Post doc report

Michael Dunne supervised by Prof. Kyle Wedgwood, Prof. Peter Challenor

April 2025

1 Principal component analysis

1.1 Introduction and singular value decomposition

Principal component analysis (PCA) derives dominant patterns of variability from a random vector in a statistical field (in this case p-dimensional Euclidean space) (Joliffe & Morgan 1992, Storch & Zwiers 1999) and forms new orthogonal axes to explain as much of the variability in the data as possible. These new orthogonal axes are called principal components. This process is done by taking an $N \times p$ (N is the number of data points, p is the number of variables) sized dataset M where each column has been centred (to mean 0) and scaled (to variance 1) and then determining the direction of greatest variance, which is denoted V_1; the first singular vector. Then V_2 is determined, this finds the direction of greatest variance under the condition $V_2 \perp V_1$. Then V_3 finds the greatest direction such that $V_3 \perp (V_2 \perp V_1)$. This continues for all $i = 1 \dots k$ where $k \leq \min(n, p)$ is the rank of the dataset M.

One can find the proportion of variance explained by each singular vector V_i by calculating

$$\Lambda_i = \frac{\lambda_i^2}{\sum_{j=1}^{k} \lambda^2}, \tag{1}$$

where λ_i^2 is the variance of the data along the axis V_i.

This process of selecting all V_is to find the directions of greatest variance of M is equivalent to determining the singular value decomposition (SVD) of M:

$$M = \underset{n\times k}{U}\ \underset{k\times k}{\Sigma}\ \underset{k\times p}{V^T}, \tag{2}$$

where M is an $N \times p$ matrix centred at 0 with each column scaled to have variance 1 and where U and V are left and right singular (and orthogonal) vectors for M respectively. This orthogonality means

$$U^T U = I_k, \tag{3}$$
$$V^T V = I_k, \tag{4}$$

where V is a $p \times k$ matrix containing all the singular vectors V_i for $i = 1 \dots k$ and Σ is the $k \times k$ diagonal matrix where the square of each singular value λ_i on the diagonal describes the proportion of variance explained by singular vector V_i in line with eq.(1). This process is illustrated in figure 1.

The SVD of M can be determined by first calculating Σ and V and rearranging eq.(2) to obtain U. By calculating $M^T M$:

$$M^T M = V\Sigma^T U^T U \Sigma V^T = V\Sigma^T I_k \Sigma V^T = V\Sigma^2 V^T, \tag{5}$$

(by applying condition (3)). Then applying eigen-decomposition to eq.(5), gives V and Σ^2 (and hence Σ). This can also be done by computing MM^T (where $MM^T = U\Sigma^2 U^T$) however $M^T M$ is $p \times p$ and MM^T is $N \times N$ so the latter leads to much larger calculations for the eigen-decomposition to determine U as $N >> p$.

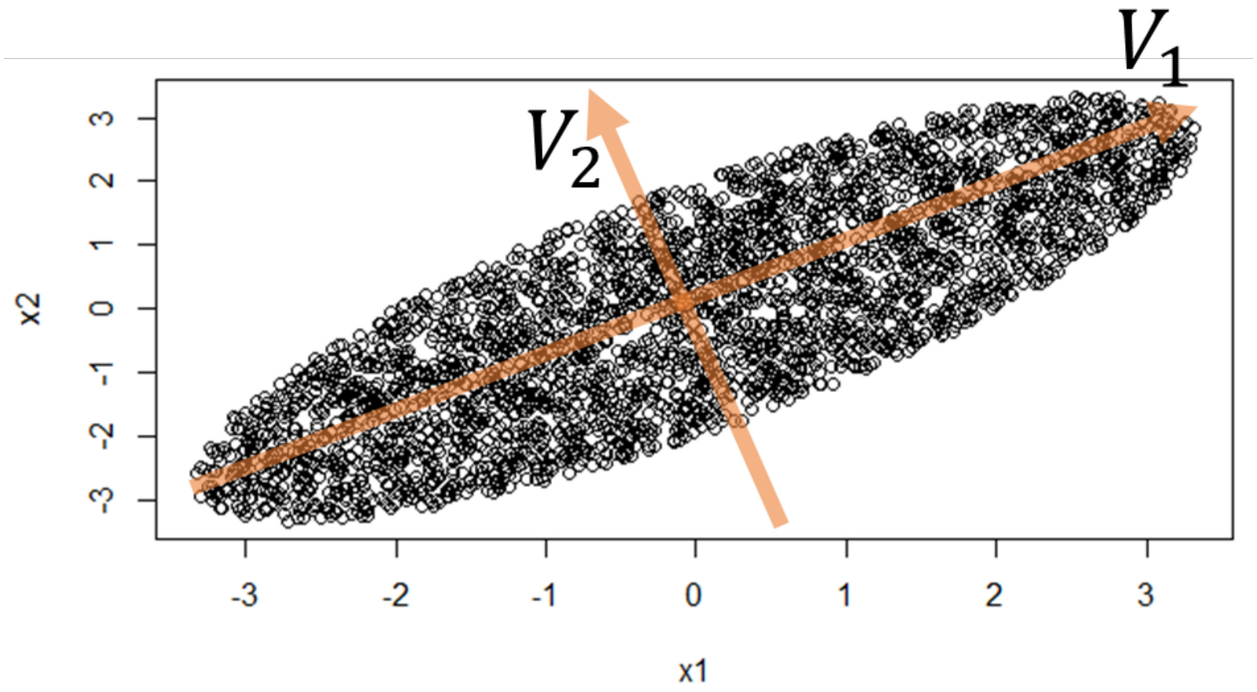


Figure 1: A dataset M (denoted by black points) where each variable has been centred (around 0) and scaled (to variance 1). The first principal component V_1 is the direction which explains direction of greatest variance, the second (V_2) represents the only other available direction which is orthogonal to V_1.

The co-ordinates/weights in this new basis V are contained within $U\Sigma$ which is obtained by rearranging eq.(2). After selecting the first $k'(< k)$ singular vectors using eq.(1), the weights are reduced to $(U\Sigma)_{1:k'}$ (the first k' columns of $U\Sigma$). The other singular vectors are then truncated.

A vector in the original p-dimensional space can be mapped onto the rotated basis using the transformation

$$w(\mathbf{x}) = \mathbf{x}^T V$$

with the inverse being

$$w^{-1}(\mathbf{x}') = \mathbf{x}'\ ^T V^T.$$

The methodologies from this section are utilised in the *R* function *prcomp* from the package *stats* (R Core Team 2013).

2 Uncertainty Quantification & Univariate Emulation

In the event that a simulator function f is too expensive to run, it is useful to instead represent it with a function that is computationally cheaper, namely an emulator. This is important as some analyses can take up to tens or hundreds of thousands of model runs (Tripathy et al. 2016), which if done directly on an expensive model is completely infeasible. Whilst f is not a random function, it can only be run at a small number of locations meaning that away from those locations the evaluation is uncertain. In a Bayesian context, that's a random function meaning that this uncertainty can be represented using a probability distribution. Mathematically, this emulator $\hat{f}$ is represented as a linear set of basis functions and a zero-mean random function

$$\hat{f}(\mathbf{x}) = \mathbf{h}^T(\mathbf{x})\beta + \epsilon(\mathbf{x}), \tag{6}$$

where $\mathbf{h}(\mathbf{x})$ are a set of m prior basis/trend functions, $\mathbf{h}$ are a set of m coefficients associated with those trend functions and $\epsilon(\mathbf{x})$ is a residual function between the data and the basis functions.

2.1 Gaussian process regression

A Gaussian process (GP) emulator is one option for ϵ in eq.(6). A Gaussian process is an infinite collection of random variables where any subset of these variables forms a multivariate normal distribution (Currin et al. 1991). Through the use of a Gaussian process, one can use Bayesian inference to make predictions on outputs (Currin et al. 1991, O'Hagan 2006, Kennedy & O'Hagan 2001, Oakley & O'Hagan 2002). The GP can be defined via a mean and covariance function which can be incorporated into the form of the emulator in eq.(6) through

$$\epsilon(\mathbf{x}) \sim GP(\mathbf{0}, \text{cov}(f(\mathbf{x}), f(\mathbf{x}'))).$$

This means the emulator $\hat{f}$ in eq.(6) has prior mean $\mathbf{h}(\mathbf{x})^T\beta$ where the basis functions $\mathbf{h}(.)$ can be chosen to be just a constant (i.e. $\mathbf{h}(\mathbf{x}) = (1)$), linear ($\mathbf{h}(\mathbf{x}) = (1, \mathbf{x}^T)^T$), quadratic, cubic, or any functions of input variables, and where β are the regression coefficients. The prior covariance between two outputs $f(\mathbf{x})$ and $f(\mathbf{x}')$ is given by

$$cov(f(\mathbf{x}), f(\mathbf{x}')) = \sigma^2 c(\mathbf{x}, \mathbf{x}') \tag{7}$$

where the form of the correlation function $c(.,.)$ is chosen such that $c(\mathbf{x}, \mathbf{x}) = 1$, $c(\mathbf{x}, \mathbf{x}')$ is close to 1 if the distance between $\mathbf{x}$ and $\mathbf{x}'$ is small and low if the distance is high.

The model for $\hat{f}(.)$ requires the estimation of parameters β and σ^2 where a prior can be specified over these two variables. In many applications, a weak prior is used which takes the form

$$p(\beta, \sigma^2) \propto \sigma^{-2}. \tag{8}$$

The emulator is trained using model runs with the output denoted

$$\mathbf{y_D} = (f(\mathbf{x}_1), \dots, f(\mathbf{x}_n)) = f(\mathbf{x_D}).$$

An assumption can be made using eq.(8) to model the output $\mathbf{y_D}$ as

$$\mathbf{y_D}|\beta, \sigma^2 \sim N(H\beta, \sigma^2 A),$$

where $H = (\mathbf{h}(\mathbf{x}_1), \dots, \mathbf{h}(\mathbf{x}_n))$ and

$$A = \begin{pmatrix} 1 & c(\mathbf{x}_1, \mathbf{x}_2) & \dots & c(\mathbf{x}_1, \mathbf{x}_n) \\ c(\mathbf{x}_2, \mathbf{x}_1) & 1 & & \vdots \\ \vdots & & \ddots & \vdots \\ c(\mathbf{x}_n, \mathbf{x}_1) & \dots & \dots & 1 \end{pmatrix}. \tag{9}$$

This allows for the emulator $\hat{f}$ to take the following form:

$$\hat{f}(.)|\beta, \sigma^2, \mathbf{y_D} \sim GP(m^*(.), cov^*(.,.)),$$

where

$$\begin{aligned} m^*(\mathbf{x}) &= \mathbf{h}^T(\mathbf{x})\beta + \mathbf{t}(\mathbf{x})^{\mathbf{T}} A^{-1}(\mathbf{y_D} - H\beta), && (10) \\ cov^*(.,.) &= \sigma^2 c^*(.,.), \\ c^*(\mathbf{x}, \mathbf{x}') &= c(\mathbf{x}, \mathbf{x}') - \mathbf{t}(\mathbf{x})A^{-1}\mathbf{t}(\mathbf{x}'), && (11) \\ H^T &= (\mathbf{h}(\mathbf{x}_1), \dots, \mathbf{h}(\mathbf{x}_n)), \\ \mathbf{t}(\mathbf{x})^T &= (c(\mathbf{x}, \mathbf{x}_1), \dots, c(\mathbf{x}, \mathbf{x}_n)) \text{ and} \\ \mathbf{y_D}^T &= (f(\mathbf{x}_1), \dots, f(\mathbf{x}_n)). \end{aligned}$$

The posterior mean in eq.(10) and posterior correlation in eq.(11) are updated from $\mathbf{h}$ and $c(\mathbf{x}, \mathbf{x}')$ respectively using the derivations outlined in Krzanowski (1988). Now what remains is to remove the conditioning on the unknown parameters β and σ^2. The result from integrating out β gives

$$\hat{f}(.)|\mathbf{y_D}, \sigma^2 \sim GP(m^{**}(.), cov^{**}(.,.))$$

where

$$
\begin{aligned}
& m^{**}(\mathbf{x}) = \mathbf{h^T}\hat{\beta} + \mathbf{t}(\mathbf{x})^T A^{-1}(\mathbf{y_D} - H\hat{\beta}), && (12)\\
& cov^{**}(.,.) = \sigma^2 c^{**}(.,.), \\
& c^{**}(\mathbf{x},\mathbf{x'}) = c(\mathbf{x},\mathbf{x'}) - \mathbf{t}(\mathbf{x})A^{-1}\mathbf{t}(\mathbf{x'}) && (13)\\
& + (\mathbf{h^T} - \mathbf{t}(\mathbf{x})^\mathbf{T} A^{-1}H)(H^T A^{-1}H)^{-1}(\mathbf{h}(\mathbf{x'})^\mathbf{T} - \mathbf{t}(\mathbf{x'})^\mathbf{T} A^{-1}H)^T, \\
& H^T = (\mathbf{h}(\mathbf{x_1}), \dots, \mathbf{h}(\mathbf{x_n})), \\
& \mathbf{t}(\mathbf{x})^T = (c(\mathbf{x},\mathbf{x_1}), \dots, c(\mathbf{x},\mathbf{x}_n)), \\
& \mathbf{y}_\mathbf{D}^T = (f(\mathbf{x}_1), \dots, f(\mathbf{x}_n)) \text{ and} \\
& \beta = (H^T A^{-1}H)^{-1}H^T A^{-1}\mathbf{y_D}. && (14)
\end{aligned}
$$

Finally, integrating out σ^2 leads to the property that

$$
\frac{f(\mathbf{x}) - m^{**}(\mathbf{x})}{\hat{\sigma}\sqrt{c^{**}(\mathbf{x},\mathbf{x'})}} \sim t_{n-m} \tag{15}
$$

where t_{n-m} is a student t random variable with $n-m$ degrees of freedom and

$$
\hat{\sigma}^2 = \frac{\mathbf{y}_\mathbf{D}^T A^{-1}\mathbf{y_D} - \hat{\beta}^T(H^T A^{-1}H)\hat{\beta}}{n-m-2}. \tag{16}
$$

2.1.1 Nugget

A nugget (or 'jitter') is an, often small (see Andrianakis & Challenor (2012), Baker et al. (2020)), noise term added to the covariance matrix A in order to improve the inversion (Andrianakis & Challenor 2012) whilst having a small effect on the model output $f(\mathbf{x})$ (Neal 1997, Andrianakis & Challenor 2012).

It is incorporated into the model by adding an extra term $\nu(.)$ to eq.(6):

$$
\hat{f}(\mathbf{x}) = \mathbf{h}^T(\mathbf{x})\hat{\beta} + \epsilon(\mathbf{x}) + \nu(\mathbf{x}), \tag{17}
$$

where $\nu(\mathbf{x})$ is a normally distributed random variable with mean 0 and variance τ^2 at any point $\mathbf{x}$. The covariance of $\nu(.)$ is given by

$$
cov(\nu(\mathbf{x}),\nu(\mathbf{x'})) = \begin{cases} \tau^2, & \mathbf{x} = \mathbf{x'}, \\ 0, & \textit{otherwise}. \end{cases}
$$

2.1.2 Correlation functions

The choice of covariance function (or kernel) is important as it encodes the assumptions made about the function being emulated (Rasmussen & Williams 2006). In section 2.1 the form of the covariance function was introduced via eq.(7). In this section the types of correlation functions (namely $c(.,.)$) are outlined. Correlation functions can be defined for each input variable x_i individually and combined such that

$$
c(\mathbf{x},\mathbf{x'}) = \prod_{i=1}^{p} c_i(x_i, x_i'), \tag{18}
$$

where x_i and x_i' are the ith components of $\mathbf{x}$ and $\mathbf{x'}$ respectively.

The kernels defined in this section are stationary kernels defined by $c_i(\mathbf{x},\mathbf{x'}) = g(\mathbf{x} - \mathbf{x'})$. Kernels of the form $c_i(\mathbf{x},\mathbf{x'}) = g(|\mathbf{x} - \mathbf{x'}|)$ are known as isotropic. A kernel that has the property $c_i(\mathbf{x},\mathbf{x'}) = c_i(\mathbf{x'},\mathbf{x})$ is called symmetric implying that any isotropic kernel is symmetric (Rasmussen & Williams 2006).

One group of kernels are the family of Matérn kernels (Matérn 1947) which are defined by

$$c_i(x_i, x_i') = \frac{2^{1-\eta}}{\Gamma(\eta)} \left(\frac{\sqrt{2\eta}|x_i - x_i'|}{\theta_i} \right)^{\eta} K_\eta \left(\frac{\sqrt{2\eta}|x_i - x_i'|}{\theta_i} \right) \tag{19}$$

where K_η is the modified Bessel function. The Matérn has $p + 1$ parameters: $\eta > 0$ and the lengthscale for each variable $\theta_i > 0$ for $i = 1, \ldots, p$. This has the property of being $\lfloor \eta \rfloor$ times mean-square differentiable. By setting $\eta = s + 1/2$ (where $s \in \mathbb{N}_0$) in eq.(19) the following simplification is obtained:

$$c_i(x_i, x_i') = \exp\left(-\frac{\sqrt{2\eta}|x_i - x_i'|}{\theta_i} \right) \frac{\Gamma(s+1)}{\Gamma(2s+1)} \sum_{i=0}^{s} \frac{(s+i)!}{(s-i)!\; i!} \left(\frac{\sqrt{8\eta}|x_i - x_i'|}{\theta_i} \right)^{s-i}.$$

Common choices for η are 3/2 and 5/2 (Tripathy et al. 2016, Despotovic et al. 2020, Tuo & Wang 2020) (to the point that these kernels are generally called "Matérn 3/2" and "Matérn 5/2"). It is a widely-used class of kernels noted for its ability to specify the smoothness of the Gaussian process through the control of the differentiability along its realisations (Borovitskiy et al. 2020).

The realisations from the squared exponential kernel are very smooth in contrast to those from the Matérn 3/2 and 5/2 as it has a form that is infinitely differentiable and is obtained from the Matérn by letting $\eta \to \infty$ in eq.(19) (Rasmussen & Williams 2006). The squared exponential takes the form

$$c_i(x_i, x_i') = \exp\left(-\frac{|x - x_i'|^2}{2\theta_i^2} \right), \tag{20}$$

which is defined by the lengthscale parameter θ_i of variable X_i. This kernel has been used by O'Hagan (2006) and Oakley & O'Hagan (2002) in emulating a simple one-dimensional function and Oakley & O'Hagan (2004) used it to demonstrate their sensitivity analysis methods on much higher dimensional functions. If the correlation function for all variables are chosen to be the squared exponential (eq.(20)), then eq.(18) can be written as

$$c(\mathbf{x}, \mathbf{x}') = \exp\left((\mathbf{x} - \mathbf{x}')^T B (\mathbf{x} - \mathbf{x}') \right)$$

where B is a $p \times p$ matrix which specifies the inverse of the lengthscales on the diagonal but allows for interaction between inputs also.

There is also the power-exponential kernel (Higdon et al. 2008) (also called γ-exponential in Rasmussen & Williams (2006)) given by

$$c_i(x_i, x_i') = \exp\left(-\left(\frac{|x - x_i'|}{\theta_i} \right)^{\gamma_i} \right),$$

where θ_i is the lengthscale for variable X_i and $0 < \gamma_i \leq 2$ but is only mean-square differentiable at $\gamma_i = 2$ (the squared exponential). This is to be used when wanting to specify the degree of smoothness in each input dimension where $\gamma_i = 2$ is smoothest and is less smooth the more γ_i decreases (Rasmussen & Williams 2006).

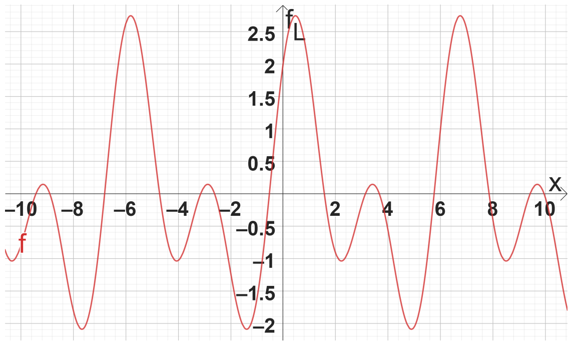
2.1.3 Estimating hyperparameters

Fitting the Gaussian process requires the estimation of the chosen kernel's hyperparameters θ (and γ in the case of the power-exponential (Kennedy & O'Hagan 2001)) which are estimated from the data $\mathbf{D} = (\mathbf{x_D}, \mathbf{y_D})$.

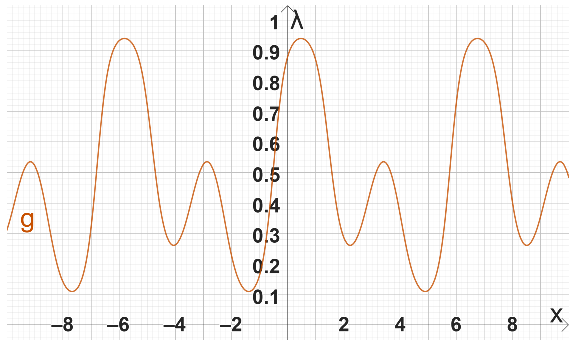
A common method is using maximum likelihood estimation (MLE), derived by Rasmussen & Williams (2006)(Ch 2):

$$\hat{\mathbf{\Theta}}_{\text{MLE}} = \arg\max_{\mathbf{\Theta}} \log L(\mathbf{y_D}|\mathbf{x_D}, \mathbf{\Theta}) = \arg\max_{\mathbf{\Theta}} \; -\frac{1}{2}\left(\mathbf{y_D}^T A_\tau^{-1} \mathbf{y_D} + \log|A_\tau| + n \log 2\pi \right), \tag{21}$$

where $A_\tau = A + \tau^2 I_n$ and τ^2 is the variance of the nugget (see section 2.1.1). This method is incorporated in the *R* package *DiceKriging* (Roustant et al. 2012).



(a) Example latent function $f_L(\mathbf{x})$.



(b) Squashed latent function $f_L(\mathbf{x})$ through the logistic function.

Figure 2: The process of squishing a latent function f_L through the logistic function λ. Notice that for all $\mathbf{x}$ such that $f_L(\mathbf{x}) > 0$, $\lambda(f_L(\mathbf{x})) \in (0.5, 1]$ and vice versa.

Alternatively, one can adopt a Bayesian approach by specifying priors on each of the hyperparameters $\mathbf{\Theta}$ and using the data $\mathbf{D} = (\mathbf{x_D}, \mathbf{y_D})$ to infer the posterior distribution (Higdon et al. 2004). The point estimate of the mode of this posterior is called maximum a posteriori (MAP) estimation which is represented as

$$\hat{\mathbf{\Theta}}_{\text{MAP}} = \arg\max_{\mathbf{\Theta}} \ \log L(\mathbf{y_D}, \mathbf{x_D}, \mathbf{\Theta}) g(\mathbf{\Theta}), \tag{22}$$

where $g(\mathbf{\Theta})$ is the prior over all hyperparameters and $L(\mathbf{y_D}|\mathbf{x_D}, \mathbf{\Theta})$ is represented in eq.(21). This method is incorporated in the Python package *mogp* (de Wolff et al. 2020).

The Gaussian process emulator can be fitted using the function *km* from the *R* package *DiceKriging* which utilises maximum likelihood estimation to estimate the optimal hyperparameters.

2.2 Gaussian process classification

A Gaussian process classifier implements a GP to make a probabilistic classification, in which test predictions take the form of class probabilities (Pedregosa et al. 2011). This is done using a two-step process: the first is placing a GP prior over a latent function $\mathbf{f}_L(\mathbf{x})$, the second being to squish this through a logistic link function $\lambda(\mathbf{x})$ to obtain a probabilistic classification (Rasmussen & Williams 2006). In figure 2, for a given latent function $\mathbf{f}_L$, this is squashed through the logistic function

$$\lambda(z) = \frac{1}{1 + \exp(-z)},$$

i.e. figure 2b is the plot of $\lambda(\mathbf{f}_L(\mathbf{x})$.

For a given test case $\mathbf{x}_*$, the distribution of the latent variable f_* is calculated by integrating out $\mathbf{f}_L$:

$$p(f_*|\mathbf{x_D}, \mathbf{y_D}, \mathbf{x}_*) = \int p(f_*|\mathbf{x_D}, \mathbf{x}_*, \mathbf{f}_L) p(\mathbf{f}_L|\mathbf{x_D}, \mathbf{y_D}) d\mathbf{f}_L, \tag{23}$$

where $p(\mathbf{f}_L, \mathbf{x_D}, \mathbf{y_D}$ is posterior for the latent variable. One can then make a probabilistic prediction using the distribution over f_*:

$$\hat{\pi}_* \triangleq p(y_* = +1|\mathbf{x_D}, \mathbf{y_D}, \mathbf{x}_*) = \int \lambda(f_*) p(f_*|\mathbf{x_D}, \mathbf{y_D}, \mathbf{x}_*) df_*, \tag{24}$$

where $\lambda(z) = 1/(1 + \exp(-z))$ is the sigmoid function.

The calculations in this subsection are done using the function *GaussianProcessClassifier* in the Python package *scikit-learn* (Pedregosa et al. 2011).

3 Multivariate emulation using PCA

The emulation approach used in this section is the use of singular vectors obtained through principal component analysis to model the output (Wilkinson 2010, Salter et al. 2019, Salter & Williamson 2022, Higdon et al. 2008, Sexton et al. 2012). After an ensemble of n model runs, the outputs can be represented in a $n \times q$ matrix $\mathbf{Y_D}$, where q is the number of outputs. In line with the procedure behind singular value decomposition (SVD) (see section 1.1), $\mathbf{Y_D}$ can be first centred and scaled (mean 0, variance 1), then decomposed in the following way:

$$\mathbf{Y_D} = \underset{n\times n}{U} \underset{n\times k}{\Sigma} \underset{k\times q}{V^T},$$

where $k = rank(\mathbf{Y_D}) \leq min(n, q)$, Σ is a diagonal matrix consisting of singular values, V are a set of singular vectors which form the rotation and $U\Sigma$ are the weights where in each row i, this represents the mapping of co-ordinates of output space onto the new basis for the ith row of $\mathbf{Y_D}$.

Looking at Σ, one can determine the proportion of variability in the data explained by each singular value by calculating,

$$\Lambda_i = \frac{\lambda_i^2}{\sum_{j=1}^{k} \lambda_j^2}, \tag{25}$$

where $\Sigma_{i,i} = \lambda_i$. From this one can select the first k' singular vectors which reach a threshold T_{var} of variance explainability and truncate the remaining $(k'+1), \ldots, k$ singular vectors. This can be easily calculated by setting k' to be the smallest value such that

$$\sum_{i=1}^{k'} \Lambda_i > T_{var}.$$

The emulation of these k' principal components can be represented by

$$E\left[\hat{\mathbf{f}}(\mathbf{x})\right] = \left(E\left[\hat{f}_1(\mathbf{x})\right], \ldots, E\left[\hat{f}_{k'}(\mathbf{x})\right]\right)^T, \tag{26}$$

$$\text{Var}\left[\hat{\mathbf{f}}(\mathbf{x})\right] = \text{diag}\left(\text{Var}\left[\hat{f}_1(\mathbf{x})\right], \ldots, \text{Var}\left[\hat{f}_{k'}(\mathbf{x})\right]\right). \tag{27}$$

Note that both eq.(26)-(27) are in k'-dimensional space. To represent them in the original q-dimensional space, one simply rotates back using the first k' columns of V.

$$\overset{q\times 1}{E\left[\hat{\mathbf{f}}^*(\mathbf{x})\right]} = \overset{q\times k'}{V_{1:k'}} \overset{k'\times 1}{E\left[\hat{\mathbf{f}}(\mathbf{x})\right]}, \tag{28}$$

$$\underset{q\times q}{\text{Var}\left[\hat{\mathbf{f}}^*(\mathbf{x})\right]} = \underset{q\times k'}{V_{1:k'}} \underset{k'\times k'}{\text{Var}\left[\hat{\mathbf{f}}(\mathbf{x})\right]} \underset{k'\times q}{V_{1:k'}^T} + \underset{q\times(k-k')}{V_{-1:k'}} \underset{(k-k')\times(k-k')}{\Sigma_{-1:k'}} \underset{(k-k')\times q}{V_{-1:k'}^T}, \tag{29}$$

where $\hat{\mathbf{f}}^*(\mathbf{x})$ is the emulator representing the full q-dimensional output. By inverting the scaling and centring of $E\left[\hat{\mathbf{f}}^*(\mathbf{x})\right]$ and $\text{Var}\left[\hat{\mathbf{f}}^*(\mathbf{x})\right]$, one obtains the emulator mean and uncertainty in the original output space.

Eq.(28) shows how the mean estimate in the full q-dimensional output is represented in the much smaller k' dimensions as the discarded singular vectors ($k - k'$ of them) are modelled as noise with zero-mean. Eq.(29) shows how the variability from these discarded singular vectors (term 2) can be combined with the uncertainty arising from the emulators representing singular vectors $1, \ldots, k'$ (term 1).

One may be tempted to perhaps emulate every output individually; however for large values of q, the method of using PCA provides several advantages:

1. the dramatic reduction in the number of outputs that need to be considered, in that a smaller number of emulators are required if one where to attempt to emulate each output separately;

2. these new outputs live on a new orthogonal basis V, meaning each principal component can be modelled independently using a GP (Higdon et al. 2008) without the need for inferring correlations between them;
3. lastly any history matching that is conducted on a q-sized output results in the inversion of a $q \times q$ matrix each time the implausibility is calculated however Salter & Williamson (2022) show that this implausibility can be calculated in $k' << q$ dimensions while still accounting for the full uncertainty in q dimensions.

The next section describes history matching and how it is done with one dimensional and q-dimensional outputs using singular vectors.

4 History Matching

4.1 Introduction

History matching is a procedure for identifying a collection of points for which the evaluation gives an acceptable match to an observation z (Vernon et al. 2010). Before this method, the determination of an acceptable match for z was done through trial and error by engineers running a model at random inputs, or through the use of least-squares Bayesian calibration (Craig et al. 1996). Now however, this process is simplified through the ruling out of different configurations of inputs until left with the set of points which give an acceptable match.

To compare the output of a model to the observation from reality requires accounting for two levels of uncertainty:

1. The observation z is a measurement taken of the real-world process $\mathbf{y}$ which therefore incurs the error induced by making this measurement. This is called *observation error.* This is expressed as
$$\mathbf{y} = z \oplus \mathbf{e}_O, \tag{30}$$
where $\mathbf{e}_O$ is the observation error (Bower et al. 2010).
2. There is also an error induced by modelling the real world process $\mathbf{y}$ using the model f evaluated at the 'perfect input' $\mathbf{x}^*$, which is called *model discrepancy.* This is expressed as
$$\mathbf{y} = \mathbf{f}(\mathbf{x}^*) \oplus \mathbf{e}_M, \tag{31}$$
where $\mathbf{e}_M$ is the model discrepancy.

The aim of history matching is to find all possible values $\mathbf{x}^* \in \chi$ (where χ represents the input parameter space) that satisfy eqns.(30)-(31). In the case that output size $q = 1$, these equations can be combined to give:
$$E\left[(z - f(\mathbf{x}^*))^2\right] = Var[e_O] + Var[e_M]. \tag{32}$$
Given the expensive nature of the model, evaluating f to find $\mathbf{x}^*$ is not feasible, therefore an emulator $\hat{f}$ can be used to approximate f. For a given $\mathbf{x}$, $\hat{f}(\mathbf{x})$ gives both a mean and variance therefore changing eqn.(32) to
$$\left(z - E\left[\hat{f}(\mathbf{x}^*)\right]\right)^2 = Var\left[\hat{f}(\mathbf{x}^*)\right] + Var[e_O] + Var[e_M]. \tag{33}$$
By comparing the ratio of the left-hand side to the right-hand side of eq.(33), this leads to a computationally cheap way to assess how implausible a given input $\mathbf{x}$ is. This ratio is just the centring and scaling using the posterior mean and covariance of $\hat{f}(\mathbf{x})$ (such that the standard deviation is 1); then assessing where the observation z sits within this scaled posterior. This ratio is called the *implausibility measure*:
$$I^2(\mathbf{x}) = \frac{\left(z - E\left[\hat{f}(\mathbf{x}^*)\right]\right)^2}{Var\left[\hat{f}(\mathbf{x}^*)\right] + Var[e_O] + Var[e_M]}, \tag{34}$$

where I denotes implausibility, with the higher implausibility implying a lower chance of a point $f(\mathbf{x})$ being close to z. This therefore involves establishing a threshold T by which $I(.)$ must be less than for any point $\mathbf{x}$ not to be deemed implausible given z:

$$\Omega = \{\mathbf{x} \in \chi : I(\mathbf{x}) < T\},$$

where χ represents the whole input parameter space. Many papers (Craig et al. 1997, Vernon et al. 2010, Iskauskas et al. 2022, Williamson et al. 2017) use a result from Pukelsheim (1994) which states that any univariate and unimodel distribution has 95% of the data in at most $\pm 3\sigma$ from the mean. For this reason $T = 3$ is widely used as the threshold.

The form of the implausibility $I(.)$ in eq.(34) is only for the case that $q = 1$.

A generalisation of eq.(34) for multi-dimensional outputs is the following, which can allow for correlated outputs, errors and discrepancies:

$$I^2(\mathbf{x}) = \left(\mathbf{z} - E\left[\hat{\mathbf{f}}(\mathbf{x})\right]\right)^T \left(Var\left[\hat{\mathbf{f}}(\mathbf{x})\right] + Var[\mathbf{e}_O] + Var[\mathbf{e}_M]\right)^{-1} \left(\mathbf{z} - E\left[\hat{\mathbf{f}}(\mathbf{x})\right]\right), \tag{35}$$

where the variance term on right-hand side (middle) is a $q \times q$ matrix to be inverted. This is equivalent to computing the Mahalanobis distance between the observation $\mathbf{z}$ and an input to $\hat{\mathbf{f}}(.)$. In this case, the choice of $T = 3$ is no longer appropriate given the higher dimensions. A commonly used threshold is $T^2 = \Omega^2_{q,0.995}$, which represents the 99.5th percentile of the Chi-squared distribution with q degrees of freedom (Vernon et al. 2010, Andrianakis et al. 2015, Salter et al. 2019). Often it is difficult to specify the full correlation structure for $Var[\mathbf{e}_O]$ and $Var[\mathbf{e}_M]$ so it is often assumed that these errors are uncorrelated so the matrices are diagonal (Andrianakis et al. 2015).

4.2 The algorithm

History matching is performed in iterations (commonly called waves). The process starts with the whole input space designated as NROY (not ruled out yet) space. A design is then sampled from this space, suitable emulators are built and using an implausibility measure, some input configurations are ruled out thereby defining a new NROY space. This process then repeats until either NROY is the empty set or no more space can be ruled out due to prescribed errors and discrepancies.

The following algorithm is consistent with the recently defined procedures:

1. Identify target outputs $\mathbf{z}$, model discrepancy $\mathbf{e}_M$ and observation error $\mathbf{e}_O$ using elicitation from model experts/engineers. Choose ranges for each input variable. Construct space filling design $\mathbf{D}_1$. Set wave number $j = 1$ and wave 0 NROY space $\Omega_0 = \chi$.

2. Build emulators $\hat{\mathbf{f}}(\mathbf{x})$ for each output.

3. Calculate implausibility (eq.(35)) $I_j(\mathbf{x})$ using $\mathbf{z}, \mathbf{e}_M, \mathbf{e}_O$ and emulators $\hat{\mathbf{f}}(\mathbf{x})$. Set $\Omega_j = \{\mathbf{x} \in \chi : I_1(\mathbf{x}), \ldots, I_j(\mathbf{x}) < T\}$.

4. Sample in the space Ω_j to obtain design $\mathbf{D}_{j+1}$.

5. If (a) $Var\left[\hat{\mathbf{f}}\right] << Var[\mathbf{e}_M] + Var[\mathbf{e}_O]$ OR (b) $\Omega_j = \emptyset$, proceed to step 6. Else return to step 1 and set $j = j + 1$.

6. If (a) then sample from Ω_j to obtain inputs close to target output. If (b) then restart process with higher model discrepancy or observation error.

4.3 Specifying model discrepancy and observation error

Stage 1 of the history matching process involves the setup of the procedure. Part of this stage involves choosing the model discrepancy $\mathbf{e}_M$ and observation error $\mathbf{e}_O$. This involves consulting and eliciting from model experts the values for these two terms. These discrepancies between the simulator and reality can be difficult to incorporate into the analysis (Craig et al. 1996).

In the case of one output ($q = 1$), this only requires the specification of one variance each for $\mathbf{e}_O$ and $\mathbf{e}_M$. In Dunne et al. (2022) the model discrepancy variance was chosen through the assumption that the model predicts the output to be within $\pm 20\%$ of the observation 95% of the time. This was done through consultation with the model developer.

For higher q, this requires the specification of a $q \times q$ sized matrix for both model discrepancy and observation error. In the case of these errors being uncorrelated (i.e. error in one of the outputs of a model doesn't affect another), then these matrices are diagonal; vastly reducing the elicitation required from experts. In Craig et al. (1997) for instance, the engineer was satisfied that the observation error could be modelled as uncorrelated; therefore only requiring $q = 77$ specifications.

In the case of correlated errors however, this requires the specification of the full $q \times q$ matrix for both observation error and model discrepancy which leads to obvious elicitation difficulties.

4.4 History matching using singular vectors

To incorporate the use of singular vectors in the emulation process, replace step 2 of history matching algorithm with the following sub-steps:

a Take the design output $\mathbf{Y_D}$ ($n \times q$) from the design $\mathbf{D}_j$, centre (to 0) and scale (to variance 1) and apply the singular value decomposition from eq.(2).

b Map the design output $\mathbf{Y_D}$ onto new basis: $\mathbf{Y'_D} = \mathbf{Y_D}V$.

c Select first k' columns of $\mathbf{Y'_D}$ according to eq.(25).

d Construct k' emulators $\hat{\mathbf{f}}^*(\mathbf{x})$, where the ith emulator is trained by the inputs $\mathbf{x_D}$ and the output being the ith column of $\mathbf{Y'_D}$. These emulators take the form of eqs.(26) & (27).

e Use eqs.(28) & (29) to represent the full q-dimensional output. Finally, reverse the scaling and centring.

For a point in input space to be deemed non-implausible, it needs to satisfy $I(\mathbf{x}) < T$, where T represents the threshold. Vernon et al. (2010), Andrianakis et al. (2015), Salter et al. (2019) all use a threshold of $T^2 = \chi^2_{q,0.995}$, which is the 99.5th percentile of the chi-squared with q degrees of freedom.

Upon construction of the emulator $\hat{\mathbf{f}}^*(\mathbf{x})$, history matching can be conducted. One may be tempted to history match in the k' ($<< k \leq q$) dimensional subspace (to lower the implausibility threshold) however Salter et al. (2019) show how one could unintentionally rule out parts of q-dimensional space consistent with $\mathbf{z}$ when history matching this way.

The first wave is carried out by applying eq.(35) across the input space and applying the threshold T thus defining the following space

$$\Omega_1 = \{\mathbf{x} \in \chi : I_1^2(\mathbf{x}) < T\},$$

where I_i^2 represents eq.(35) applied to the emulator from wave i.

This leaves a subspace Ω_1 of the original which is non-implausible given the data, observation and preprescribed errors. This space is then sampled, emulators built and implausibility applied to define

$$\Omega_2 = \{\mathbf{x} \in \chi : \mathbf{x} \in \Omega_1,\ I_2^2(\mathbf{x}) < T\},$$

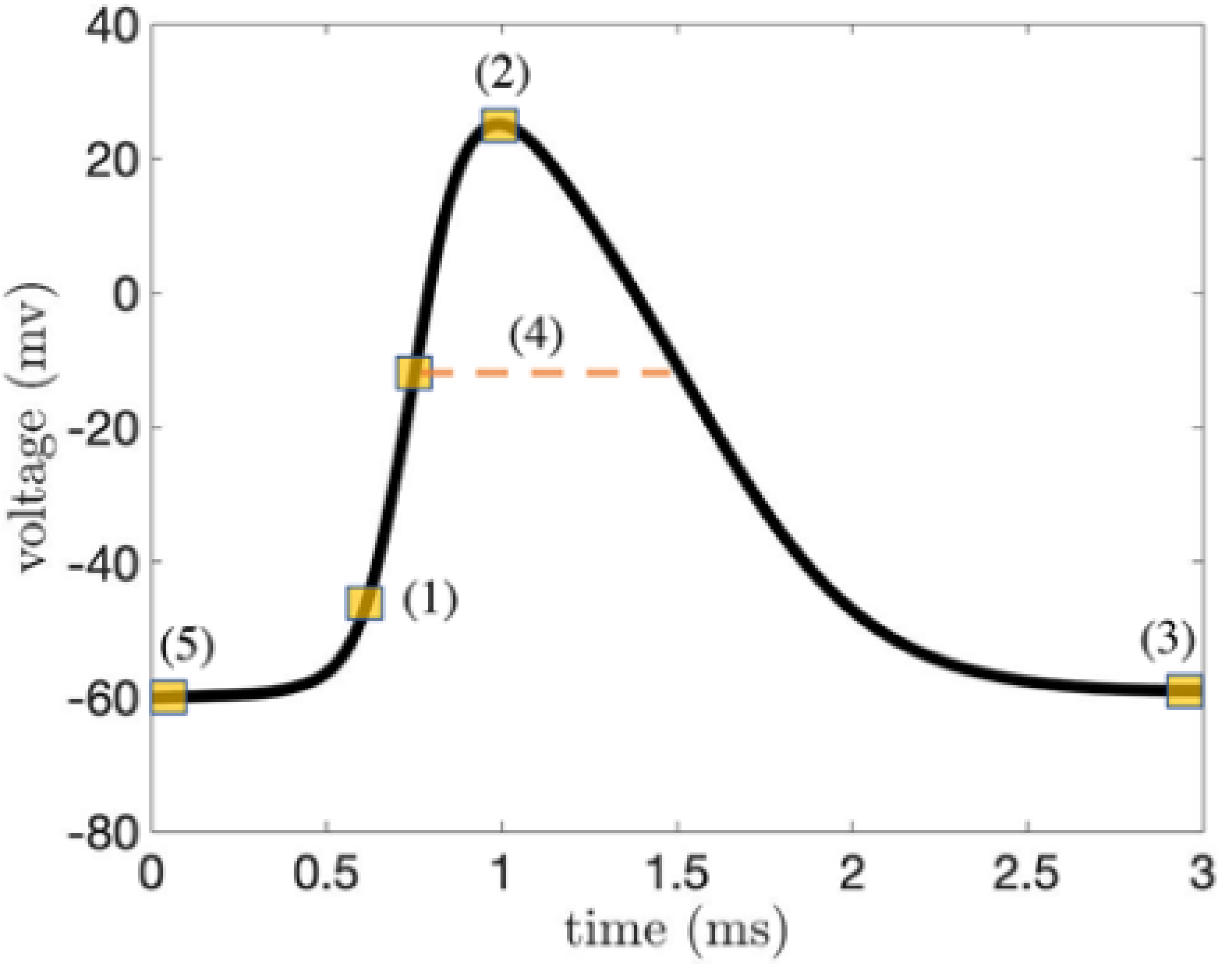


Figure 3: A curve describing an action potential from Saghafi et al. (2024). This example is characterised in 5 ways: (1): action potential threshold, (2): action potential peak; (3): action potential trough; (4): action potential width and (5): action potential minimum before pulse.

leading to the generalisation for wave i:

$$\Omega_i = \{\mathbf{x} \in \chi : \mathbf{x} \in \Omega_1, \dots, \Omega_{i-1},\ I_i^2(\mathbf{x}) < T\}.$$

History matching continues in these waves until either all space is ruled out (in which case the process restarts with a larger $\mathbf{e}_O$ or $\mathbf{e}_M$), or if the uncertainty from the emulation is much smaller than the pre-prescribed errors (signifying that the non-implausible space is almost as small as it could get).

5 Problem

The problem centres around having real-world data and wanting to find parameter values which give suitable match to this data when evaluated by model. This real-world data is an action potential which can be broken down into characteristics (see figure 3). However, the models that represent this process have two main outcomes: either a spike doesn't occur, therefore resulting in no other output values being produced; or a spike does occur, leading to the model producing output values corresponding to the characteristics of that spike. The problem is therefore that we have a simulator $\mathbf{y} = \mathbf{f}(\mathbf{x})$, where

$$\mathbf{y} = \begin{cases} \{\mathrm{y}_{class}, \mathbf{y}_{char}\} & \text{if } \mathbf{y}_{class} = 1 \\ \{\mathrm{y}_{class}\} & \text{if } \mathbf{y}_{class} = 0. \end{cases}$$

Therefore, the idea is to find suitable parameter values which when evaluated by $\mathbf{f}$ give characteristics close to the real-world data.

6 Framework to solve problem

The problems in this process are two-fold: one is finding the areas of parameter space that result in a spike, and the second is the characteristics of that spike conditional on if it occurs. In this report, these problems are treated separately; first a Gaussian process classifier is constructed to divide the parameter space into two regions (spike (1) and not spike (0)), and second, within the spike space, the characteristics of that spike is emulated using Gaussian process emulators.

6.1 Classification problem

A Latin hypercube design (Carnell 2021) is used to sample across parameter space. The output of this design is either a 1 (spike) or 0 (no spike). A Gaussian process classifier (see section 2.2) is constructed using this training data with the domain over the parameter space. Therefore, at a given point in this parameter space, a probability is given of a spike being induced.

The initial design is uniform, however in practise it is likely that different regions of parameter space require a higher concentration of points to clearly establish the boundary between spiking and not spiking regions. Because of this, after the first GP classifier is constructed, it may be useful to also sample close to the boundary of 0.5 (probability of spiking), therefore further samples are taken whose predicted probabilities are close to this value. This has the effect of pushing the probabilities across the parameter space towards 1 and 0 enabling a more reliable prediction.

Several iterations of this can be done, more clearly establishing the boundary between spiking and not spiking. In practise, a check at the proportion of points close to the boundary was taken and stopping criteria was initiated when the improvement after each new iteration was small.

6.2 Characteristic problem

The result from using the classifier is a region of parameter space where the probability of spiking is greater than 0.5. A more conservative approach (to perhaps avoid accidentally removing space that resulted in a spike but with predicted probability of lower than 0.5) can be used by choosing a probability lower than 0.5 or a more restrictive approach by choosing a higher probability.

In the space that is remaining (denoted Ω_0), the characteristics of the simulator output are emulated and history matching can be undertaken to rule out regions of parameter space deemed implausible given the model runs and target outputs.

Wave 1 is undertaken by sampling across the whole parameter space such that it is contained in Ω_0. This is done using the methodologies established in section 4.4. There are a number of characteristics that can be emulated (see figure 3) which therefore changes the number of outputs q and the implausibility threshold T. The result of this history matching wave reduces the space to Ω_1.

The next wave begins with sampling in Ω_1, constructing emulators whose domain is in this region and once again ruling out space using the implausibility measure. This in turn gives Ω_2. This process continues until the variance of the emulators is small compared to the pre-prescribed observation error and model discrepancy.

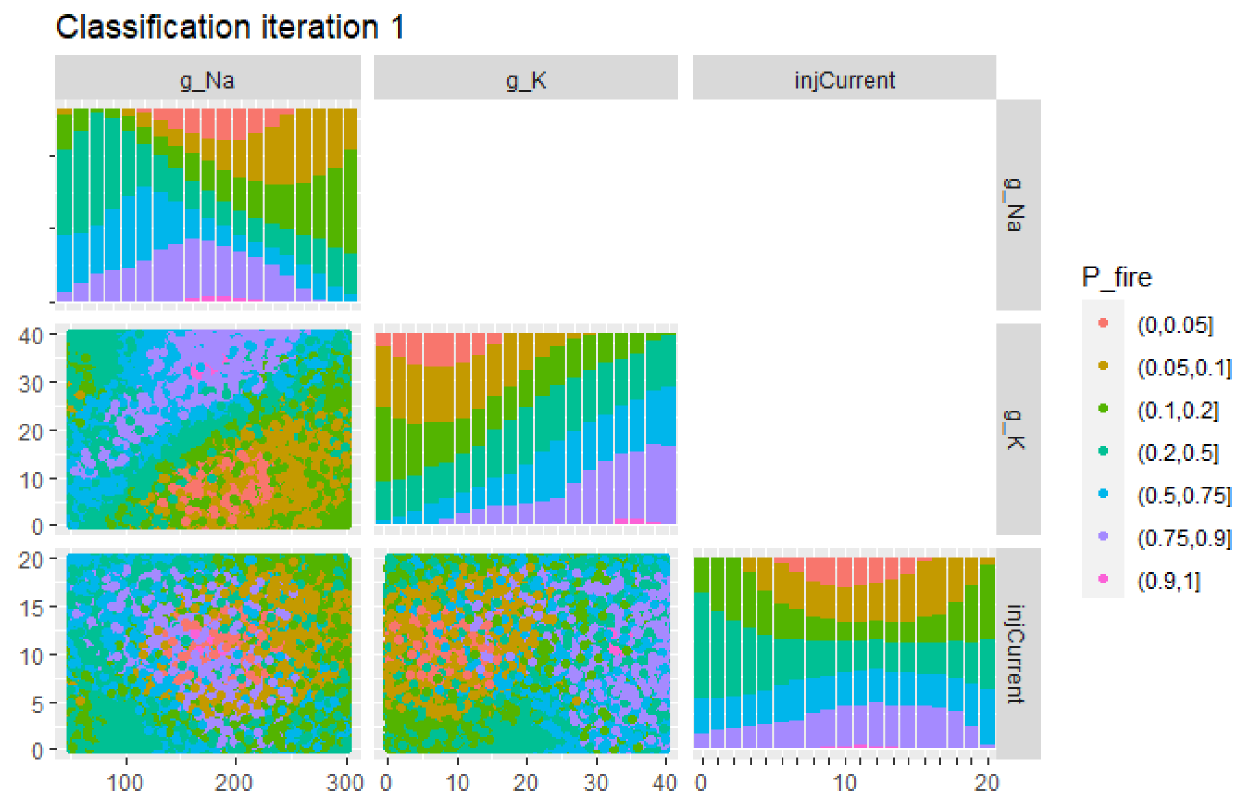


Figure 4: Iteration 1 of the Gaussian process classifier. The diagonal shows the probabilities across the marginal of each variable. The lower triangle shows the probabilities across each of the two-dimensional windows. The different colours represent probabilities which are sorted into bins listed to the right of the plot.

7 Example 1

7.1 Setup

Hodgkin-Huxley model compressed into 3 inputs: $g_{Na}, g_K, injCurrent$ with ranges $[50, 300], [0, 40], [0, 20]$ respectively. There are 6 outputs: *class, maxV, minV, width, minVbp, threshold.* One model run at a random input value is used to generate the observation to history match to: i.e. $\mathbf{y}^* = f(\mathbf{x}^*)$. For this example:

$$\mathbf{x}^* = (g^*_{Na}, g^*_K, injCurrent^*) = (163.62, 19.855, 3.0570)$$

and

$$\begin{aligned}\mathbf{y}^* &= (class^*, maxV^*, minV^*, width^*, minVbp^*, threshold^*)\\ &= (1, 43.329, -54.460, 1.2300, -54.460, -47.412).\end{aligned}$$

7.2 Classification GP results

Applying the methodology in section 6.1 for one iteration yields figure 4. Looking at the diagonal plot, one sees the proportion of parameter space which lies in the different probability bins. Much of the probabilities in the parameter space are between 0.2 and 0.75. Sampling from this space again starts to separate out these regions and move points away from the boundary.

By iteration 5, far more of these probabilities have moved away from the 0.5 boundary and pushed towards 0 and 1. Looking at the off-diagonal plots show how these different probabilities show the boundaries

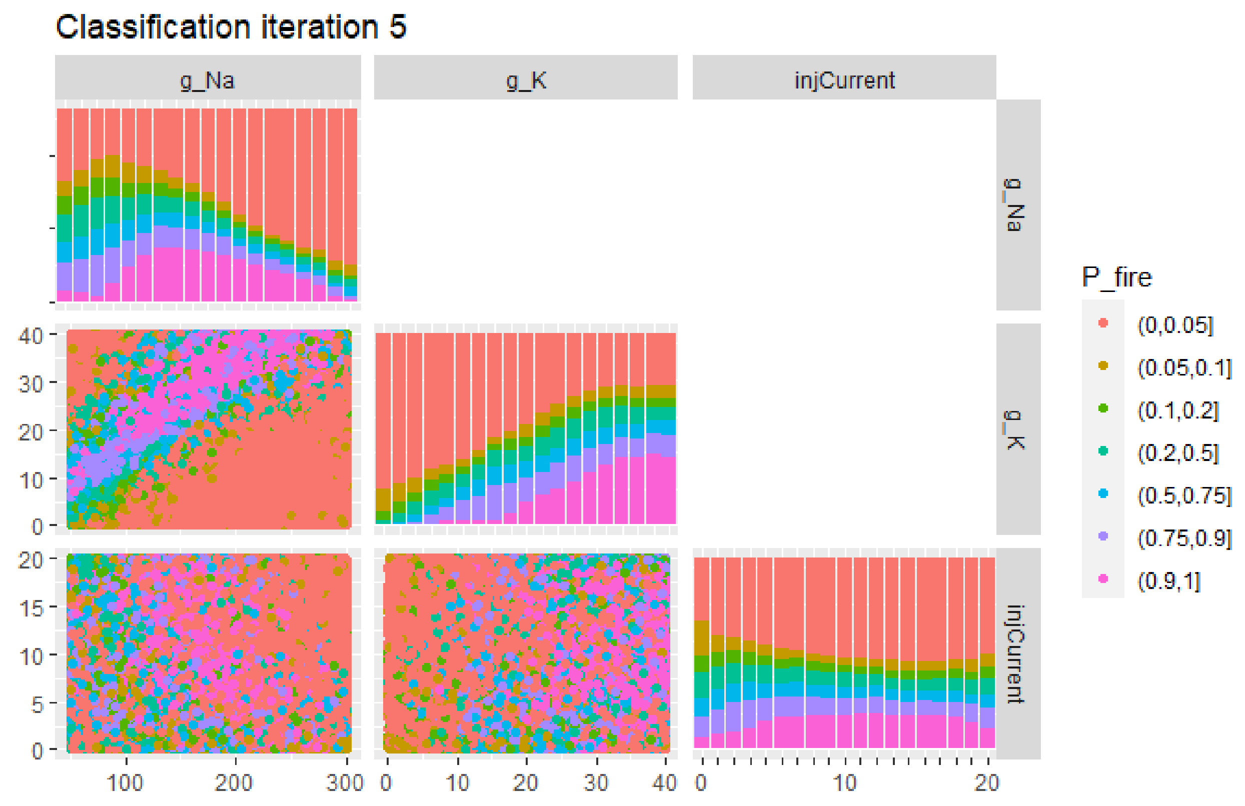


Figure 5: Iteration 5 of the Gaussian process classifier. The diagonal shows the probabilities across the marginal of each variable. The lower triangle shows the probabilities across each of the two-dimensional windows. The different colours represent probabilities which are sorted into bins listed to the right of the plot.

between firing and not firing.

After this point, the improvement in each new iteration was negligible therefore this was chosen to be the final iteration. Therefore the space remaining is defined as

$$\Omega_0 = \{\mathbf{x} : P(\mathbf{x} = fire) > 0.5\}.$$

This space is plotted in figure 6.

7.3 Characteristic GP results

Now remains the history matching procedure where the target output is $\mathbf{y}^*$. In practise, an observation error variance and model discrepancy variance would also be specified but as this example serves as a demonstration of the methods, instead these have been set to 10^{-6} for each output.

History matching will be done over the space defined by Ω_0. This space is sampled, evaluated by the emulator and thus able to define a new set Ω_1. After one wave of history matching, figure 7 shows the space remaining and figure 8 shows the space contained in Ω_2.

This continued until wave 15, in which approximately 99.9999% of parameter space has been removed. The zoomed in plot of $\Omega_1 5$ is shown in figure 9 with the input values scaled back to their true scales. Using the points contained in this plot, the emulator can be evaluated and a prediction made as to what $\mathbf{y}^*$ is. The guess whose mean was closest (using Euclidean norm) in the output space (after centring and scaling)

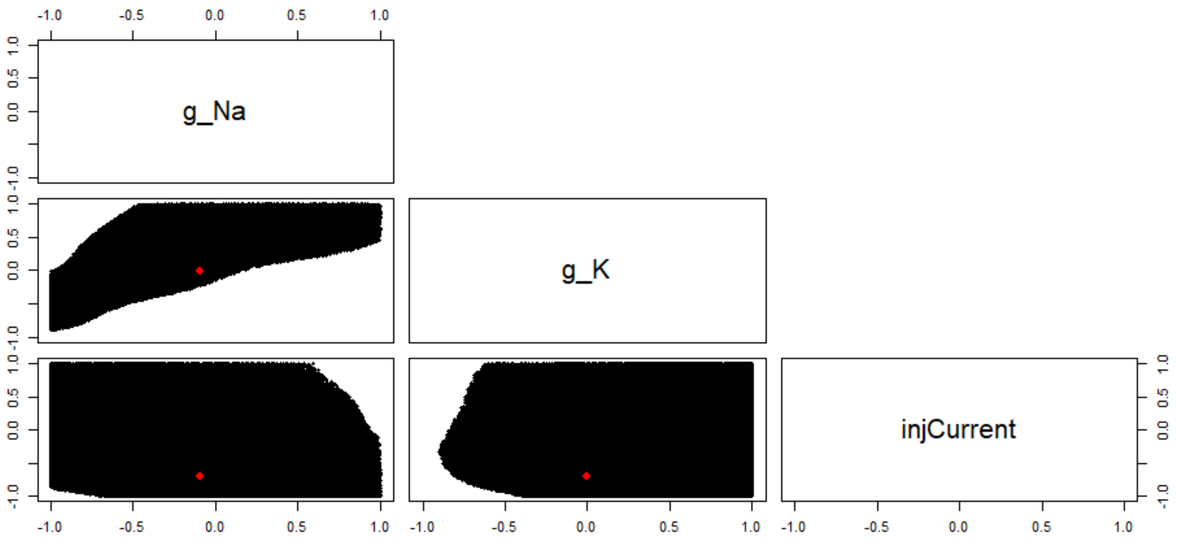


Figure 6: NROY space after using Gaussian process classifier. The red dot is $\mathbf{x}^*$. All inputs have been scaled to $[-1, 1]^p$.

resulted in a guess of $\mathbf{x}^*$ to be:

$$\hat{\mathbf{x}}^* = (166.58, 19.421, 3.5529),$$

which predicted the output

$$\hat{\mathbf{y}}^* = (1, 43.332, -54.466, 1.2327, -54.466, -47.423).$$

It is worth noting that to achieve this result, approximately 700 model runs were used in total throughout the classification and history matching phases.

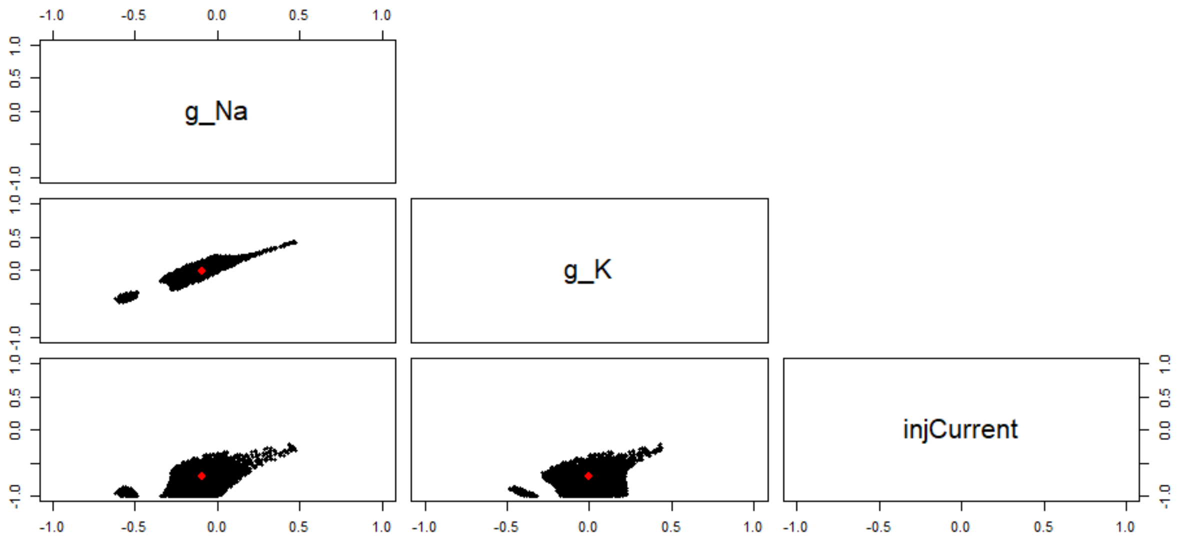


Figure 7: NROY space after 1 wave of history matching. The red dot is $\mathbf{x}^*$. All inputs have been scaled to $[-1, 1]^p$.

References

Andrianakis, I. & Challenor, P. G. (2012), 'The effect of the nugget on Gaussian process emulators of computer models', *Computational Statistics and Data Analysis* **56**(12), 4215–4228.
URL: *http://dx.doi.org/10.1016/j.csda.2012.04.020*

Andrianakis, I., Vernon, I. R., McCreesh, N., McKinley, T. J., Oakley, J. E., Nsubuga, R. N., Goldstein, M. & White, R. G. (2015), 'Bayesian History Matching of Complex Infectious Disease Models Using Emulation: A Tutorial and a Case Study on HIV in Uganda', *PLoS Computational Biology* **11**(1).

Baker, E., Challenor, P. & Eames, M. (2020), 'Predicting the Output From a Stochastic Computer Model When a Deterministic Approximation is Available', *Journal of Computational and Graphical Statistics* **29**(4), 786–797.

Borovitskiy, V., Terenin, A., Mostowsky, P. et al. (2020), 'Matérn gaussian processes on riemannian manifolds', *Advances in Neural Information Processing Systems* **33**, 12426–12437.

Bower, R. G., Vernon, I., Goldstein, M., Benson, A. J., Lacey, C. G., Baugh, C. M., Cole, S. & Frenk, C. S. (2010), 'The parameter space of galaxy formation', *Monthly Notices of the Royal Astronomical Society* **407**(4), 2017–2045.

Carnell, R. (2021), *lhs: Latin Hypercube Samples*. R package version 1.1.3.
URL: *https://CRAN.R-project.org/package=lhs*

Craig, P. S., Goldstein, M., Seheult, A. H. & Smith, J. A. (1996), 'Bayes Linear Strategies for Matching Hydrocarbon Reservoir History - ON MY DESK', *Bayesian Statistics 5* pp. 69–96.

Craig, P. S., Goldstein, M., Seheult, A. H. & Smith, J. A. (1997), 'Pressure Matching for Hydrocarbon Reservoirs: A Case Study in the Use of Bayes Linear Strategies for Large Computer Experiments', pp. 37–93.

Currin, C., Mitchell, T., Morris, M. & Ylvisaker, D. (1991), 'Bayesian prediction of deterministic functions, with applications to the design and analysis of computer experiments', *Journal of the American Statistical Association* **86**(416), 953–963.

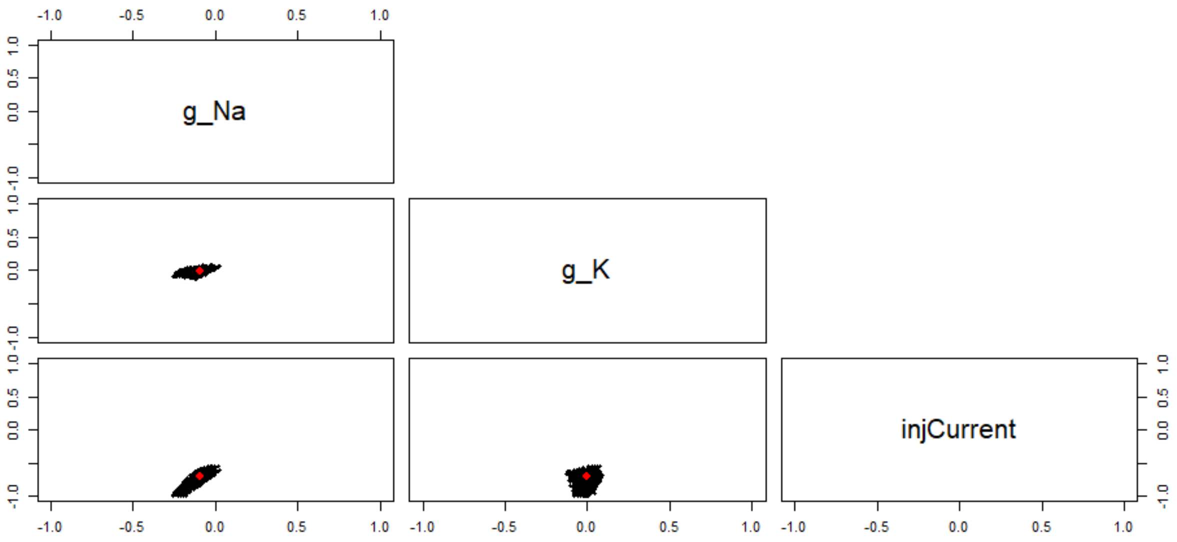


Figure 8: NROY space after 2 waves of history matching. The red dot is $\mathbf{x}^*$. All inputs have been scaled to $[-1, 1]^p$.

de Wolff, T., Cuevas, A. & Tobar, F. (2020), 'MOGPTK: The Multi-Output Gaussian Process Toolkit', *Neurocomputing* .
URL: *https://github.com/GAMES-UChile/mogptk*

Despotovic, V., Skovranek, T. & Schommer, C. (2020), 'Speech Based Estimation of Parkinson's Disease Using Gaussian Processes and Automatic Relevance Determination', *Neurocomputing* **401**, 173–181.
URL: *https://doi.org/10.1016/j.neucom.2020.03.058*

Dunne, M., Mohammadi, H., Challenor, P., Borgo, R., Porphyre, T., Vernon, I., Firat, E. E., Turkay, C., Torsney-weir, T., Goldstein, M., Reeve, R., Fang, H. & Swallow, B. (2022), 'Complex model calibration through emulation , a worked example for a stochastic epidemic model', *Epidemics* **39**(April), 100574.
URL: *https://doi.org/10.1016/j.epidem.2022.100574*

Higdon, D., Gattiker, J., Williams, B., Rightley, M., Higdon, D., Gattiker, J., Williams, B., Rightley, M., Igdon, D. H., Attiker, J. G., Illiams, B. W. & Ightley, M. R. (2008), 'Computer Model Calibration Using High- Dimensional Output', *American Statistical Association* **103**(482), 570–583.

Higdon, D., Kennedy, M. C., Cavendish, J. C., Cafeo, J. & Ryne, R. D. (2004), 'Combining Field Data and Computer Simulations for Calibration and Prediction', *SIAM Journal on Scientific Computing* **26**(2), 448–466.

Iskauskas, A., Vernon, I., Goldstein, M., Scarponi, D., McKinley, T. J., White, R. G. & McCreesh, N. (2022), 'Emulation and History Matching using the hmer Package', pp. 1–40.
URL: *http://arxiv.org/abs/2209.05265*

Joliffe, I. & Morgan, B. (1992), 'Principal component analysis and exploratory factor analysis', *Statistical Methods in Medical Research* **1**(1), 69–95. PMID: 1341653.
URL: *https://doi.org/10.1177/096228029200100105*

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

Krzanowski, W. J. (1988), *Principles of Multivariate Analysis*, Oxford University Press, USA, New York.

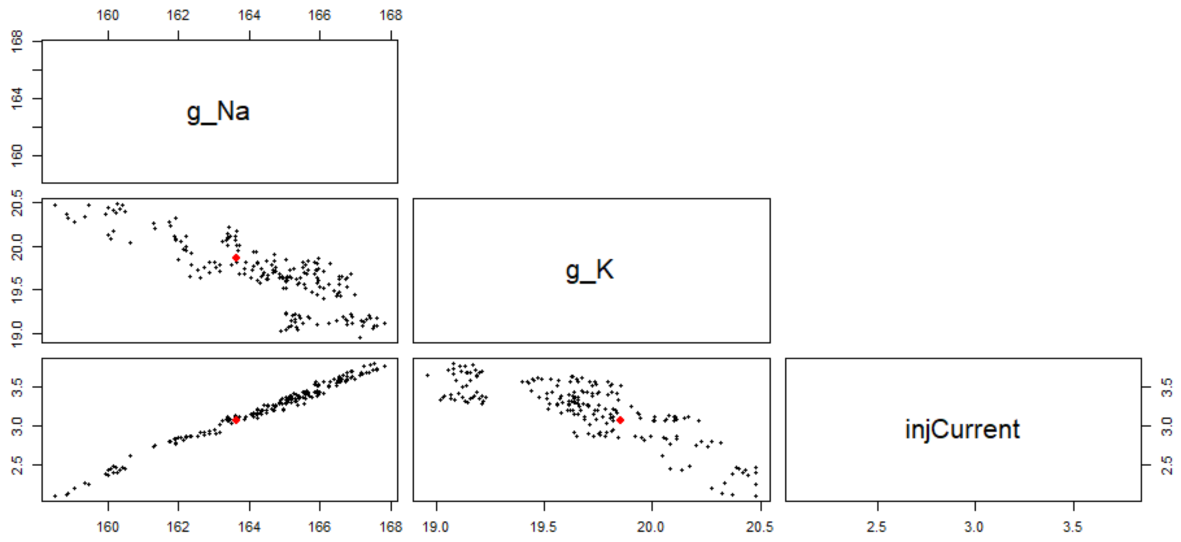


Figure 9: NROY space after 15 waves of history matching, zoomed in. The red dot is $\mathbf{x}^*$.

Matérn, B. (1947), 'Methods of estimating the accuracy of line and sample plot surveys.', *Swedish with English summary* .

Neal, R. M. (1997), 'Monte Carlo Implementation of Gaussian Process Models for Bayesian Regression and Classification', (9702), 1–24.
URL: *http://arxiv.org/abs/physics/9701026*

Oakley, J. E. & O'Hagan, A. (2002), 'Bayesian inference for the uncertainty distribution of computer model outputs', *Oxford University Press on behalf of Biometrika Trust* **89**(4), 769–784.

Oakley, J. E. & O'Hagan, A. (2004), 'Probabilistic sensitivity analysis of complex models: A Bayesian approach', *Journal of the Royal Statistical Society. Series B: Statistical Methodology* **66**(3), 751–769.

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

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M. & Duchesnay, E. (2011), 'Scikit-learn: Machine learning in Python', *Journal of Machine Learning Research* **12**, 2825–2830.

Pukelsheim, F. (1994), 'The three sigma rule', *The American Statistician* **48**(2), 88–91.

R Core Team (2013), *R: A Language and Environment for Statistical Computing*, R Foundation for Statistical Computing, Vienna, Austria. ISBN 3-900051-07-0.
URL: *http://www.R-project.org/*

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

Roustant, O., Ginsbourger, D. & Deville, Y. (2012), 'DiceKriging, DiceOptim: Two R packages for the analysis of computer experiments by kriging-based metamodeling and optimization', *Journal of Statistical Software* **51**(1), 1–55.
URL: *https://www.jstatsoft.org/v51/i01/*

Saghafi, S., Rumbell, T., Gurev, V., Kozloski, J., Tamagnini, F., Wedgwood, K. C. & Diekman, C. O. (2024), 'Inferring Parameters of Pyramidal Neuron Excitability in Mouse Models of Alzheimer's Disease Using Biophysical Modeling and Deep Learning', *Bulletin of Mathematical Biology* **86**(5), 1–29.
URL: *https://doi.org/10.1007/s11538-024-01273-5*

Salter, J. M. & Williamson, D. B. (2022), 'Efficient calibration for high-dimensional computer model output using basis methods', *International Journal for Uncertainty Quantification* **12**(6).

Salter, J. M., Williamson, D. B., Scinocca, J. & Kharin, V. (2019), 'Uncertainty Quantification for Computer Models With Spatial Output Using Calibration-Optimal Bases', *Journal of the American Statistical Association* **114**(528), 1800–1814.
URL: *https://doi.org/10.1080/01621459.2018.1514306*

Sexton, D. M., Murphy, J. M., Collins, M. & Webb, M. J. (2012), 'Multivariate probabilistic projections using imperfect climate models part i: outline of methodology', *Climate dynamics* **38**, 2513–2542.

Storch, H. V. & Zwiers, F. W. (1999), *Empirical Orthogonal Functions*, Cambridge University Press, p. 293–316.

Tripathy, R., Bilionis, I. & Gonzalez, M. (2016), 'Gaussian processes with built-in dimensionality reduction: Applications to high-dimensional uncertainty propagation', *Journal of Computational Physics* **321**, 191–223.
URL: *http://dx.doi.org/10.1016/j.jcp.2016.05.039*

Tuo, R. & Wang, W. (2020), 'Kriging prediction with isotropic matérn correlations: Robustness and experimental designs', *J. Mach. Learn. Res.* **21**(1).

Van Rossum, G. & Drake, F. L. (2009), *Python 3 Reference Manual*, CreateSpace, Scotts Valley, CA.

Vernon, I., Goldstein, M. & Bower, R. (2010), 'Galaxy formation: A Bayesian uncertainty analysis', *Bayesian Analysis* **5**(4), 619–670.

Wilkinson, R. D. (2010), 'Bayesian calibration of expensive multivariate computer experiments', *Large-Scale Inverse Problems and Quantification of Uncertainty* p. 195–215.

Williamson, D. B., Blaker, A. T. & Sinha, B. (2017), 'Tuning without over-tuning: Parametric uncertainty quantification for the NEMO ocean model', *Geoscientific Model Development* **10**(4), 1789–1816.