Weaving the digital tapestry: emulating cohorts of digital twins using Gaussian processes

Christopher Lanyon[1], Cristobal Rodero[2], Abdul Qayyum[2], Steven A. Niederer[2,3], Richard D. Wilkinson[1]

[1]School of Mathematical Sciences, University of Nottingham
[2]Cardiac Electro-Mechanics Research Group (CEMRG), National Heart and Lung Institute, Faculty of Medicine, Imperial College London, London, United Kingdom
[3]Turing Research and Innovation Cluster in Digital Twins (TRIC: DT), The Alan Turing Institute, London, United Kingdom

GPSS 2024

Project team

Cristobal

Abdul

Steven

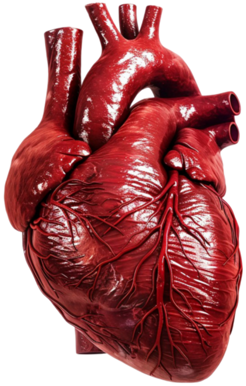
Richard

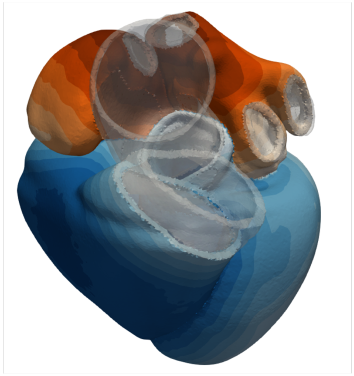
Outline

- Introduction to Digital twins and computer model emulation
- Motivating example: Cohort of Cardiac Digital Twins
- Cohort Learning methods

Digital Twins

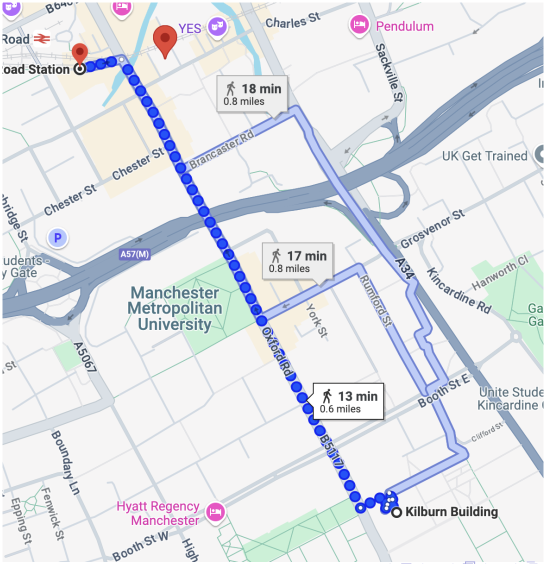
Digital twins are in-silico representations of real world objects, usually encoded into mathematical or computer models (a simulator), connected by data.





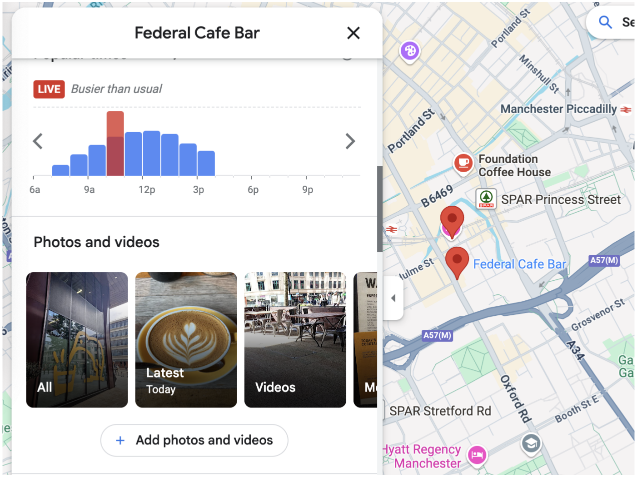
The most frequently used digital twin?

Google maps is, arguably, a digital twin of the surface of the earth, with one-billion active monthly users.



The most frequently used digital twin?

Data is collected by Google, alongside crowdsourced data, allowing Google maps to model the world in near real-time.



Digital Twins: examples

- Personalised cardiac models: meshes generated via imaging, measurements used to inform parameter values, model outputs used to guide treatment
- Models of machinery in industrial settings: used to optimise maintenance schedules.
- Digital twins of energy grids: used to predict surges, and optimise energy distribution to cities

Computer model emulation

To capture an appropriate amount of detail from the real world, digital twins often require very complicated or computationally expensive mathematical models. In these cases it can be helpful to use emulators or surrogate models to speed up processing.
Possible surrogate models or emulators:

- Simplified mathematical model (e.g. linearisation, symbolic regression)
- Statistical model (e.g. regression, clustering)
- Machine learning models (e.g. Neural networks, Gaussian processes)

Computer model emulation

When might surrogate models be helpful?

- Fast paced real-world settings:
    - Clinical recomendations
    - Structural fault identification
- When many simulations are required:
    - Sensitivity analysis
    - Calibration
    - Parameter optimisation

Cohorts of DTs: Weaving the digital tapestry

- In some cases, one might have a cohort of digital twins, with personalised models for each cohort member (e.g. vehicles, patients, factories). Digital twin cohorts can be used for outlier detection, scheduling and in-silico trials.

Cohorts of DTs: Weaving the digital tapestry

- In some cases, one might have a cohort of digital twins, with personalised models for each cohort member (e.g. vehicles, patients, factories). Digital twin cohorts can be used for outlier detection, scheduling and in-silico trials.
- As with individual digital twins, emulators and surrogate models can increase the efficiency and scalability of digital twin cohorts.

Cohorts of DTs: Weaving the digital tapestry

- In some cases, one might have a cohort of digital twins, with personalised models for each cohort member (e.g. vehicles, patients, factories). Digital twin cohorts can be used for outlier detection, scheduling and in-silico trials.
- As with individual digital twins, emulators and surrogate models can increase the efficiency and scalability of digital twin cohorts.
- Traditionally, emulators for each individual digital twin would be trained separately, requiring multiple runs of the computationally expensive simulator.

Cohorts of DTs: Weaving the digital tapestry

- In some cases, one might have a cohort of digital twins, with personalised models for each cohort member (e.g. vehicles, patients, factories). Digital twin cohorts can be used for outlier detection, scheduling and in-silico trials.
- As with individual digital twins, emulators and surrogate models can increase the efficiency and scalability of digital twin cohorts.
- Traditionally, emulators for each individual digital twin would be trained separately, requiring multiple runs of the computationally expensive simulator.
- If the underlying real world structures are sufficiently similar, we believe it should be possible for new emulators to learn from the existing emulated cohort simultaneously, reducing the required number of simulations significantly.

Cohorts of DTs: What makes a cohort of emulators?

- A group of N real world objects, $\{O_n\}_{n=1}^{N}$

Cohorts of DTs: What makes a cohort of emulators?

- A group of N real world objects, $\{O_n\}_{n=1}^N$
  - These objects differ from each other according to some set of latent variables $\{l_n\}_{n=1}^N$, where l_n describes some underlying quality of O_n.
  - Note: we may not have an explicit expression for these latent variables, but it might be possible to learn one

Cohorts of DTs: What makes a cohort of emulators?

- A group of N real world objects, $\{O_n\}_{n=1}^N$
  - These objects differ from each other according to some set of latent variables $\{l_n\}_{n=1}^N$, where l_n describes some underlying quality of O_n.
  - Note: we may not have an explicit expression for these latent variables, but it might be possible to learn one
- A digital representation or simulator, $f(\theta)$, such that a digital twin of object n is given by
$$f_n(\theta) = f(\theta, l_n)$$

Cohorts of DTs: What makes a cohort of emulators?

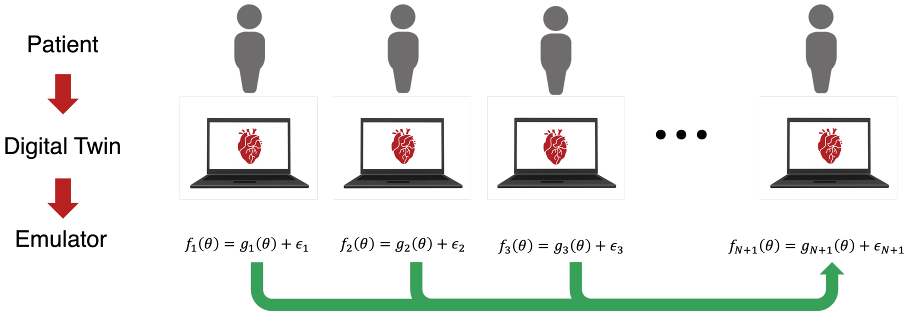
- A group of N real world objects, $\{O_n\}_{n=1}^{N}$
  - These objects differ from each other according to some set of latent variables $\{l_n\}_{n=1}^{N}$, where l_n describes some underlying quality of O_n.
  - Note: we may not have an explicit expression for these latent variables, but it might be possible to learn one
- A digital representation or simulator, $f(\theta)$, such that a digital twin of object n is given by
$$f_n(\theta) = f(\theta, l_n)$$
- An emulator for f, g, such that $f_n(\theta) = g_n(\theta) + \epsilon_n$.

Cohorts of DTs: What makes a cohort of emulators?

- A group of N real world objects, $\{O_n\}_{n=1}^{N}$
  - These objects differ from each other according to some set of latent variables $\{l_n\}_{n=1}^{N}$, where l_n describes some underlying quality of O_n.
  - Note: we may not have an explicit expression for these latent variables, but it might be possible to learn one
- A digital representation or simulator, $f(\theta)$, such that a digital twin of object n is given by
$$f_n(\theta) = f(\theta, l_n)$$
- An emulator for f, g, such that $f_n(\theta) = g_n(\theta) + \epsilon_n$.

If we add an $N+1$th member of the cohort, we want to be able to utilise our knowledge of the cohort to make emulating f_{N+1} faster and more computationally efficient.

Case study: A cohort of cardiac digital twins


Patient
Digital Twin
Emulator
$f_1(\theta) = g_1(\theta) + \epsilon_1$
$f_2(\theta) = g_2(\theta) + \epsilon_2$
$f_3(\theta) = g_3(\theta) + \epsilon_3$
$f_{N+1}(\theta) = g_{N+1}(\theta) + \epsilon_{N+1}$

Case study: A cohort of cardiac digital twins

- Cohort: 18 patient specific cardiac meshes generated using CT imaging
- Simulator: Reaction-eikonal model of electrophysiology
- Latent features: Outputs from a statistical shape model (each mesh described uniquely by an array of real numbers)

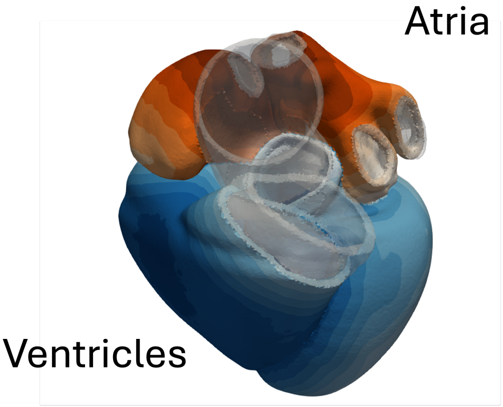
See: Rodero, Cristobal, et al. "Linking statistical shape models and simulated function in the healthy adult human heart." PLoS computational biology 17.4 (2021): e1008851.

Case study: Cohort

- CT scans of patient hearts were converted to personalised cardiac meshes using image segmentation
- This generates personalised cardiac meshes for each cohort member

Case study: Cohort

- CT scans of patient hearts were converted to personalised cardiac meshes using image segmentation
- This generates personalised cardiac meshes for each cohort member



Case study: Simulator

- Heartbeats are governed by the propagation of electrical signals across the heart. The reaction-eikonal model of electrophysiology models this electrical wave propagation.
- Our outputs from the model are the activation times across the heart's surface: how long it takes for the signal to travel from its origin to a given point on the heart.

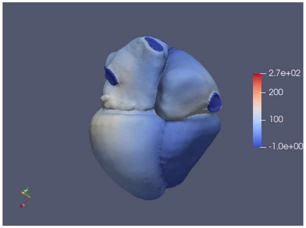
Case study: Simulator

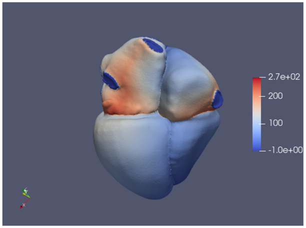
- Heartbeats are governed by the propagation of electrical signals across the heart. The reaction-eikonal model of electrophysiology models this electrical wave propagation.
- Our outputs from the model are the activation times across the heart's surface: how long it takes for the signal to travel from its origin to a given point on the heart.
- Chris! Remember to show the video here.

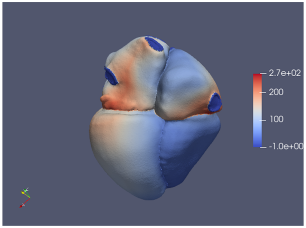
Case study: Simulator

Simulator outputs:

- Activation times across the entire heart, summarised as the total activation times over the atria and ventricles:
  - V_{TAT}: Ventricle total activation time
  - A_{TAT}: Atrial total activation time







Case study: Simulator

Parameters:

- $CV_{\mathrm{ventricles}}$: Ventricle conduction velocity
- CV_{atria}: Atrial conduction velocity
- $k_{\mathrm{ventricles}}$: Ventricle anisotropy ratio
- k_{atria}: Atria anisotropy ratio
- k_{BB}: Bachman bundle anisotropy ratio
- k_{FEC}: Fast endocardial conduction layer anisotropy ratio

Case study: Latent features

- Rodero et al. generated a statistical shape model (SSM) to describe the variance between meshes
- Each mesh was uniquely described by an array of real numbers (modes)
- Principal component analysis was used to determine which modes contributed most to variance
- The first 9 modes capture $\approx 90\%$ of the explained variance

Case study: Data summary

18 meshes, 180 simulations per mesh

6 input parameters

- $CV_{\mathrm{ventricles}}$: Ventricle conduction velocity
- CV_{atria}: Atrial conduction velocity
- $k_{\mathrm{ventricles}}$: Ventricle anisotropy ratio
- k_{atria}: Atria anisotropy ratio
- k_{BB}: Bachman bundle anisotropy ratio
- k_{FEC}: Fast endocardial conduction layer anisotropy ratio

2 outputs

- A_{TAT}
- V_{TAT}

9 latent features

- $\{Mode1, Mode2, ..., Mode9\}$

Gaussian process emulators

A gaussian process can be expressed via its mean, μ, and covariance function, k:

$$g(\cdot) = GP(\mu(\cdot), k(\cdot, \cdot)). \tag{1}$$

For our GPEs we used a linear mean function, such that

$$\mu = \beta_0 + \sum_{i=1}^{m} \beta_i \theta_i \tag{2}$$

where m is the dimension of the data points and chose the radial basis function (RBF) kernel for the covariance so that

$$k(\theta, \theta') = \delta_s \exp\left(-\frac{1}{2}(\theta - \theta')\Theta^{-2}(\theta - \theta')\right) \tag{3}$$

where δ_s is the output scale and Θ is the lengthscale.

Baseline: individual emulators

As a baseline for our analysis we generated individual emulators for each cohort member such that, for the n cohort member:

$$f_n(\theta) = g_n(\theta) + \epsilon_n$$

where g_n is a gaussian process trained using θ and $f_n(\theta)$.

Latent feature emulator

We have a cohort of digital twins, each with known latent features which differentiate the patients from each other:

$$\{f_n(\theta)\}_{n=1}^N = \{f(\theta, l_n)\}_{n=1}^N.$$

Rather than constructing a single emulator for each DT (i.e. $f_n(\theta) = g_n(\theta) + \epsilon_n$), one could instead construct a single large emulator for the entire cohort:

$$f(\theta, l) = g(\theta; l) + \epsilon$$

Here, the semi-colon notation, as in '$(\theta; l)$', refers to concatenating the input parameters and the latent features to train the GP.

Latent feature emulator

Proposed advantages:

- With large enough cohort sizes, may be able to predict for unseen members
- Only have to train a single emulator for the entire cohort

However

- Requires the latent features to be known (or learned)
- GPEs can become unwieldy for very large training sets

Discrepancy emulator

We propose that each member of the DT cohort differs from another by some weight, a, and some discrepancy term δ:

$$f_1(\theta) = af_0(\theta) + \delta$$

Discrepancy emulator

We propose that each member of the DT cohort differs from another by some weight, a, and some discrepancy term δ:

$$f_1(\theta) = af_0(\theta) + \delta$$

If $g_0(\theta)$ is an emulator of $f_0(\theta)$ then we propose a new emulator for $f_1(\theta)$:

$$g_1(\theta) = ag_0(\theta) + \delta(\theta)$$

where $\delta(\theta)$ is modelled using a gaussian process with mean m_δ and kernel k_δ, and a is some constant weight that can either be set by the user or learned statistically.

Discrepancy emulator

We propose that each member of the DT cohort differs from another by some weight, *a*, and some discrepancy term δ:

$$f_1(\theta) = af_0(\theta) + \delta$$

If $g_0(\theta)$ is an emulator of $f_0(\theta)$ then we propose a new emulator for $f_1(\theta)$:

$$g_1(\theta) = ag_0(\theta) + \delta(\theta)$$

where $\delta(\theta)$ is modelled using a gaussian process with mean m_δ and kernel k_δ, and *a* is some constant weight that can either be set by the user or learned statistically. NB: This method does not require an explicit representation of the latent features of each cohort member

Discrepancy emulator: Single reference

As g_0 and δ_i are Guassian processes we can employ the conditional property of gaussian processes:

$$ag_0(\theta)|\theta \sim N(am_0(\theta), a^2 k_0(\theta, \theta)$$

and

$$\delta(\theta)|\theta \sim N(m_\delta(\theta), k_\delta(\theta, \theta))$$

Discrepancy emulator: Single reference

As g_0 and δ_i are Guassian processes we can employ the conditional property of gaussian processes:

$$ag_0(\theta)|\theta \sim N(am_0(\theta), a^2 k_0(\theta, \theta)$$

and

$$\delta(\theta)|\theta \sim N(m_\delta(\theta), k_\delta(\theta, \theta))$$

and therefore

$$g_1(\theta)|\theta \sim N(am_0(\theta) + m_\delta(\theta), a^2 k_0(\theta, \theta) + k_\delta(\theta, \theta))$$

.

Thus g_1 is a Gaussian process with mean $m_1(\cdot) = am_0(\cdot) + m_\delta(\cdot)$ and covariance $k_i(\cdot, \cdot) = a^2 k_0(\cdot, \cdot) + k_\delta(\cdot, \cdot)$

Discrepancy emulator: Full cohort

We extend the reference emulator approach to account for the full emulator cohort, $\{g_n\}_{n=1}^N$, by using a weighted sum over the cohort as our reference.

$$g_{N+1}(\theta) = \sum_{n=1}^{N} a_n g_n(\theta) + \delta_c(\theta)$$

where δ_c is the cohort discrepancy. We model δ_c as a GP with mean μ_{δ_c} and kernel k_{δ_c}. As g_i is a sum of Gaussian processes, it is a gaussian process with mean

$$\sum_{n=1}^{N} a_n m_n(\cdot) + m_{\delta_c}(\cdot)$$

and covariance

$$\sum_{n=1}^{N} a_n^2 k_n(\cdot,\cdot) + k_{\delta_c}(\cdot,\cdot)$$

and as the hyper-parameters for m_n and k_n are known, we only need to learn the hyperparameters for m_{δ_c} and k_{δ_c}.

Discrepancy emulator: how should we learn $\{a_n\}_{a=1}^{N}$

User determined (if you have prior knowledge):

- Choose a single reference: setting $a_j \neq 0$ and $a_{i \neq j} = 0$ is equivalent to the single emulator case

Discrepancy emulator: how should we learn $\{a_n\}_{a=1}^{N}$

User determined (if you have prior knowledge):

- Choose a single reference: setting $a_j \neq 0$ and $a_{i\neq j} = 0$ is equivalent to the single emulator case
- Similarly you could choose a subset $\{a_m\}_{m\in M} \neq 0$ where $M \subset 1...N$ and $\{a_i\}_i \notin M = 0$

Discrepancy emulator: how should we learn $\{a_n\}_{a=1}^{N}$

User determined (if you have prior knowledge):

- Choose a single reference: setting $a_j \neq 0$ and $a_{i \neq j} = 0$ is equivalent to the single emulator case
- Similarly you could choose a subset $\{a_m\}_{m \in M} \neq 0$ where $M \subset 1...N$ and $\{a_i\}_i \notin M = 0$

Statistical methods:

Discrepancy emulator: how should we learn $\{a_n\}_{a=1}^{N}$

User determined (if you have prior knowledge):

- Choose a single reference: setting $a_j \neq 0$ and $a_{i \neq j} = 0$ is equivalent to the single emulator case
- Similarly you could choose a subset $\{a_m\}_{m \in M} \neq 0$ where $M \subset 1...N$ and $\{a_i\}_i \notin M = 0$

Statistical methods:

- Optimise $\{a_n\}_{n=1}^{N}$ during GPE training, i.e., include them as a GPE hyperparameter

Discrepancy emulator: how should we learn $\{a_n\}_{a=1}^{N}$

User determined (if you have prior knowledge):

- Choose a single reference: setting $a_j \neq 0$ and $a_{i \neq j} = 0$ is equivalent to the single emulator case
- Similarly you could choose a subset $\{a_m\}_{m \in M} \neq 0$ where $M \subset 1 \ldots N$ and $\{a_i\}_i \notin M = 0$

Statistical methods:

- Optimise $\{a_n\}_{n=1}^{N}$ during GPE training, i.e., include them as a GPE hyperparameter
- Use lasso regression to learn $\{a_m\}_{m \in M}$.

Discrepancy emulator: how should we learn $\{a_n\}_{a=1}^{N}$

User determined (if you have prior knowledge):

- Choose a single reference: setting $a_j \neq 0$ and $a_{i \neq j} = 0$ is equivalent to the single emulator case
- Similarly you could choose a subset $\{a_m\}_{m \in M} \neq 0$ where $M \subset 1 \ldots N$ and $\{a_i\}_i \notin M = 0$

Statistical methods:

- Optimise $\{a_n\}_{n=1}^{N}$ during GPE training, i.e., include them as a GPE hyperparameter
- Use lasso regression to learn $\{a_m\}_{m \in M}$.
- Use lasso regression as an indicator function and treat the non-zero weights as hyperparameters.

Model summary

Individual emulators

- Inputs: θ, $f_i(\theta)$
- Outputs: $g_i(\theta) = (g_{iA}(\theta), g_{iV}(\theta))$

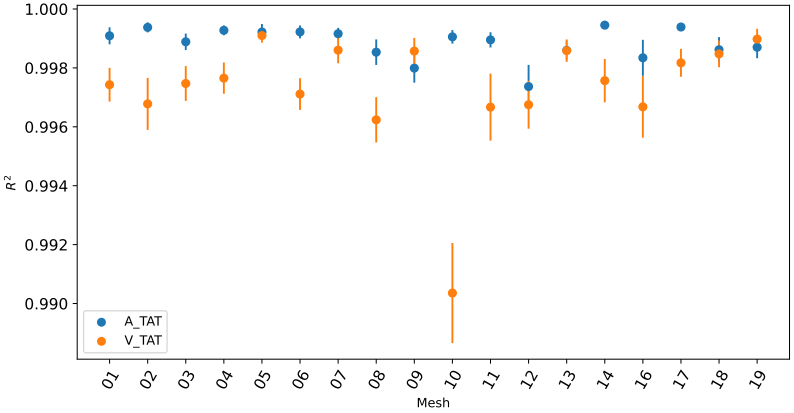
Latent emulators

- Inputs: $(\theta; l)$, $\{f_n(\theta)\}_{n=1}^{N}$
- Outputs: $g((\theta; l)) = (g_A((\theta; l)), g_V((\theta; l))$

Discrepancy emulators

- Inputs: θ, $f_i(\theta)$, $\{m_n\}$, $\{k_n\}$: means and covariances of reference emulators
- Outputs: $g_i(\theta) = (g_{iA}(\theta), g_{iV}(\theta))$

Results: individual emulators, 144 training points



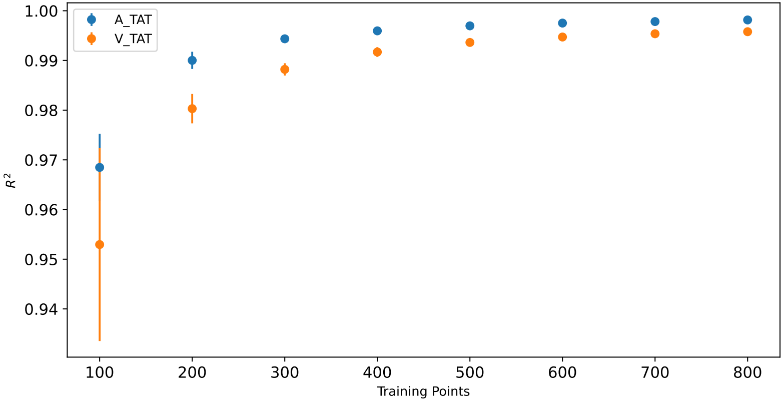
Results: Training set size for individual emulators

N	A_{TAT}	V_{TAT}
20	0.895	0.915
40	0.988	0.977
60	0.995	0.988
80	0.996	0.992
100	0.998	0.994
120	0.998	0.995
140	0.999	0.997

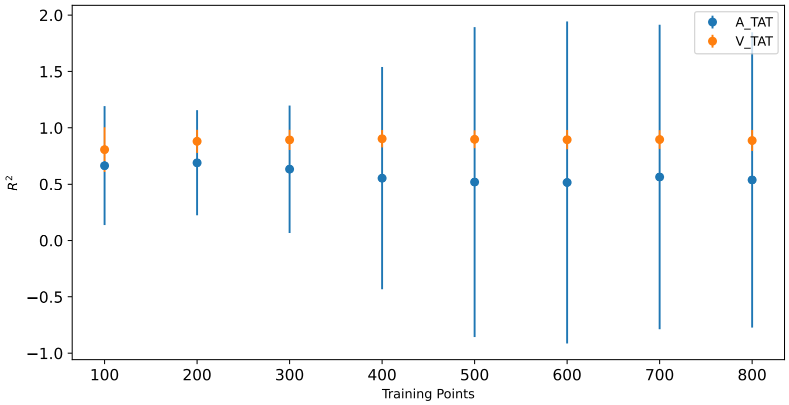
Results: Latent emulators

- Latent emulators were trained using 17 out of the 18 possible meshes then tested on test data from both the left in and left out meshes.
- Results shown here are the averages over the 18 different latent emulators (each one leaving out one mesh).

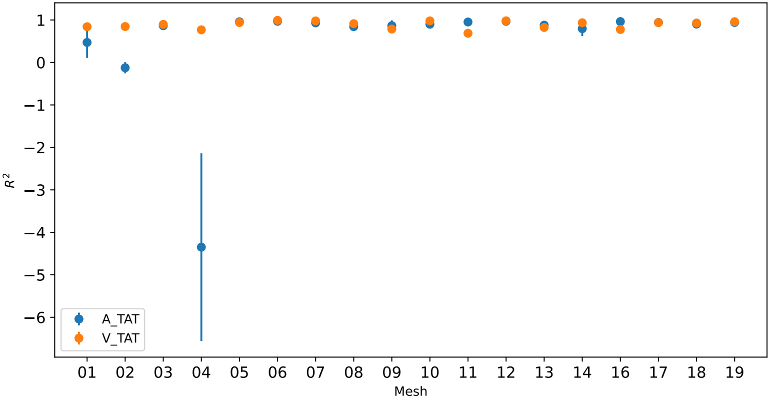
Results: Latent emulators, left in meshes



Results: Latent emulators, left out meshes



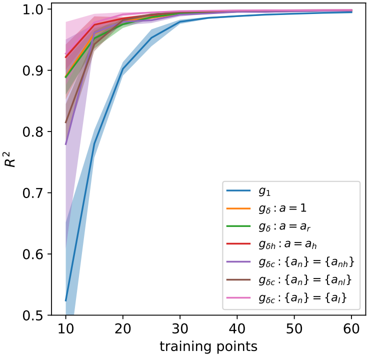
Results: Latent emulators, left out meshes

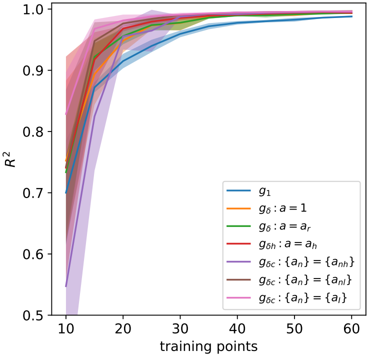


Results:Discrepancy emulator models

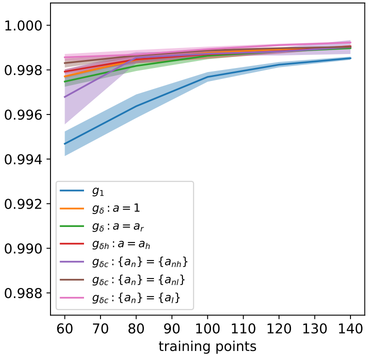
- g_1, individual emulator, used as baseline
- $g_\delta : a = 1$: One reference emulator and $a = 1$
- $g_\delta : a = a_r$: One reference emulator and a learned via regression
- $g_{\delta h} : a = a_h$: One reference emulator and a learned as a GPE hyperparameter
- $g_{\delta c} : \{a_n\} = \{a_{nh}\}$: Full cohort of reference emulators and $\{a_n\}$ learned as GPE hyperparameters
- $g_{\delta c} : \{a_n\} = \{a_{nl}\}$: Full cohort of reference emulators and $\{a_n\}$ learned using lasso regression
- $g_{\delta c} : \{a_n\} = \{a_l\}$: Full cohort of reference emulators and $\{a_l\} \in \{a_n\}$ learned as GPE hyperparameters using lasso regression as an indicator function to select the subset

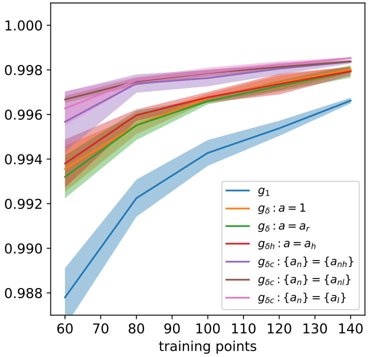
Results: Discrepancy emulators





Results: Discrepancy emulators





Results: Discrepancy emulators

Model	m=20		m=40		m=60	
g_1	0.895	0.915	0.988	0.977	0.995	0.988
$g_\delta : a = 1$	0.980	0.941	0.996	0.988	0.998	0.994
$g_\delta : a = a_r$	0.976	0.952	0.996	0.989	0.997	0.993
$g_{\delta h} : a = a_h$	0.983	0.959	0.996	0.991	0.998	0.994
$g_{\delta c} : \{a_n\} = \{a_{nh}\}$	0.976	0.959	0.995	0.994	0.997	0.996
$g_{\delta c} : \{a_n\} = \{a_{nl}\}$	0.980	0.976	0.997	0.994	0.998	**0.997**
$g_{\delta c} : \{a_n\} = \{a_l\}$	**0.993**	**0.984**	**0.998**	**0.994**	**0.999**	0.996
IE Equivalent:	m=50		m=100		m=140	

Conclusions

- Developed two methods for cohort emulation of digital twins
- Latent feature mode:
  - Could replace need for training any future emulators
  - Suffers from inconsistency
- Discrepancy emulator model:
  - Learns directly from previous emulators, utilising properties of Gaussian processes
  - Halves the number of simulations required for equivalent accuracy
  - Doesn't require knowledge of latent features
  - More computationally expensive than the latent model to add new cohort members.

Conclusions

- Developed two methods for cohort emulation of digital twins
- Latent feature mode:
  - Could replace need for training any future emulators
  - Suffers from inconsistency
- Discrepancy emulator model:
  - Learns directly from previous emulators, utilising properties of Gaussian processes
  - Halves the number of simulations required for equivalent accuracy
  - Doesn't require knowledge of latent features
  - More computationally expensive than the latent model to add new cohort members.

Thank you for listening!