    trialTensors(i, 3) = table2array(HalfFemurTrial1(i,3));
    trialTensors(i, 4) = table2array(HalfFemurTrial1(i,4));
    trialTensors(i, 5) = table2array(HalfFemurTrial1(i,5));
    trialTensors(i, 6) = table2array(HalfFemurTrial1(i,6));
    trialTensors(i, 7) = table2array(HalfFemurTrial1(i,7));
end

%% finding max and min normal strains (not required later)
maxNormStrainTrial1 = max(trialTensors(:,5:7));
trueMaxTrial1 = max(maxNormStrainTrial1)

maxNormStrain = max(finalTensors(:,5:7));
trueMaxFinal = max(maxNormStrain)

%% plotting the x, y, z normal strain tensor components
% surface is not used like other methods, since only 2D dataset is
% available
x1 = trialTensors(:,5);
y1 = finalTensors(:,5);
x2 = trialTensors(:,6);
y2 = finalTensors(:,6);
x3 = trialTensors(:,7);
y3 = finalTensors(:,7);

figure
plot(x1, y1, 'r*')
hold on
plot(x2, y2, 'bo')
hold on
plot(x3, y3, 'kx')
```

```
xlabel('Actual Strain');
ylabel('Predicted Strain');

title('Normal Strain Components',...
    'Actual vs Predicted Using SPM')
legend('x','y','z', 'south')
ax = gca;
ax.TitleFontSizeMultiplier = 1.5;

% find CPU time expired during construction, training and execution of model
tEnd = cputime - tStart
```

## Appendix E - Superposition Principle Method Surrogate Model (Non-linear Model): MATLAB Script

```
% code was created by Thomas Rundle, Flinders University
%% initialise all data. Load relevent files
clc;
clear all;
tStart = cputime; % starts counting for CPU time

%% loading X normal strain components (excel)
load addX11.mat
load addY11.mat
load addZ11.mat
load hipX11.mat
load hipY11.mat
load hipZ11.mat
load kneeX11.mat
load kneeY11.mat
load kneeZ11.mat
load T1_11.mat
load T2_11.mat
load T3_11.mat


%% loading X normal strain components (excel)
load addX22.mat
load addY22.mat
load addZ22.mat
load hipX22.mat
load hipY22.mat
load hipZ22.mat
load kneeX22.mat
load kneeY22.mat
load kneeZ22.mat
load T1_22.mat
load T2_22.mat
load T3_22.mat


%% loading X normal strain components (excel)
load addX33.mat
load addY33.mat
load addZ33.mat
load hipX33.mat
load hipY33.mat
load hipZ33.mat
load kneeX33.mat
load kneeY33.mat
load kneeZ33.mat
load T1_33.mat
load T2_33.mat
load T3_33.mat

%% setting the multipliers for the strain tensor addition
hip_x_multi = -0.5;
hip_y_multi = 0.25;
hip_z_multi = -0.3;

knee_x_multi = -0.12;
knee_y_multi = -0.4;
knee_z_multi = 0.02;
```

```
adduct_x_multi = 0.55;
adduct_y_multi = 0.21;
adduct_z_multi = 0.01;

nodes = length(table2array(XNORMALspmaddx2));

%% creating the tensors for the isolated hip responses, followed by knee and adductor
% remember, each force has three components (x, y, z)
% each component has three normal strains (x, y, z)
% hence there are 9 measured normal strain values per force, + 1 column to
% count nodes
hip_tensors = zeros(nodes/2, 10);

for i = 1:(nodes/2)
    hip_tensors(i, 1) = i;
    hip_tensors(i, 2) = table2array(XNORMALspmhipx2(1, 2*i));
    hip_tensors(i, 3) = table2array(XNORMALspmhipy2(1, 2*i));
    hip_tensors(i, 4) = table2array(XNORMALspmhipz2(1, 2*i));
    hip_tensors(i, 5) = table2array(YNORMALspmhipx2(1, 2*i));
    hip_tensors(i, 6) = table2array(YNORMALspmhipy2(1, 2*i));
    hip_tensors(i, 7) = table2array(YNORMALspmhipz2(1, 2*i));
    hip_tensors(i, 8) = table2array(ZNORMALspmhipx2(1, 2*i));
    hip_tensors(i, 9) = table2array(ZNORMALspmhipy2(1, 2*i));
    hip_tensors(i, 10) = table2array(ZNORMALspmhipz2(1, 2*i));
end

knee_tensors = zeros(nodes/2, 5);

for i = 1:(nodes/2)
    knee_tensors(i, 1) = i;
    knee_tensors(i, 2) = table2array(XNORMALspmkneex2(1, 2*i));
    knee_tensors(i, 3) = table2array(XNORMALspmkneey2(1, 2*i));
    knee_tensors(i, 4) = table2array(XNORMALspmkneez2(1, 2*i));
    knee_tensors(i, 5) = table2array(YNORMALspmkneex2(1, 2*i));
    knee_tensors(i, 6) = table2array(YNORMALspmkneey2(1, 2*i));
    knee_tensors(i, 7) = table2array(YNORMALspmkneez2(1, 2*i));
    knee_tensors(i, 8) = table2array(ZNORMALspmkneex2(1, 2*i));
    knee_tensors(i, 9) = table2array(ZNORMALspmkneey2(1, 2*i));
    knee_tensors(i, 10) = table2array(ZNORMALspmkneez2(1, 2*i));

end

adduct_tensors = zeros(nodes/2, 4);

for i = 1:(nodes/2)
    adduct_tensors(i, 1) = i;
    adduct_tensors(i, 2) = table2array(XNORMALspmaddx2(1, 2*i));
    adduct_tensors(i, 3) = table2array(XNORMALspmaddy2(1, 2*i));
    adduct_tensors(i, 4) = table2array(XNORMALspmaddz2(1, 2*i));
    adduct_tensors(i, 5) = table2array(YNORMALspmaddx2(1, 2*i));
    adduct_tensors(i, 6) = table2array(YNORMALspmaddy2(1, 2*i));
    adduct_tensors(i, 7) = table2array(YNORMALspmaddz2(1, 2*i));
    adduct_tensors(i, 8) = table2array(ZNORMALspmaddx2(1, 2*i));
    adduct_tensors(i, 9) = table2array(ZNORMALspmaddy2(1, 2*i));
    adduct_tensors(i, 10) = table2array(ZNORMALspmaddz2(1, 2*i));
end


%% creating the tensors for the responses to combination loadings
% this is the assembly of responses from known combination loadings found
% in Abaqus software
```

```
trial_tensors = zeros(nodes/2, 4);

for k = 1:(nodes/2)
    trial_tensors(k, 1) = k;
    trial_tensors(k, 2) = table2array(XNORMALT2(1, 2*k));
    trial_tensors(k, 3) = table2array(YNORMALT2(1, 2*k));
    trial_tensors(k, 4) = table2array(ZNORMALT2(1, 2*k));
end


%% creating the predicted tensors, by applying superposition. This will be compared
with the trial tensors
predicted_tensors = zeros(nodes/2, 4);

predicted_tensors(:,1) = hip_tensors(:,1);

predicted_tensors(:,2) = hip_x_multi*hip_tensors(:,2) + hip_y_multi*hip_tensors(:,3) +
hip_z_multi*hip_tensors(:,4) +...
    knee_x_multi*knee_tensors(:,2) + knee_y_multi*knee_tensors(:,3) +
knee_z_multi*knee_tensors(:,4) + ...
    adduct_x_multi*adduct_tensors(:,2) + adduct_y_multi*adduct_tensors(:,3) +
adduct_z_multi*adduct_tensors(:,4);

predicted_tensors(:,3) = hip_x_multi*hip_tensors(:,5) + hip_y_multi*hip_tensors(:,6) +
hip_z_multi*hip_tensors(:,7) +...
    knee_x_multi*knee_tensors(:,5) + knee_y_multi*knee_tensors(:,6) +
knee_z_multi*knee_tensors(:,7) + ...
    adduct_x_multi*adduct_tensors(:,5) + adduct_y_multi*adduct_tensors(:,6) +
adduct_z_multi*adduct_tensors(:,7);

predicted_tensors(:,4) = hip_x_multi*hip_tensors(:,8) + hip_y_multi*hip_tensors(:,9) +
hip_z_multi*hip_tensors(:,10) +...
    knee_x_multi*knee_tensors(:,8) + knee_y_multi*knee_tensors(:,9) +
knee_z_multi*knee_tensors(:,10) + ...
    adduct_x_multi*adduct_tensors(:,8) + adduct_y_multi*adduct_tensors(:,9) +
adduct_z_multi*adduct_tensors(:,10);

%% plotting singular normal strain components
x1 = trial_tensors(:,2);
y1 = predicted_tensors(:,2);
x2 = trial_tensors(:,3);
y2 = predicted_tensors(:,3);
x3 = trial_tensors(:,4);
y3 = predicted_tensors(:,4);

% used to find confidence interval (95%)
p1 = polyfit(x1, y1, 1);
f1 = polyval(p1, x1);
p2 = polyfit(x2, y2, 1);
f2 = polyval(p2, x2);
p3 = polyfit(x3, y3, 1);
f3 = polyval(p3, x3);

[w1, S1] = polyfit(x1, y1, 1);
[y_fit1, delta1] = polyval(w1, x1, S1);
[w2, S2] = polyfit(x2, y2, 1);
[y_fit2, delta2] = polyval(w2, x2, S2);
[w3, S3] = polyfit(x3, y3, 1);
[y_fit3, delta3] = polyval(w3, x3, S3);

% specifically made for the z component at the moment, change variables as
% needed to plot x and y components
```

```
figure
plot(x3, y3, 'b.', 'LineWidth', 0.1)
hold on
plot(x3, y_fit3, 'r-', 'LineWidth', 1)
xlim([-15*10^(-5) 10*10^(-5)])
ylim([-15*10^(-5) 10*10^(-5)])
hold on
plot(x3, y_fit3+2*delta3, 'm--',x3, y_fit3-2*delta3, 'm--')
xlabel('Actual Strain', 'FontSize', 16);
ylabel('Predicted Strain', 'FontSize',16);
title('Normal Strain - Z Component',...
    'Actual vs Predicted Using SPM')
legend('Strain','Linear Fit','95% Prediction Interval', 'FontSize',12)
ax = gca;
ax.TitleFontSizeMultiplier = 1.5;


%% plots all x y z on one axis
figure
plot(x1, y1, 'b.')
hold on
plot(x2, y2, 'k.')
hold on
plot(x3, y3, 'r.')
xlim([-15*10^(-5) 10*10^(-5)])
ylim([-15*10^(-5) 10*10^(-5)])
xlabel('Actual Strain', 'FontSize', 12);
ylabel('Predicted Strain', 'FontSize', 12);
title('Normal Strain Components - Non-Linear Model',...
    'Actual vs Predicted Using SPM')
legend('x','y','z', 'Fontsize', 12)
alpha(0.5)
ax = gca;
ax.TitleFontSizeMultiplier = 1.5;

%% validating, finding nRMSE and CPU time
% Compute squared errors
XsquaredErrors = (trial_tensors(:,2) - predicted_tensors(:,2)).^2;

Xrmse = sqrt(mean(XsquaredErrors));
Xnrmse = Xrmse/(max(trial_tensors(:,2)-min(trial_tensors(:,2))));
%disp(['X Root Mean Square Error (RMSE): ', num2str(Xrmse)]);
disp(['X Normal Root Mean Square Error (nRMSE): ', num2str(Xnrmse)]);

YsquaredErrors = (trial_tensors(:,3) - predicted_tensors(:,3)).^2;

Yrmse = sqrt(mean(YsquaredErrors));
Ynrmse = Yrmse/(max(trial_tensors(:,3)-min(trial_tensors(:,3))));
%disp(['Y Root Mean Square Error (RMSE): ', num2str(Yrmse)]);
disp(['Y Normal Root Mean Square Error (nRMSE): ', num2str(Ynrmse)]);

ZsquaredErrors = (trial_tensors(:,4) - predicted_tensors(:,4)).^2;

Zrmse = sqrt(mean(ZsquaredErrors));
Znrmse = Zrmse/(max(trial_tensors(:,4)-min(trial_tensors(:,4))));
%disp(['Z Root Mean Square Error (RMSE): ', num2str(Zrmse)]);
disp(['Z Normal Root Mean Square Error (nRMSE): ', num2str(Znrmse)]);

% calculates time required to construct, train, and execute surrogate model
% (CPU time)
tEnd = cputime - tStart
```