A framework for conducting uncertainty quantification on epidemiological models: a tutorial

Michael Dunne[1] and Peter Challenor[1]

[1]College of Engineering, Mathematics and Physical Sciences, University of Exeter, Exeter, UK

18th November 2022

Keywords: uncertainty quantification; principal component analysis; singular value decomposition; weighted singular value decomposition; Gaussian process; history matching; rotation

Contents

Key

Throughout this report we use the following key:

Key word/term	Denoted by
no. model runs	n
no. input (output) variables	p (q)
reduced no. input (output) variables	p' (q')
simulator	$f(\cdot)$
emulator	$\hat{f}(\cdot, Cov(\cdot,\cdot))$
composition of emulators	$g(\cdot, Cov(\cdot,\cdot))$
ensemble of inputs (outputs)	X (Y)
observation	$\mathbf{z}$

1 Introduction

In this section we explain (separately) the theory behind the methodology in this report: namely emulation and history matching in the uncertainty quantification framework and principal component analysis using weighted singular value decomposition. The way in which they are brought together is described in section 2.

1.1 Uncertainty Quantification

Real-world phenomena have been modelled by numerical models with more advanced features of these phenomena being accounted for as computing has become more advanced. In this case; simulating the spread of disease through a population. As more advanced features are accounted for these models take longer to run and thus the use of uncertainty quantification comes to the fore.
Uncertainty quantification (UQ) is often referred to as the connection from the mathematical and statistical analysis of complex models (also known as simulators and will be the default term from now on) to the real world (Swallow et al. 2022). In this case we are looking at statistical analysis in a Bayesian framework. One of the main facets to UQ is the use of emulators: a statistical representation of an emulator (O'Hagan 2006); trained through model runs on the simulator and acts as the surrogate for any analyses to be performed such as a sensitivity analysis or calibration as the emulator can be run almost instantaneously compared to the simulator (Lee et al. 2011).

1.1.1 Gaussian process emulators

The emulator of use throughout this project is the Gaussian process (GP) emulator. Through the Bayesian approach, we treat the model as an unknown function (it isn't unknown in this case however given that it isn't articulated through a set of equations we assume it is); we formulate a prior belief $\mathbf{h}(\mathbf{x})^{\mathbf{T}}\beta$ about the simulator which acts as a trend function (also called a prior mean) and then add in a 0-mean Gaussian process. In explicit terms we approximate the output of a simulator f by

$$\hat{f}(\mathbf{x}) = \mathbf{h}(\mathbf{x})^T\beta + GP(\mathbf{0}, \sigma^2 c(\mathbf{x}, \mathbf{x}')),$$

with a prior specification on the basis vectors $\mathbf{h}$ (for this report $\mathbf{h}(\mathbf{x}) = 1$ due to lack of knowledge on effects of variables in the model) and the coefficients are inferred from the training data. As a result of the addition of the GP, $\hat{f}$ has a posterior mean and posterior covariance function meaning for any given input, one receives a normal probability distribution as an output with a mean and variance. The emulator is updated using model runs of the simulator (Oakley & O'Hagan 2004) and the posterior mean of a GP is guaranteed to pass through these design points and the uncertainty drop to (close to) 0 (not true when emulating stochastic models, but we aren't emulating those here). Away from the design points a covariance function is used which is of the form $\sigma^2 c(\mathbf{x}, \mathbf{x}')$ which decreases as $|\mathbf{x} - \mathbf{x}'|$ increases and satisfies $c(\mathbf{x}, \mathbf{x}') = 1$ (Oakley &

O'Hagan 2002). Away from design points the covariance function regresses the posterior mean function back to the prior mean.
The emulation is implemented in the R package *DiceKriging* (Roustant et al. 2012). For a full list of formulae regarding the building of the GP emulator, the specification of the kernel and the optimisation of the hyperparameters of that kernel see section 6.1.

1.1.2 History matching

Bayesian history matching is a facet of UQ and is a pre-calibration method. It is the process of sequentially ruling out regions of input space that are inconsistent with an observation $\mathbf{z}$ (Dunne et al. 2022). This observation represents a measurement of a real system $\mathbf{y}$ allowing us to represent it via the formula

$$\mathbf{z} = \mathbf{y} + \mathbf{e},$$

with $\mathbf{e}$ being the *observation error* (Bower et al. 2010). There is also a second discrepancy to account for and that is the discrepancy between the model and the real system value $\mathbf{y}$ via the formula

$$\mathbf{y} = \mathbf{f}(\mathbf{x}^*) + \epsilon$$

with ϵ being the *model discrepancy*. The aim of history matching is to find all possible values $\mathbf{x}^*$ that satisfy the combined equation

$$\mathbf{z} = \mathbf{f}(\mathbf{x}^*) + \mathbf{e} + \epsilon. \tag{1}$$

We define the set of all possible values $\mathbf{x}^*$ by $\Omega(\mathbf{z})$.
The model however is too expensive to evaluate directly to construct $\Omega(\mathbf{z})$ therefore - using the methods from section 1.1.1 - we build a Gaussian process emulator to approximate $f(\mathbf{x})$ and thus to construct $\Omega(\mathbf{z})$ whilst adding the emulator uncertainty to eq.(1) via the equation

$$|\mathbf{z} - E[\hat{\mathbf{f}}(\mathbf{x}^*)]| = Var[\hat{\mathbf{f}}(\mathbf{x}^*)] + \mathbf{e} + \epsilon.$$

For a multi-dimensional output (specifically q dimensions), we define a measure for how implausible a given input $\mathbf{x}$ is to evaluate to give the output $\mathbf{z}$ to be:

$$I(\mathbf{x})^2 = \Big(\mathbf{z} - E[\hat{\mathbf{f}}(\mathbf{x})]\Big)^T \Big(Var[\hat{\mathbf{f}}(\mathbf{x})] + Var[\epsilon] + Var[\mathbf{e}]\Big)^{-1} \Big(\mathbf{z} - E[\hat{\mathbf{f}}(\mathbf{x})]\Big) \tag{2}$$

where $\hat{\mathbf{f}}(\mathbf{x}) = \Big(\hat{f}_1(\mathbf{x}), \ldots, \hat{f}_q(\mathbf{x})\Big)$ represents the evaluation of all q emulators built for q outputs. A cut-off value T is required such that if $I(\mathbf{x}) < T$ then $\mathbf{x}$ is non-implausible and implausible if $I(\mathbf{x}) > T$. For a q-dimensional output, Vernon et al. (2014) and Salter et al. (2019) define the square of the implausibility threshold to be $T^2 = \chi^2_{q,0.995}$, which is the 99.5th percentile of the χ^2 distribution with q degrees of freedom. The history matching process is conducted in waves and the algorithm used in Vernon et al. (2018) is used in this report. The algorithm is as follows:

1. Set $\Omega_0(\mathbf{z})$ to represent the full input space at wave '0' (i.e. before the history matching process has started). In this report we discritise this space by generating 10^6 points using Latin Hypercube Sampling (LHS) (Mckay et al. 1979) and whatever proportion of those points remain in $\Omega(\mathbf{z})$ is the proportion of space remaining that could feasibly give the observation $\mathbf{z}$. Set wave number $k = 0$.

2. Sample within the space defined by $\Omega_k(\mathbf{z})$ and evaluate those samples using the model. The number of samples chosen at each wave is $10p$ where p is number of input dimensions (recommended by Loeppky et al. (2009) as a rule of thumb) though one can raise or lower the number of design points depending on the complexity of the model.

3. Decide which outputs to build emulators for. Set q to be the number of outputs.

4. Use the samples to construct one GP emulator to each of the q outputs (see section 1.1.1).

5. Using the measure defined in eq.(2), calculate the implausibility $I(\mathbf{x})$ of each point in the discritised space $\Omega_k(\mathbf{z})$. The space remaining is defined as

$$\Omega_{k+1} = \{\mathbf{x} \in \Omega_k \ : \ I(\mathbf{x})^2 < T^2\}.$$

6. Stopping criteria; if: 1) the variance of the emulator(s) is much lower than the pre-specified model discrepancy and observation error or 2) $\Omega_{k+1}(\mathbf{z}) = \emptyset$ then stop. Else set $k = k + 1$ return to step 2.

7. If 1) then set $\Omega(\mathbf{z}) = \Omega_{k+1}$ and sample from there with the knowledge that these runs closely match the observation. If 2) then return to step 2 and increase model discrepancy and/or observation error.

Steps $2 - 6$ constitute one wave/iteration (not to be confused with an epidemiological wave)) of history matching.
History matching has its uses as a pre-calibration method. After each wave, an implausibility function is generated (step 5) which acts as an indicator function where either a point is implausible or non-implausible in that wave given the observation. If, when calibrating towards the observation, one chooses a point (via a search algorithm for instance), if that point is evaluated through every indicator function and is non-implausible each time, then it is a point worth evaluating. However if it implausible on at least one indicator function then it isn't worth evaluating. This can greatly speed up the calibration process as instantly we can reject a sample subject to these criteria.
Also, by looking at the marginal distributions for each pair of variables and individually (on the condition that each point in the distribution is non-implausible) we can see relationships and dependencies form between them as we rule out space. The shape of these distributions can firstly act as a verification that the process is working well (if the dependencies are sensible) but also to narrow the ranges of individual variables which can aid the model developer for future model runs.

1.2 Principal component analysis

Principal component analysis (PCA) is a technique for reducing the dimensionality of a dataset whilst minimising the loss of information (Jolliffe et al. 2016).
Without reducing the dimensionality, for a large number of outputs q this means building q emulators which is inefficient for two reasons: 1) the computational power it would take to build these emulators would involve inverting q matrices of size $n \times n$ so resources could be better spent elsewhere and 2) the outputs are not independent from each other so one would need to propose how the outputs interact with each other which would introduce further uncertainty. Therefore reducing the dimensionality of our output is favourable.
PCA is conducted on a dataset and involves the creation of new and uncorrelated variables that maximise variance (Jolliffe et al. 2016) whereby as more variables are added the higher percentage of variance of that dataset is explained. The creation of these variables can be done through the use of singular-value decomposition (SVD) which is used to calculate basis vectors and weights (Salter et al. 2019, Higdon et al. 2008) for a given ensemble. Take an ensemble Y of size $n \times q$ which is the result of n model runs where each run gives q outputs. Through SVD one can decompose the matrix Y into the following form:

$$Y = U\Sigma V^T, \tag{3}$$

where U of size $n \times n$ and V of size $q \times q$ are orthogonal unitary matrices and Σ of size $n \times q$ is a rectangular diagonal matrix with all $N = min\{n, q\}$ entries being non-negative and on the diagonal. U, V and Σ can be found by calculating the eigenvalues and eigenvectors of Y^TY (or YY^T) and then (forward or back) solving for the remaining matrix. From this process, the basis vectors are V^T and the weights are $U\Sigma$. One truncates (thereby reducing the dimensionality) by choosing q' (where $q' << q$) such that the basis vectors become the first q' **columns** of V^T (a.k.a. $V_{q'}^T$) and the weights (also called principal components) become the first q' **rows** of $U\Sigma$ (a.k.a. $(U\Sigma)_{q'}$). One can be aided in choosing q' by calculating the proportion of total variance explained by each principal component:

$$P_i = \frac{\lambda_i^2}{\sum_{i=1}^{p} \lambda_i^2}. \tag{4}$$

This enables one to choose the number of principal components with the knowledge that one may overfit by choosing too many (and also wasting computational resources from emulating variables that explain very little variability). One may also underfit by choosing too few principal components leading to poor emulation. One can map a vector ξ onto the space of principal components using the following formula:

$$w(\xi) = (V^T V)^{-1} V^T \xi. \tag{5}$$

An example of PCA through SVD is shown in section 6.4.
Weighted singular value decomposition (WSVD) enables one to put a constraint on the basis functions V^T from eq.(3) through a weight W which gives a measure of importance for the outputs. The constraint is

$$V^T W V = I_q, \tag{6}$$

where I_q is the identity matrix of size q and the weight matrix W is of size $q \times q$ (Jolliffe 2002, Salter et al. 2019). Note that eq.(5) becomes

$$w(\mathbf{z}) = (V^T W^{-1} V)^{-1} V^T W^{-1} \mathbf{z}$$

when including the weight matrix W (Salter et al. 2019).

1.2.1 Rotation of basis

When history matching one builds one emulator for each chosen output. Using principal component analysis, one can reduce the dimensionality of that output by truncating the basis vectors such that a chosen amount of variance is explained. This means less emulators to build and thus saves computational resources. However by performing this truncation one loses the ability to span all of the possible outputs as a small amount of variability won't be explained. Take an observation. If this observation lies in the space that is not spanned then it cannot be reconstructed via the truncated basis. This is a problem when history matching towards that observation on that truncated basis as all input space would be ruled out (see *terminal case* in Salter et al. (2019)) as there would be no set of weights within that space which would give that observation.
A solution to this problem is proposed by Salter et al. (2019) which is to rotate the basis vectors through the use of optimisation which minimises the error of reconstructing the observation $\mathbf{z}$ whilst maintaining a level of explainability of variance across the ensemble Y. See section 6.2 for details on the algorithm for rotating the basis.

2 Framework and methodology

In this section we define the framework used in this report which takes all the facets of uncertainty quantification and principal component analysis explored in section 1. This section is divided into exploring how inputs, outputs, post-processing of outputs using PCA, emulation and history matching fit in to the framework. See section 6.3 to see how these sections of the framework can be used in R.

2.1 Inputs

Models have many inputs, some of which are to be fixed some of which are to be varied. Specifying which inputs should fall into the experiment should stem from a discussion with the model developer. Each input requires an upper and lower range in which to sample from. One should look to capture as much uncertainty as possible in their input ranges so when emulating one can quantify the uncertainty as accurately as possible. Some inputs can have natural ranges; for instance a probability, proportion, or reduction factor is between $(0, 1)$ however others will require discussion with the model developer.

2.2 Outputs, principal component analysis and emulation

The models to which this framework is applied have large numbers of outputs. Emulation of multiple outputs has been done by constructing one emulator per output (Vernon et al. 2014, Dunne et al. 2022), however with a large amount of outputs this would be computationally expensive. After sampling the model, we have

an ensemble Y consisting of n model runs each with q outputs. We can therefore use principal component analysis to reduce the number of outputs (see section 1.2) to a more 'manageable' amount that still explain the majority of the variance of the original outputs.
As a result, we have n model runs each with q' outputs where $q' << q$. An emulator is constructed for each of the q' outputs where it is the weights of the basis vectors that are being emulated. Therefore one can emulate the full output using

$$\mathbf{g}(\mathbf{x}) = \sum_{i=1}^{q'} \omega_i(\mathbf{x})\phi_{\mathbf{i}} + \epsilon \tag{7}$$

where $\mathbf{g}$ is the predicted output, ω_i is the predicted weight using the emulator for the ith principal component, $\phi_{\mathbf{i}}$ is the ith basis vector as calculated using PCA and ϵ is an error term from the PCA process.

2.3 History matching and rotation of bases

The issue with history matching on the truncated basis is the possibility of a terminal case (all space ruled out when history matching) as not all of the original output space is spanned after the truncation (see section 1.2.1 and Salter et al. (2019)). Therefore prior to history matching, we rotate the bases and truncate them such that the observation can be reconstructed with minimal error whilst maintaining a level of explainability of variance across the ensemble Y.
Therefore at each wave between steps 2 and 3 of the history matching algorithm (see section 1.1.2) we apply the process of principal component analysis on the ensemble Y to obtain the basis vectors, rotating those basis vectors and finally truncating appropriately. These are the outputs q' ($<< q$) which emulators are being built for.

We now demonstrate the methodology from this section with 2 examples in sections 3 and 4.

3 Model 1: Weight, Scale and Shift for epidemic modelling

3.1 Model details

The Weight, Scale and Shift (WSS) model[1] takes current real-world case data and (among other things) predicts current and future R numbers (often days/weeks before other R number predictions which are based on hospitalisations/deaths) and from there predicts hospitalisations and deaths for a number of days into the future across 10 regions in the UK. In this framework we are only considering the number of deaths in one region: London.

3.1.1 Inputs

This model can be split into two parts: the first part using the case data to predict the R number and the second part using that R number to predict future cases, hospitalisations and deaths. As the first part can be represented by an R number then by treating this as a variable we can span all possible case data.
Knowing this we can now bring together all the inputs:

1. this model is compartmental with those being: 1) Mild; 2) ILI (influenza-like illness); 3) SARI (severe acute respiratory illness); 4) Critical; 5) Death; 6) Recovery and 7) Critical Recovery. There are 8 model parameters which is the logmean number of days it takes someone to go between the following compartments: $1 \to 6$, $2 \to 6$, $2 \to 3$, $3 \to 6$, $3 \to 5$, $3 \to 4$, $4 \to 7$, $4 \to 5$ and $7 \to 6$ with $4 \to 7$ and $4 \to 5$ having the same logmean.

2. 5 coefficients that dictate the severity and transmissability of the alpha, delta and omicron variants (except transmissability of omicron)

3. vaccine efficacy

[1] https://github.com/gjackland/WSS

Var Names	Lower	Upper
logmeanMild2R	log 8	log 15
logmeanILI2R	log 8	log 15
logmeanILI2SARI	log 8	log 15
logmeanSARI2R	log 8	log 15
logmeanSARI2D	log 8	log 15
logmeanSARI2Crit	log 8	log 15
logmeanCrit2CritRecov	log 8	log 15
logmeanCritRecov2Recov	log 8	log 15
Kentfac	0	1
Indiafac	0	1
Omicronfac	0	1
Kenttrans	0	1
Indiatrans	0	1
vacCFR	0	1
R_decay	0	1
R_no	0	6
RawCFR_beta1	0	5
RawCFR_beta2	0	5

Table 1: This table shows the names and the ranges of each input variable in the WSS model.

4. rate at which R number decays back to 1
5. 2 beta parameters that define case-fatality-ratio (CFR) for each age-group via the use of the beta distribution.
6. *R_no* from first part

Table 1 shows each variable and their corresponding ranges.

3.1.2 Outputs

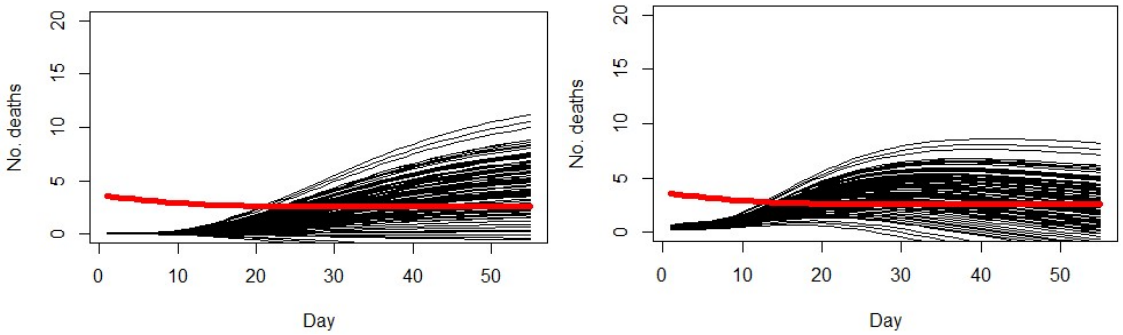
We have imposed a cut off of 55 days after the R number has been set and confined the output to the one region of the UK. More specifically we are looking at the time period between 01/06/22 and 25/07/22 in London. So to be clear: one model run will take 18 inputs and its output will be a time series of 55 days.

3.2 Application of framework

The range of deaths resulting from the parameter ranges specified in table 1 is vast. For certain combinations we can reach deaths of the order of $O(10^{18})$ hence for the purposes of emulation and history matching the methods of the inclusion of those outlier design points differ. We leave those outlier design points out of the building of the emulator as we know those omitted regions of input space won't need to be evaluated for modelling purposes and if they were included would inflate the uncertainty in other regions of space. For history matching, we need to remove regions of input space that we know won't evaluate to give a particular observation; so omitting those outlier design points won't work as the emulator won't know that those inputs give those outliers. Additionally, to aid in reducing the variability stemming from those large outputs, we put a cap of 10^6 on the number of deaths on any given day.

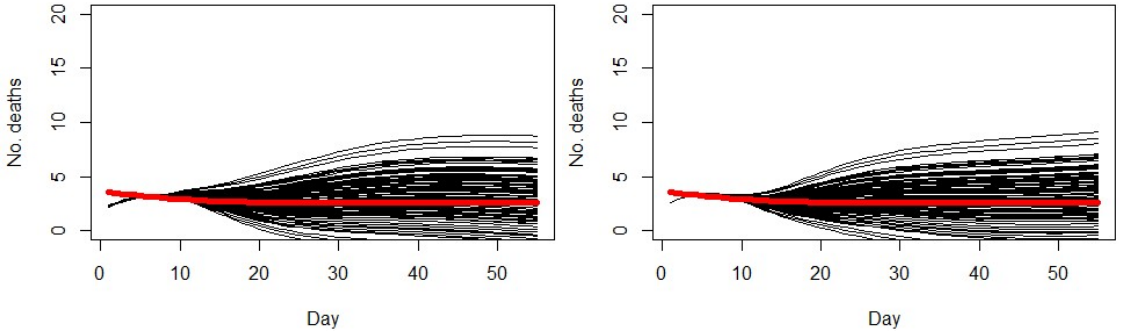
3.2.1 Emulation

We are able to represent the output time series using a number of orthogonal basis vectors and weights using PCA via a weighted singular value decomposition (WSVD). We use the sum of the observation error and model discrepancy on the diagonal of our weight matrix W (see eq.(6)). After this process we have q' basis vectors with the q' weights being the output after the post-processing.



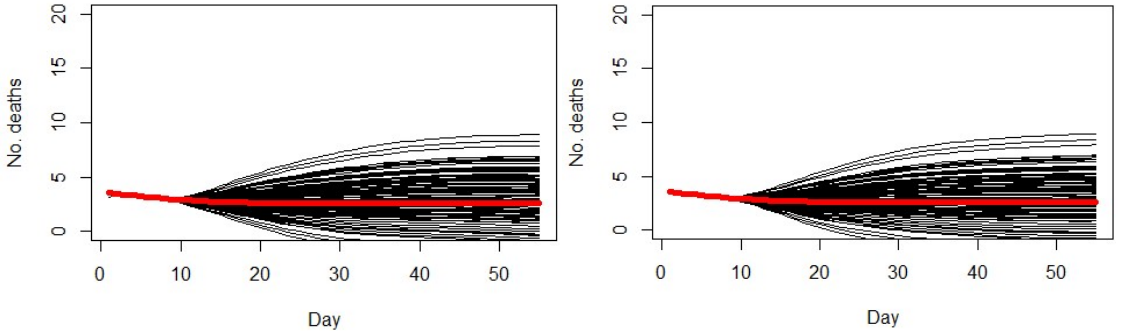
(a) 1 basis vector

(b) 2 basis vectors



(c) 4 basis vectors

(d) 5 basis vectors



(e) 6 basis vectors

(f) 7 basis vectors

Figure 1: These figures show a validation point (in red) compared to 100 draws from the posterior mean and standard deviation when using 1,2,4,5,6 and 7 basis vectors resulting from principal component analysis via a weighted singular value decomposition.

We use $n = 10p$ design points and fit an emulator to each of these weights (which are now independent from each other as their respective basis vectors are orthogonal). This enables us to (when given an input $\mathbf{x}$) predict q' weights and using the corresponding q' basis vectors reconstruct the full output time series.
Figure 1 demonstrates the importance of the number of basis vectors included in the emulation. By choosing only 2 (as shown in figure 1) we capture the output uncertainty towards the end of the time series however it misses the mark completely with the start. As we progress with more basis vectors more of the validation point (in red) is within the uncertainty bounds of the emulation up to figure 1f. Looking at figure 2 we see the proportion of variance explained by each principal component (see eq.(4)). Given that the first two principal components explain the vast majority of variability in the data, one may expect the emulation seen in figure 1 to have more accuracy. But given the previously discussed point on the variability in the output, this level of explanation may not be enough. Hence we increase our number of basis vectors to account for

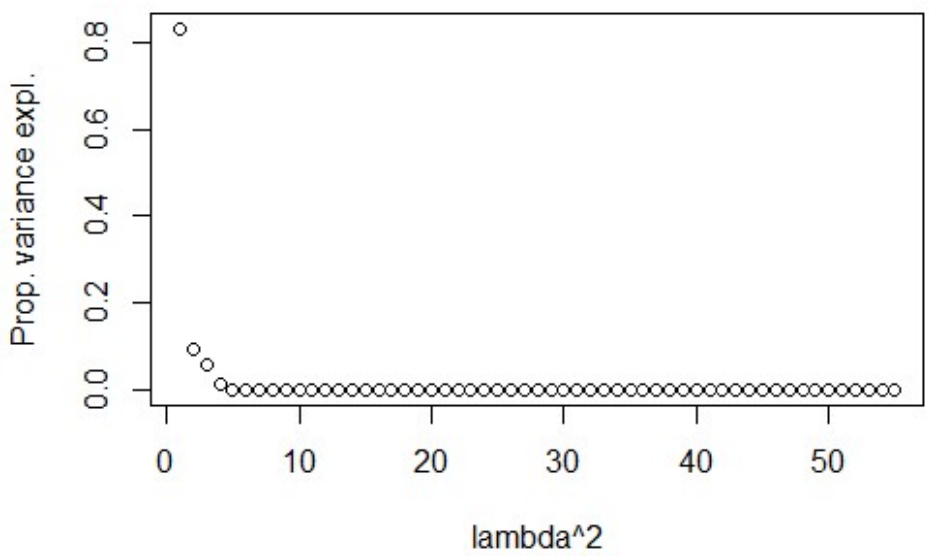


Figure 2: This figure shows the proportion of variance explained by each basis vector calculated using eq.(4).

this large variability.

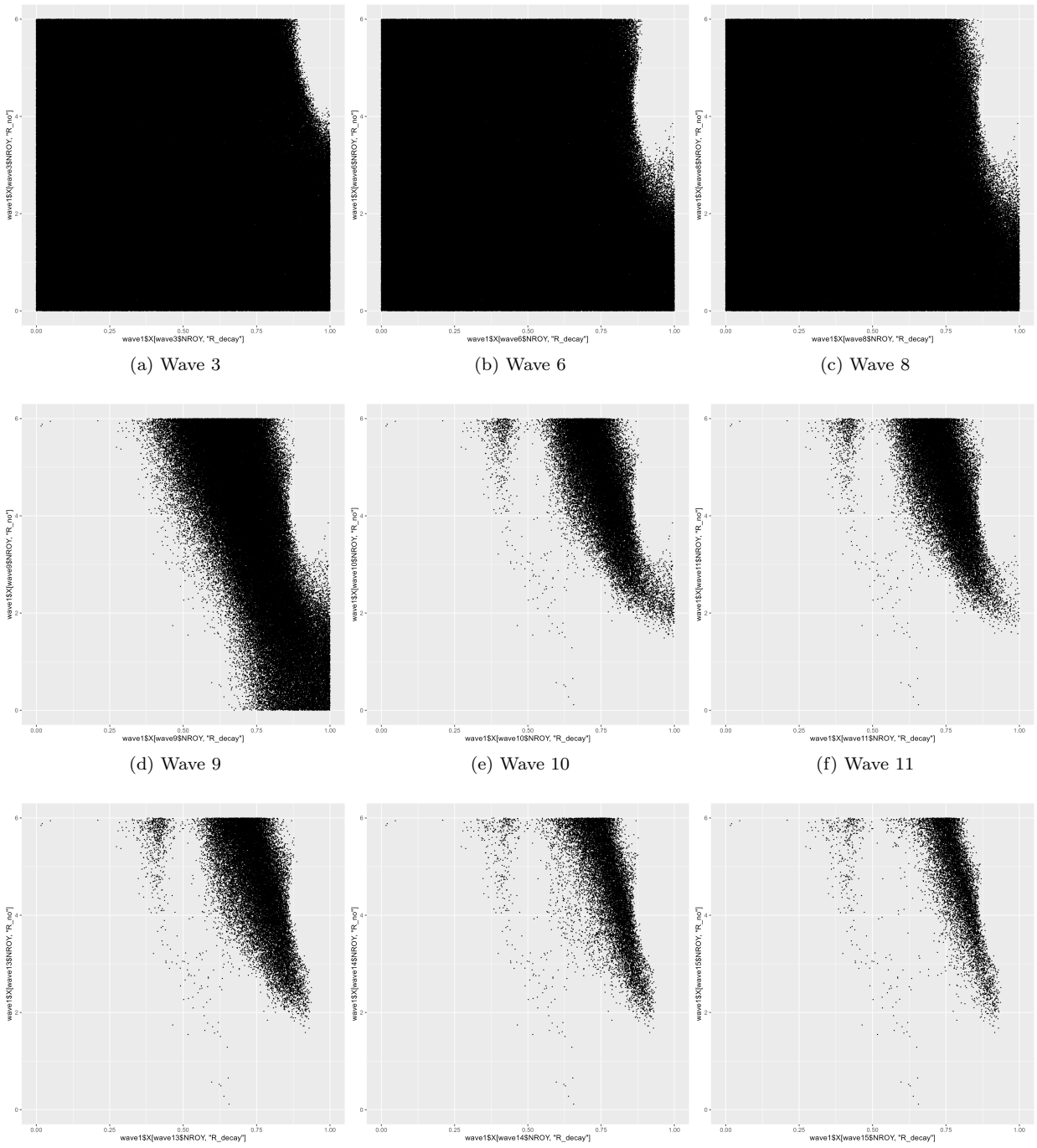
3.2.2 History matching

Before conducting the history matching, the model discrepancy and observation error need to be determined. Neither of these are available so the sum of these are calculated using eq.(1) using a historic simulation and observation and solving for $Var[\mathbf{e}] + Var[\epsilon]$. In regards to the observation $\mathbf{z}$ we applied a LOESS smoothing filter (the mean error of which was added to the observation error) to enable a smoother output which aids emulation and projection onto a basis. The observations were found using the England COVID-19 dashboard[2], specifically the number of people dying within 28 days of a positive test in the London region between 01/06/22 and 25/07/22.
As explained in section 3.2.1, we don't exclude any design points.
After conducting PCA and rotating the bases to ensure explainability of the data and a small reconstruction error of the observation $\mathbf{z}$, we now begin the history matching process. In between each wave, after sampling from the model (step 2 of the algorithm in section 1.1.2) we conduct PCA and the rotation of bases each time. This step is taken due to the high variability of the output and it may not be wise to only estimate weights from the existing basis vectors that were formed in that high variability environment.
In figure 3 we see waves of history matching displayed on a two-dimensional window into the input space

[2]https://coronavirus.data.gov.uk/

Wave no.	Prop. space remaining	Wave no.	Prop. space remaining
1	0.967	**9**	0.137
2	0.927	**10**	0.036
3	0.890	**11**	0.036
4	0.873	**12**	0.030
5	0.862	**13**	0.030
6	0.840	**14**	0.016
7	0.828	**15**	0.010
8	0.815		

Table 2: This table shows the proportion of input space remaining ($|\Omega_k(\mathbf{z})|/10^6$) after each wave of history matching for the WSS model.



(a) Wave 3 (b) Wave 6 (c) Wave 8

(d) Wave 9 (e) Wave 10 (f) Wave 11

(g) Wave 13 (h) Wave 14 (i) Wave 15

Figure 3: These plots represent a two dimensional window into the input space via two variables R_{decay} (on horizontal axis) and R_{no} (on vertical axis). Each plot represents a different wave with black dots indicating the regions of discritised input space that are still feasible given the output $\mathbf{z}$.

via two variables R_{decay} and R_{no}. These variables have been found to have the most impact on the outputs given that their input space has been constrained the most. It took many waves to start removing space (see table 2) as the history matching process had to deal with the high variability areas first (despite the cap). We conclude these areas of high variability took 8 waves to remove given how little space was removed

during that time. After the 9th wave however, we see the variability has decreased to the order of $O(10)$ necessary to start removing vast regions of input space which evaluate to those levels of output. After 9 waves we also see a linear relationship between R_{no} and R_{decay} in the remaining space. R_{decay} is the rate at which the R_{no} reduces down to 1 whereby each day the R_{no} is being calculated by

$$R_{no}[day + 1] = (R_{no}[day] - 1)^* R_{decay} + 1.$$

Through the use of history matching we have ruled out approximately 99% of input space which if one wanted to perform a calibration, they would have an instant set of indicator functions of whether a point in the input space can plausibly give the observation as an output.

4 Model 2: Micromob model

4.1 Model details

The MicroMob model[3] models the spread of infection amongst humans and mosquitoes. It predicts the number of infections over humans and mosquitoes over 3 compartments. For humans these are susceptible (S), infectious (I) and protected by drug treatment (P); for mosquitoes these are total mosquito density (M), infected mosquito density (Y) and infectious mosquito density (Z). The model predicts over 4 spatial patches with interactions between the patches. This model has a stochastic and deterministic mode, this report is only exploring the deterministic mode.

4.1.1 Inputs

After discussion with the model developer we are using 13 inputs which are displayed in table 3 alongside suitable ranges.

4.1.2 Outputs

The output for this model is a time series of 1095 days over 4 spatial patches with the 3 compartments for both humans and mosquitoes. This altogether gives a total of 26280 outputs. However we cut this down by only taking data at 28 day intervals and by ignoring the susceptible humans and the total mosquito density. This leaves a time series of 39 timesteps over 4 spatial patches and 2 compartments for humans and mosquitoes; overall giving 624 outputs.

4.2 Application of framework: emulation and history matching

As in section 3 we use the weighted singular value decomposition (the weight matrix being obtained from the sum of the observation error and model discrepancy) to conduct principal component analysis. Looking at figure 4 we see the vast majority of the variance is explained in the first 5 basis vectors therefore we choose the number of principal components to be emulated to be $q' = 5$.
To compensate for the large output, we are using 20 design points per input dimension (260 total), double the amount we are using for the WSS model. Once again we are able to emulate each of those 5 (now independent) outputs separately then combine them using eq.(7).
Looking at the plots in figure 5 we have mixed success in regards to predictive ability. In spatial patches $1-3$ we see that in general the model output (in red) is contained within the 200 emulator draws and yet in the plots for spatial patch 4 this is not maintained and represents an extremely poor fit. Prior to emulation we use principal component analysis to reduce the dimensionality of the output space which includes all compartments, species and spatial patches. These outputs can vary vastly from order $O(10)$ to $O(10^5)$. Perhaps treating these separately or only grouping appropriate outputs (and not all of the outputs) together would yield a better validation curve in all instances.
Whilst the validation curve is within the range of emulator draws, the shape of the line is not captured either. This further strengthens the idea that attempting to construct basis functions across all outputs would not

[3] github.com/slwu89/bioko_island$_travel_materials/blob/master/MicroMoB_config/emulation/bioko_sim_micromob_regions.R$

Input	Desc	Range
pfpr1	Prevalence of malaria in humans in spatial patch 1	[0.05,0.18]
pfpr2	Prevalence of malaria in humans in spatial patch 2	[0.02,0.17]
pfpr3	Prevalence of malaria in humans in spatial patch 3	[0.03,0.23]
pfpr4	Prevalence of malaria in humans in spatial patch 4	[0.05,0.21]
f	Mosquito feeding rate	[0.25,0.5]
q	Proportion of bites by mosquitos on humans	[0.5,1]
p	Probability of daily mosquito survival	[0.8,0.95]
eip	Incubation period in mosquitoes	[10,12]
b	Probability bite from infectious mosquito infects susceptible human	[0.45,0.65]
c	Probability bite from susceptible mosquito on infectious human infects the mosquito	[0.1,0.2]
r	Rate of recovery from infection for humans	[0.004,0.007]
eta	Rate at which drug protection decays for humans	[0.025,0.04]
rho	Proportion of new infections in human population which are successfully treated	[0.01,0.2]

Table 3: This table shows the inputs, their descriptions and ranges for the MicroMob model.

be a method for this style of output.
With regards to history matching it is a very similar story (so much so that the graphs look very similar to those in figure 5). The variability has reduced slightly given that 99.5% of input space has been ruled out (after 3 waves) however the poor fit of spatial patch 4 remains in those ranges of inputs.

5 Conclusions and future work

To conclude, we have developed a framework for conducting facets of uncertainty quantification applied to epidemiological models. We have introduced Gaussian process emulation, history matching, principal component analysis and rotation of bases and constructed a methodology on how to best combine them for emulating models with large outputs.
We have seen the success behind applying this framework to the WSS model in section 3, where we explored how to appropriately choose a number of basis functions to best emulate a time series and shown the predictive power through the use of a validation run. We also successfully ruled out over 99% of input space when history matching where we saw how certain relationships between the inputs started to develop (mainly between R_{decay} and R_{no}) and which inputs were of most importance to the output given how constrained they were as more waves were conducted. Where our methodology appeared to falter however was when modelling a much more complex output which had different species, spatial locations and compartments each of which had a time series also. Although part of the output was validated reasonably, spatial patch 4 had a poor fit.
In regards to future avenues of exploration in the fields covered in this report we recommend the following:

1. The choice of location of new design points in each wave of history matching: throughout all the waves of history matching conducted in this report, the choice of new design points in the non-implausible

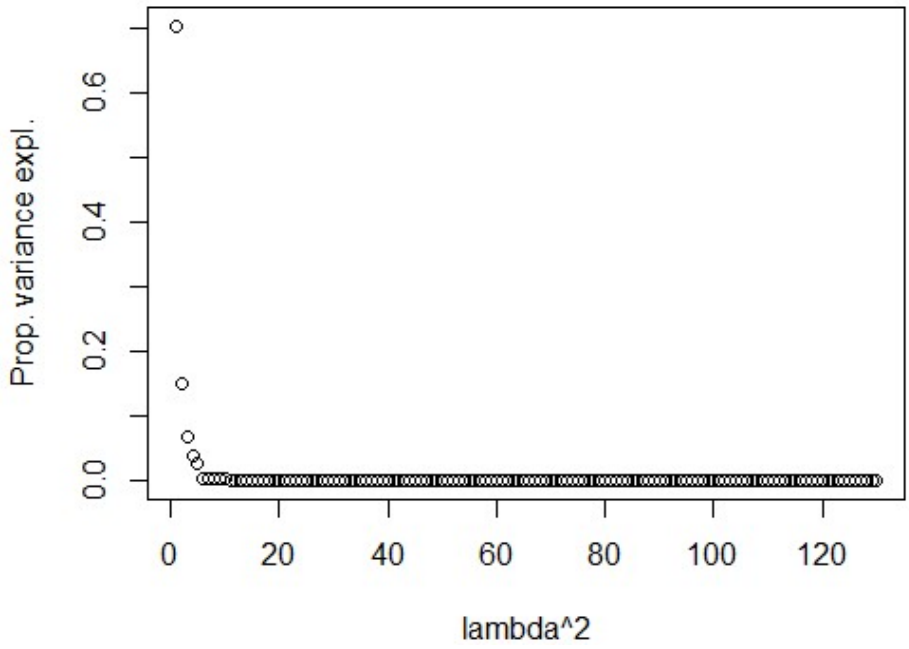


Figure 4: This plot shows the proportion of variance explained by each basis vector calculated using eq.(4) after undergoing principal component analysis via a weighted singular value decomposition.

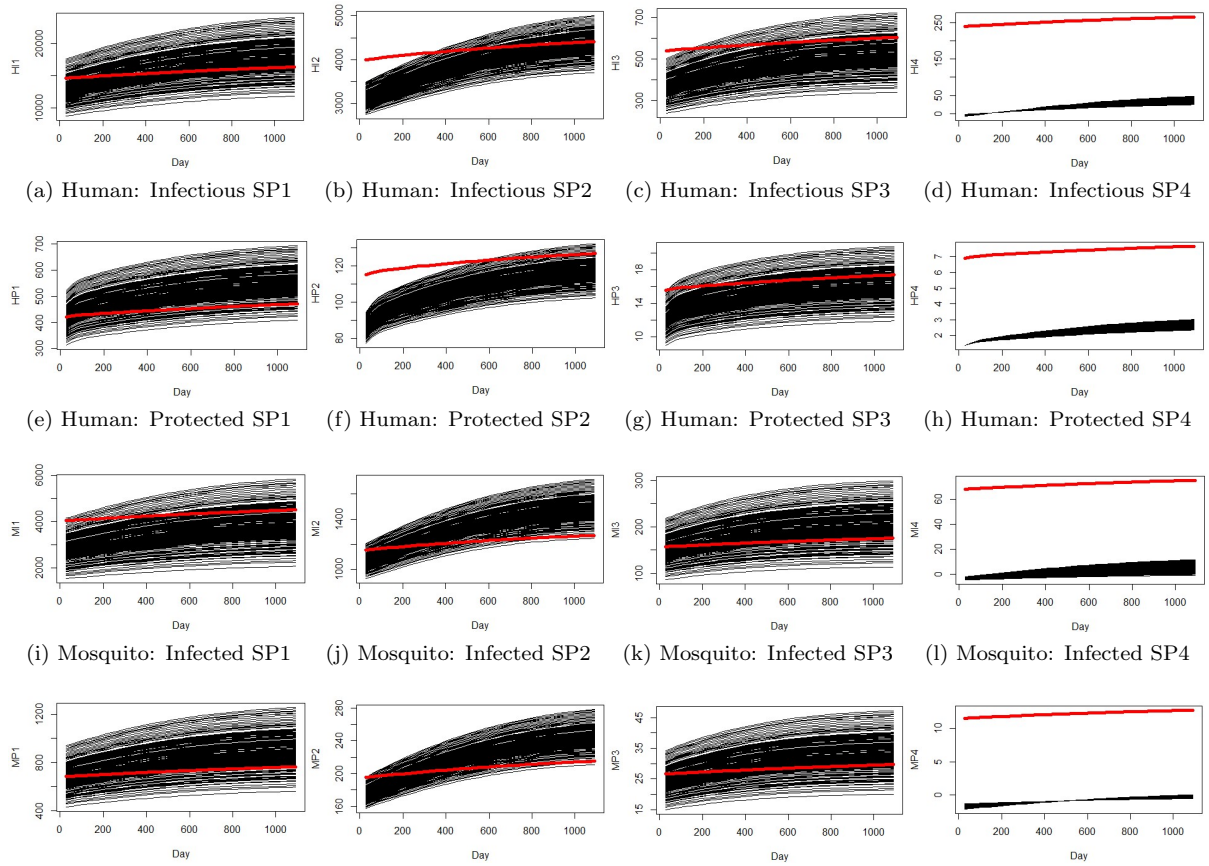
space $\Omega_k(\mathbf{z})$ was randomised from the discritised space. This could have played a part in the number of waves taken to rule out a small (but high variable) part of input space in section 3. The R package *hmer* (Iskauskas 2022) can sample close to the boundary between the non-implausible and implausible regions. Another method has been used by Williamson & Vernon (2013) to uniformly sample in non-implausible space defined by a membership function. One more simpler method used in practice is sampling using a space filling design over the original input space and stopping when the quota of the number of design points has been met that are inside the non-implausible region.

2. We can project into the future by only using a proportion of the full observation when history matching. If in section 3 we only used the first 15 days of the observation when history matching, we can identify the input space corresponding to that time period and then extrapolate outwards from $16 - 55$ days to make a projection into the future.

3. Stochastic models: in this report we only focused on deterministic models however many simulators are stochastic and require a slight modification to the emulation and history matching processes. By emulating the mean and variance (see Dunne et al. (2022)), we can capture the sample mean and variance over the repeated runs. We represent the stochasticity by including an extra term in the denominator of the implausibility measure in eq.(2).

4. Have a non-constant prior mean for the emulators. Each emulator has been build with a constant prior mean function (see sections 1.1.1 and 6.1 for more details on prior mean functions) however through discussion with model developers we can find the most influential variables and give them more weight in the prior. This could mean the emulator has less interpolation to do, thus reducing the uncertainty which is beneficial for emulation and history matching.

5. Look into more detail on how to emulate more complex outputs (as seen in section 4) where there is a spatio-temporal field rather than pursuing a general approach.



(a) Human: Infectious SP1 (b) Human: Infectious SP2 (c) Human: Infectious SP3 (d) Human: Infectious SP4

(e) Human: Protected SP1 (f) Human: Protected SP2 (g) Human: Protected SP3 (h) Human: Protected SP4

(i) Mosquito: Infected SP1 (j) Mosquito: Infected SP2 (k) Mosquito: Infected SP3 (l) Mosquito: Infected SP4

(m) Mosquito: Infectious SP1 (n) Mosquito: Infectious SP2 (o) Mosquito: Infectious SP3 (p) Mosquito: Infectious SP4

Figure 5: These figures show 200 draws of the predicted output of an input using the emulators compared to the model output (in red). This is for all emulated outputs: the 2 compartments for both species across all spatial patches (SP).

Acknowledgements

The authors would like to acknowledge the following personnel for their contributions to this report:

- James Salter who for his PhD and his paper Salter et al. (2019) provided much of the background code for this project. Thank you also for the proof reading of the sections concerning principal component analysis, singular value decomposition and rotation of bases.
- Graeme Ackland (the developer of the WSS model) for his time taken to split the model in two for the purposes of emulation. Also for our long chat over coffee on a hot Friday afternoon where we discussed the model and the input ranges.
- Sean Wu (part of the development team for the MicroMob model) for this time taken to meet with us and discuss the model and how best to emulate it.
- Kenji Takeda for chairing the 4-weekly sessions and providing the environment for us to present our work.
- The attendees of those 4-weekly meetings who have not yet been mentioned by name for their contributions and insightful questions.

6 Appendices

6.1 Uncertainty quantification: Gaussian process emulator formulae

The posterior mean function $E[\hat{f}]$ is defined by eq.(8) and the posterior covariance function defined by eq.(9). Note that $cov^*(\mathbf{x}, \mathbf{x}) = Var[\hat{f}(\mathbf{x})]$.

$$E[\hat{f}(\mathbf{x})] = \mathbf{h}(\mathbf{x})^T\beta + \mathbf{t}(\mathbf{x})^T A^{-1}(\mathbf{d} - H\beta); \tag{8}$$

$$cov^*(f(\mathbf{x}), f(\mathbf{x}')) = \sigma^2 c^*(\mathbf{x}, \mathbf{x}') \tag{9}$$

where:

$$\begin{aligned}
c^*(\mathbf{x}, \mathbf{x}') =& c(\mathbf{x}, \mathbf{x}') - \mathbf{t}(\mathbf{x})^T A^{-1}\mathbf{t}(\mathbf{x}') \\
&+ (\mathbf{h}(\mathbf{x})^T - \mathbf{t}(\mathbf{x})^T A^{-1}H)(H^T A^{-1}H))^{-1}(\mathbf{h}(\mathbf{x}')^T - \mathbf{t}(\mathbf{x}')^T A^{-1}H)^T; \\
\sigma^2 =& \frac{\mathbf{d}^T A^{-1}\mathbf{d} - \beta^T H^T A^{-1} H\beta}{n-2}; \\
\beta =& (H^T A^{-1}H)^{-1}(H^T A^{-1}\mathbf{d}); \\
t_i(\mathbf{x}) =& c(\mathbf{x}, \mathbf{x}_i) \text{ for } i \in \{1, \ldots, n\}; \\
H_{i,j} =& h_j(\mathbf{x}_i) \text{ for } i \in \{1, \ldots, n\},\ j \in \{1, \ldots, q\}; \\
A_{i,j} =& c(\mathbf{x}_i, \mathbf{x}_j) \text{ for } i, j \in \{1, \ldots, n\}; \\
d_i =& f(\mathbf{x}_i) \text{ for } i \in \{1, \ldots, n\}.
\end{aligned}$$

In this report we choose the prior covariance function to be a form of the family of Matern covariance functions, specifically the 'Matern 5/2':

$$c_{\nu=5/2}(\mathbf{x}, \mathbf{x}') = \prod_{i=1}^{n} \left(1 + \frac{\sqrt{5}\ |\mathbf{x}_i - \mathbf{x}'_i|}{l_i} + \frac{5\ |\mathbf{x}_i - \mathbf{x}'_i|^2}{3l_i^2}\right) \exp\left(-\frac{\sqrt{5}\ |\mathbf{x}_i - \mathbf{x}'_i|}{l_i}\right)$$

where l_i is a hyperparameter defining the lengthscale of each variable. This kernel was chosen as the underlying nature of the space in the model isn't known however if there is a known smooth structure then a squared exponential

$$k_{SE}(\mathbf{x}, \mathbf{x}') = \prod_{i=1}^{n} \exp\left(-\frac{|\mathbf{x}_i - \mathbf{x}'_i|^2}{2l_i^2}\right)$$

can be chosen instead (Rasmussen & Williams 2008). For full comparisons between covariance functions see chapter 4 of Rasmussen & Williams (2008).
The hyperparameters are determined through the maximisation of a log-likelihood function

$$log\ p(\mathbf{y}|X, \theta) = -\frac{1}{2}\left(d^T K^{-1} d + log|K| + n\ log\ 2\pi\right), \tag{10}$$

where $K = A + \sigma^2 I_n$ with I being the identity matrix of size n and θ being the set of all hyperparameters l_i. One can see the full derivation of the log-likelihood function in chapter 2.2 of Rasmussen & Williams (2008).

6.2 Summary of algorithm for rotating a basis

The algorithm for rotating a basis is aiming to ensure a high level of explainability for an ensemble Y and being able to reconstruct observation $\mathbf{z}$ for a truncated basis.
Prior to specifying the algorithm, Salter et al. (2019) define terms:

1. Reconstruction error given a basis $\mathbf{B}$ and vector $\mathbf{z}$ given the uncertainty $\mathbf{W}$ on $\mathbf{z}$:

   $$R_{\mathbf{W}}(\mathbf{B}, \mathbf{z}) = ||\mathbf{z} - \mathbf{B}(\mathbf{B^T W^{-1} B})^{-1}\mathbf{B^T W^{-1} z})||_{\mathbf{W}}$$

   where $||\mathbf{v}||_{\mathbf{W}} = \mathbf{v^T W^{-1} v}$ and $\mathbf{W}$ is a positive-definite weight matrix. Given that $\mathbf{W}$ is the uncertainty on $\mathbf{z}$, therefore $\mathbf{W}$ should be chosen as $\mathbf{W} = \mathbf{\Sigma}_\epsilon + \mathbf{\Sigma_e}$ which represent the variance of the model discrepancy and observation error respectively.

2. The explanation of variance of the centred ensemble $\mathbf{F}_\mu$ by a basis $\mathbf{B}$:

$$\nu(\mathbf{B}, \mathbf{F}_\mu) = \sum_{j=1}^{n} \frac{||\mathbf{B}(\mathbf{B^T W^{-1} B})^{-1}\mathbf{B^T W^{-1}}{\mathbf{F}_\mu}^j||_{\mathbf{W}}}{||{\mathbf{F}_\mu}^j||_{\mathbf{W}}},$$

where ${\mathbf{F}_\mu}^j$ is the jth row of an $n \times q$ matrix which represents the ensemble $\mathbf{F}$ centred by mean μ. Also the explanation of variance of the centred ensemble $\mathbf{F}_\mu$ by a singular basis vector $\mathbf{b_k}$:

$$\nu_k(\mathbf{B}, \mathbf{F}_\mu) = \sum_{j=1}^{n} \frac{||\mathbf{b_k}(\mathbf{b_k^T W^{-1} b_k})^{-1}\mathbf{b_k^T W^{-1}}{\mathbf{F}_\mu}^j||_{\mathbf{W}}}{||{\mathbf{F}_\mu}^j||_{\mathbf{W}}}.$$

Once it is shown that the observation $\mathbf{z}$ can be reconstructed from the **original** basis V^T (meaning at the very least, a terminal case doesn't occur) the algorithm in each iteration finds a linear combination of the basis that minimises the reconstruction error and satisfies the constraint of explaining the ensemble variability. After this step, if the reconstruction error of the truncated basis is smaller than the threshold $T = \chi^2_{q',0.995}$ (where q' is the number of truncated basis vectors) then stop as those basis vectors are sufficient to history match to $\mathbf{z}$ and the corresponding weights are emulatable. See Salter et al. (2019) for an explicit description of this algorithm.

6.3 R code

Inside the "epi-UQ" repository[4] lie the *R* functions necessary to implement the methods described in section 2.

```
pca = function(Y,W = NULL)
```

The function *pca* implements principal component analysis for an ensemble Y via singular value decomposition as seen in section 1.2. The argument W represents the weights should the user want to implement principal component analysis via a weighted singular value decomposition (Jolliffe 2002). As an output it gives the basis vectors, the respective weights to reconstruct the ensemble Y and the total variance explained by each principal component *VarExpl* (see eq.(4)).

```
salter_rotation_basis = function(basis.vectors,Y,z,W,kmax=5,
                                 v = c(rep(0.1,5)),vtot = 0.95,MaxTime=60,
                                 no.change = FALSE)
```

The function *salter_rotation_basis* uses the algorithm highlighted in section 1.2.1 and developed in Salter et al. (2019). It takes the basis vectors computed in *pca* and rotates them using the aforementioned algorithm towards the observation z using W as an uncertainty on z. *kmax* is the maximum number of iterations of the algorithm taken; v is the minimum proportion of variance in the ensemble data Y explained by the corresponding rotated basis vector; *vtot* is the minimum proportion of ensemble variability to be explained by the truncated basis; *MaxTime* seconds is the maximum time one iteration can be performed and *no.change* means not changing the basis from the one generated by function *pca* (the *hmwave* function only accepts basis vectors from the *salter_rotation_basis* function). This function produces the rotated basis, the reconstruction error of z and the amount of ensemble variability explained by each vector from the rotated basis.

```
emulation_weights = function(X,Y, setseed = 1,
                             HM = FALSE, prev.wave = NULL, size_NROY = 10^6,
                             var.names=var.names, upper.range=upper.range,
                             lower.range=lower.range)
```

The function *emulation_weights* is used to build q' Gaussian process emulators corresponding to the input data X of size $n \times p$ and the truncated output data Y of size $n \times q'$. *set.seed* is used for reproducibility of the building of emulators given that when optimising for hyperparameters (see eq.(10)) the starting point is randomised. The remainder of the arguments into the function concern the process of history matching. By setting *HM* to *TRUE*, one can evaluate the emulators at each point in either the whole discritised space (made up of *size_NROY* points) or (if *prev.wave* is specified) the space remaining from a previous wave.

[4] https://github.com/m-d-1882/epi-UQ/

```
predict_outputs = function(pred.inputs, emulators, full.output = FALSE,
                           basis.vectors = NULL)
```

The function *predict_outputs* takes an ensemble of inputs *predict.inputs* and evaluates them through the list of emulators in *emulators*. The *emulators* argument can be specified from *emulation_weights$emulators* or from *hmwave$emulation$emulators*. If *full.output* is FALSE then just the weights are returned, if TRUE then after specifying a basis through the argument *basis.vectors* the full output is reconstructed. The output is the mean and standard deviation of the weights or full output.

```
prediction_curves = function(pred.input, emulators, basis.vectors,
                             no.draws)
```

The function *prediction_curves* takes a singular input *pred.input* and through the same method as *predict_outputs* produces a mean and standard deviation using the *emulators* and *basis.vectors* arguments. The final step is using that mean and standard deviation to produce *no.draws* draws of that distribution so one can produce a family of output curves.

```
hmwave = function(wave.no, X, Y, new.pca = FALSE, no.pcs = NULL, pca = NULL,
z, obs.err, model.disc, var.names = var.names, upper.range = upper.range,
lower.range = lower.range, prev.wave = NULL, size_NROY = 10^6, setseed = 1,
no.new.samples = 10x length(var.names))
```

The function *hmwave* conducts one wave of history matching. It takes the design points to be used for that wave (X and Y), and either using a pre-specified basis (via *pca* and choosing number of weights via *no.pcs*) or a new basis can be generated within the function (by setting *new.pca* to *TRUE*) and being prompted to choose number of weights. It then history matches towards $\mathbf{z}$ with the uncertainty on $\mathbf{z}$ being variances of the observation error and model discrepancy *obs.err* and *model.disc* respectively. It uses the implausibility measure (eq.(2)) to rule out a portion of the remaining points in the discritised input space specified either by *prev.wave* or (if *prev.wave = NULL*) then it is assumed *wave.no* $= 1$ and thus the whole input space is discritised into *size_NROY* points. After the history matching process is complete, one can specify the number of new samples *no.new.samples* one can evaluate using the simulator for the next wave. This function outputs the basis vectors used to represent the full output space, the emulators, the discritised space and whether they are still in the implausible space, and the new samples for evaluation for the next history matching wave.

6.4 Example: Principal component analysis via singular value decomposition

This example was conducted by Strang (2005)[5].
Take a 2×2 matrix

$$A = \begin{bmatrix} 4 & 4 \\ -3 & 3 \end{bmatrix}$$

which we want to decompose into the form

$$A = U\Sigma V^T \tag{11}$$

where U and V are orthogonal unitary matrices and Σ is a diagonal rectangular matrix. By calculating $A^T A$ we obtain

$$A^T A = \left(U\Sigma V^T\right)^T \left(U\Sigma V^T\right) = V\Sigma^T U^T U\Sigma V^T = V\Sigma^* V^T$$

where Σ^* is a diagonal square matrix with the same size as V with all terms in the diagonal the square of those diagonal terms in Σ. Note that for an orthogonal matrix: $M^T = M^{-1}$. Therefore by finding the eigenvalues and eigenvectors of $A^T A$ we can obtain Σ^* and V. Through substitution back into eq.(11) we can solve for U.
Applying this to our example matrix A, $A^T A$ gives us

$$A^T A = \begin{bmatrix} 25 & 7 \\ 7 & 25 \end{bmatrix}$$

[5] https://www.youtube.com/watch?v=TX_vooSnhm8

giving us eigenvalues $\lambda_{1,2} = 32, 18$ and eigenvectors $v_1 = (1/\sqrt{2},\ 1/\sqrt{2})$ and $v_2 = (1/\sqrt{2},\ -1/\sqrt{2})$. Through substitution into eq.(11) we obtain

$$U = \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix}$$

leading to the full SVD of

$$A = \begin{bmatrix} 4 & 4 \\ -3 & 3 \end{bmatrix} = U\Sigma V^T = \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix} \begin{bmatrix} 4\sqrt{2} & 0 \\ 0 & 3\sqrt{2} \end{bmatrix} \begin{bmatrix} 1/\sqrt{2} & 1/\sqrt{2} \\ 1/\sqrt{2} & -1/\sqrt{2} \end{bmatrix}.$$

In terms of application to principal component analysis; to reduce the dimensionality of A one must centre the data first (i.e. take away from each column the average of that column) and conduct the above process. The basis vectors are represented by V^T and the corresponding weights by $U\Sigma$. One can select the proportion of variance to be represented by applying eq.(4) which yields $P = (0.5714, 1.0000)$ and truncate. One truncates by choosing q such that the basis vectors become the first q **columns** of V^T (a.k.a. V_q^T) and the weights become the first q **rows** of $U\Sigma$ (a.k.a. $(U\Sigma)_q$).

References

Bower, R. G., Goldstein, M. & Vernon, I. (2010), 'Galaxy formation: a bayesian uncertainty analysis', *Bayesian Analysis* **5**(4), 619–669.
URL: *https://projecteuclid.org/journals/bayesian-analysis/volume-5/issue-4/Galaxy-formation-a-Bayesian-uncertainty-analysis/10.1214/10-BA524.full?*

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

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 High-Dimensional Output', *American Statistical Association* **103**(482), 570–583.

Iskauskas, A. (2022), *hmer: History Matching and Emulation Package*. R package version 1.2.0.
URL: *https://CRAN.R-project.org/package=hmer*

Jolliffe, I. T. (2002), *Principal Component Analysis*, Springer Series in Statistics, 2 edn, Springer.

Jolliffe, I. T., Cadima, J. & Cadima, J. (2016), 'Principal component analysis : a review and recent developments', *Philosophical Transactions of the Royal Society A* **374**(2065), 714:.

Lee, L. A., Carslaw, K. S., Pringle, K. J., Mann, G. W. & Spracklen, D. V. (2011), 'Emulation of a complex global aerosol model to quantify sensitivity to uncertain parameters', *Atmospheric Chemistry and Physics* **11**(23), 12253–12273.
URL: *https://acp.copernicus.org/articles/11/12253/2011/*

Loeppky, J. L., Sacks, J. & Welch, W. J. (2009), 'Choosing the sample size of a computer experiment: A practical guide', *Technometrics* **51**(4), 366–376.

Mckay, A. M. D., Beckman, R. J., Conover, W. J., Technometrics, S., May, N., Mckay, M. D. & Beckman, R. J. (1979), 'A Comparison of Three Methods for Selecting Values of Input Variables in the Analysis of Output from a Computer Code A Comparison of Three Methods for Selecting Values of Input Variables in the Analysis of Output from a Computer Code', *Technometrics* **21**(2), 239–245.

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.

R Core Team (2021), *R: A Language and Environment for Statistical Computing*, R Foundation for Statistical Computing, Vienna, Austria.
URL: *https://www.R-project.org/*

Rasmussen, C. E. & Williams, C. K. I. (2008), *Gaussian processes for machine learning*, MIT Press.

Rougier, J. (2012), *tensor: Tensor product of arrays*. R package version 1.5.
URL: *https://CRAN.R-project.org/package=tensor*

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/*

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*

Serge, D. J. G. (2015), *far: Modelization for Functional AutoRegressive Processes*. R package version 0.6-5.
URL: *https://CRAN.R-project.org/package=far*

Strang, G. (2005), '29. singular value decomposition'.

Swallow, B., Birrell, P., Blake, J., Burgman, M., Challenor, P., Coffeng, L. E., Dawid, P., De Angelis, D., Goldstein, M., Hemming, V., Marion, G., McKinley, T. J., Overton, C. E., Panovska-Griffiths, J., Pellis, L., Probert, W., Shea, K., Villela, D. & Vernon, I. (2022), 'Challenges in estimation, uncertainty quantification and elicitation for pandemic modelling', *Epidemics* **38**, 100547.
URL: *https://doi.org/10.1016/j.epidem.2022.100547*

Vernon, I., Goldstein, M. & Bower, R. (2014), 'Galaxy formation: Bayesian history matching for the observable universe', *Statistical Science* **29**(1), 81–90.

Vernon, I., Liu, J., Goldstein, M., Rowe, J., Topping, J. & Lindsey, K. (2018), 'Bayesian uncertainty analysis for complex systems biology models: Emulation, global parameter searches and evaluation of gene functions', *BMC Systems Biology* **12**(1), 1–29.

Wickham, H., Averick, M., Bryan, J., Chang, W., McGowan, L. D., François, R., Grolemund, G., Hayes, A., Henry, L., Hester, J., Kuhn, M., Pedersen, T. L., Miller, E., Bache, S. M., Müller, K., Ooms, J., Robinson, D., Seidel, D. P., Spinu, V., Takahashi, K., Vaughan, D., Wilke, C., Woo, K. & Yutani, H. (2019), 'Welcome to the tidyverse', *Journal of Open Source Software* **4**(43), 1686.

Williamson, D. & Vernon, I. (2013), 'Efficient uniform designs for multi-wave computer experiments', pp. 1–31.
URL: *http://arxiv.org/abs/1309.3520*

Yang Xiang, Gubian, S., Suomela, B. & Hoeng, J. (2013), 'Generalized simulated annealing for efficient global optimization: the GenSA package for R.', *The R Journal Volume 5/1, June 2013* .
URL: *https://journal.r-project.org/archive/2013/RJ-2013-002/index.html*