Efficient emulation of high-dimensional outputs using manifold learning: Theory and applications

Akeel Shah, Wei Xing, Vasilis Triantafyllidis
(WCPM)

Prasanth Nair

(University of Toronto)

THE UNIVERSITY OF WARWICK

Outline of talk

- Problems and definition of emulator
- Types and basic construction
- 'Data' from PDE models
- Gaussian process emulation (scalar case)
- Gaussian process emulation for PDE outputs
- Gaussian process emulation for PDE outputs using dimensionality reduction
- Limitations of linear dimensionality reduction
- Nonlinear dimensionality reduction
- Application to PDE output data
- Emulation of multiple fields

Problem

- High fidelity simulation of fields, e.g., velocity or temperature field, in many computational physics and engineering problems
  - Design optimization (4 inputs, 10 values = 10^4 cases!)
  - Model calibration, validation, sensitivity analysis, uncertainty analysis (parameter/structural, numerical error)
- Computational cost of Monte Carlo based methods for sensitivity/uncertainty analysis can be prohibitive
- For field problems, the dimesionality of the output space is typically high, e.g., numerical grid 100 × 100 × 100 = 10^6 points
- Use emulators (meta-model, surrogate) to replace calls to full simulation model (simulator)
- Rapidly predict and visualize behavior of complex systems as functions of design parameters

Types of emulators

- How to construct emulators?
- Two basic approaches:
  - Data driven (statistical)
  - Physics based
- Data-driven emulators constructed by applying function approximation/machine learning (e.g., neural network, Gaussian process modeling) to input-output data
- Physics-based emulation works directly with the PDE model to achieve model order reduction, e.g. reduced basis expansion. No natural way of dealing with nonlinearities
- Both require careful design of experiment

PDE Problems: Spatio-temporal data

- **Nonlinear PDE model**

$$\partial_t u_i + \mathcal{F}_i(\boldsymbol{q}, t, \boldsymbol{u}, D\boldsymbol{u}, D^2\boldsymbol{u}, \ldots, D^n\boldsymbol{u}; \boldsymbol{x}) = 0 \quad \text{in } \Omega \times (0, T]$$

$$i = 1, \ldots, J$$

$$\boldsymbol{u} = (u_1, \ldots, u_J)^T, \ \boldsymbol{x} \in \mathcal{X} \subset \mathbb{R}^l \qquad \boldsymbol{q} = (\xi, \chi)$$

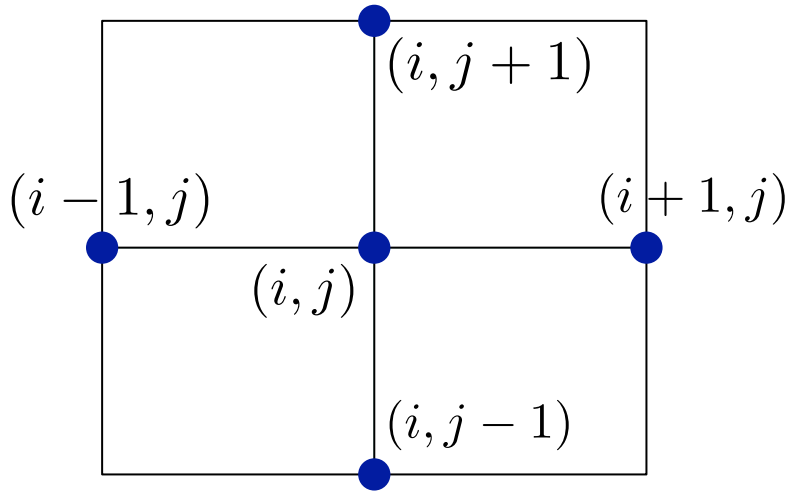
- $\mathcal{F}_i, i = 1, \ldots, J$ are nonlinear, parameterized operators
- In 2D, spatial grid

$$(\xi_i, \chi_j), \ i = 1, \ldots N_\xi, \ j = 1, \ldots N_\chi.$$

- Consider 1 quantity of interest (steady state): $u(\boldsymbol{q}; \boldsymbol{x})$.

PDE Problems: Spatio-temporal data

- Choose design inputs $\boldsymbol{x}^{(k)} \in \mathcal{X} \subset \mathbb{R}^l,\ k = 1, \ldots, m,$
- Simulator outputs $u^{(k)}(\xi_i, \chi_j),\ i = 1, \ldots N_\xi,\ j = 1, \ldots N_\chi$



$$\boldsymbol{y}^{(k)} = (u^{(k)}_{1,1}, \ldots, u^{(k)}_{1,N_\chi}, u^{(k)}_{2,1}, \ldots, u^{(k)}_{2,N_\chi}, \ldots\ldots\ldots, u^{(k)}_{N_\xi,1}, \ldots, u^{(k)}_{N_\xi,N_\chi})^T \in \mathbb{R}^d$$

- For time dependent case treat time as additional input parameter. Then there are mN_t samples (training points)
- For spatially uniform problems $d = N_t$.

PDE Problems: Spatio-temporal data

- Simulator can be considered as a mapping from the design space to the output space

$$\boldsymbol{\eta} : \mathcal{X} \rightarrow \mathbb{R}^d$$

- The goal of emulation is to approximate this mapping using training points generated as the design points

$$\boldsymbol{y}^{(i)} = \boldsymbol{\eta}(\boldsymbol{x}^{(i)}) \quad i = 1, \ldots, m.$$

- Machine learning strategies for such general (input-output) problems, e.g., neural networks, linear regression, Gaussian process emulation
- GP emulation is attractive because it is Bayesian and non-parametric (no basis functions to choose)

Gaussian process emulation: Scalar

- A GP is a stochastic process S_x, $x \in \mathcal{X}$: the joint probability density function restricted to a finite subset of the index set is multivariate Gaussian. A GP is specified by its mean and covariance functions
- Consider a scalar valued simulator $\eta : \mathbb{R}^l \to \mathbb{R}$
- Run as before at design points $\boldsymbol{x}^{(i)} \in \mathcal{X} \subset \mathbb{R}^l$
- Outputs are $y^{(i)} = \eta(\boldsymbol{x}^{(i)}), \quad i = 1, \ldots, m$
- GP prior assumption: $\eta(\cdot)$ is regarded as a GP indexed by the inputs $\boldsymbol{x}$

$$\eta(\boldsymbol{x}) = \boldsymbol{\beta}^T \boldsymbol{h}(\boldsymbol{x}) + \mathcal{G}(\boldsymbol{x}) + \epsilon(\boldsymbol{x})$$

Regression functions

Zero mean GP

Modelling measurement error

Gaussian process emulation: Scalar

- In Bayesian linear regression (without noise)

$$\eta(\boldsymbol{x}) = \boldsymbol{\beta}^T \boldsymbol{h}(\boldsymbol{x})$$

- Place a prior distribution on the weight vector, e.g., Gaussian with i.i.d coordinates, $\boldsymbol{\beta} \sim \mathcal{N}(\mathbf{0}, b^2 I)$
- Underlying function is distributed according to a Gaussian process indexed by the inputs $\boldsymbol{x}$, namely

$$\eta(\boldsymbol{x}) \sim \mathcal{GP}(0, b^2 \boldsymbol{h}(\boldsymbol{x})^T \boldsymbol{h}(\boldsymbol{x}'))$$

- GPE generalizes this concept by directly placing a covariance structure on the GP – allows for a much broader class of functions

Gaussian process emulation: Scalar

- Covariance function

$$k(\boldsymbol{x}, \boldsymbol{x}') = \sigma^2 c(\boldsymbol{x}, \boldsymbol{x}'; \boldsymbol{\theta})$$

- Stationary correlation function

$$c(\boldsymbol{x}, \boldsymbol{x}'; \boldsymbol{\theta}) = \exp\left(-(\boldsymbol{x} - \boldsymbol{x}')^T \mathrm{diag}(\theta_1, \ldots, \theta_l)(\boldsymbol{x} - \boldsymbol{x}')\right)$$

- Expresses smoothness of the GP (of the mean square derivatives to all orders)
- Conditional distribution of data $\boldsymbol{t} = (y^{(1)}, \ldots, y^{(m)})^T$

$$\boldsymbol{t}|\boldsymbol{\beta}, \sigma^2, \boldsymbol{\theta}, \sim \mathcal{N}(H\boldsymbol{\beta}, \sigma^2 C)$$

$$H = \left[\boldsymbol{h}(\boldsymbol{x}^{(1)}) \ldots \boldsymbol{h}(\boldsymbol{x}^{(m)})\right]^T$$
$$[C]_{ij} = c(\boldsymbol{x}^{(i)}, \boldsymbol{x}^{(j)}; \boldsymbol{\theta})$$

Gaussian process emulation: Scalar

- Predictive distribution given data and hyperparameters for a test point $\boldsymbol{x}$

$$\eta(\cdot)|\boldsymbol{\beta}, \sigma^2, \boldsymbol{\theta}, \boldsymbol{t} \sim \mathcal{GP}\left(\mu'(\cdot), \sigma^2 \nu'(\cdot,\cdot)\right)$$

$$\mu'(\boldsymbol{x}) = \boldsymbol{\beta}^T \boldsymbol{h}(\boldsymbol{x}) + \boldsymbol{a}(\boldsymbol{x})^T C^{-1}(\boldsymbol{t} - H\boldsymbol{\beta})$$
$$\nu'(\boldsymbol{x}, \boldsymbol{x}') = c(\boldsymbol{x}, \boldsymbol{x}'; \boldsymbol{\theta}) - \boldsymbol{a}(\boldsymbol{x})^T C^{-1} \boldsymbol{a}(\boldsymbol{x}')$$
$$\boldsymbol{a}(\boldsymbol{x}) = \left(c(\boldsymbol{x}^{(1)}, \boldsymbol{x}; \boldsymbol{\theta}), \ldots c(\boldsymbol{x}^{(m)}, \boldsymbol{x}; \boldsymbol{\theta})\right)^T$$

- Once hyperparameters are known, predictions can be made easily and quickly with simple formulae
- Formulae can be extended to predict several points simultaneously

Gaussian process emulation: Scalar

- Fully Bayesian approach places a prior $f(\boldsymbol{\theta})$ on these hyperparameters and uses MCMC to present predictions as a sample
- Time consuming process so often 'plug in' (point) estimates are used
- Maximum a-posteriori (MAP) using a conjugate prior
- Maximum likelihood estimate (MLE)

$$\boldsymbol{\theta}_{MLE} = \arg\max_{\boldsymbol{\theta}} \left(-\frac{1}{2}\ln|C| - \frac{1}{2}\boldsymbol{t}^T C^{-1}\boldsymbol{t} - \frac{m}{2}\ln(2\pi) \right)$$

- Depends on number of samples m

GP emulation for PDE outputs

- How do we extend the scalar case to outputs in a high dimensional space?
- A naive approach (Kennedy and O'Hagan (2001)) is to treat the output index as an additional parameter and perform scalar GPE – for emulating a whole field this requires *d* computations
- How do we overcome this problem?
- Consider (Paulo et al. (2102)) the linear model of coregionalization (Wackernagel (1995))

$$\boldsymbol{\eta}(\boldsymbol{x}) = A\boldsymbol{w}(\boldsymbol{x}) \qquad \boldsymbol{w}(\boldsymbol{x}) = (w_1(\boldsymbol{x}), \ldots, w_J(\boldsymbol{x}))^T$$

- Independent zero-mean GPs. A independent of $\boldsymbol{x}$

GP emulation for PDE outputs

- Correlation functions for the GPs are $c_i(\cdot, \cdot, \boldsymbol{\theta}_i)$
- The covariance function for the multivariate GP is

$$\mathrm{Cov}(\boldsymbol{\eta}(\boldsymbol{x}), \boldsymbol{\eta}(\boldsymbol{x}')) = \sum_{i=1}^{J} \boldsymbol{a}_i \boldsymbol{a}_i^T c_i(\boldsymbol{x}, \boldsymbol{x}', \boldsymbol{\theta}_i)$$

- Fairly general assumption – allows different scales to be incorporated by using a linear combination of correlation functions with different scale parameters $\boldsymbol{\theta}_i$
- We can assume a separable structure for the covariance by taking all w_i to be i.i.d., i.e. a single correlation function
- Leads to a tractable problem (Conti & O'Hagan (2009); Rougier (2008)) – equivalent to a multivariate GP prior

GP emulation for PDE outputs using DR

- An alternative method is to use the data to find a 'suitable' A and at the same time restrict the number of univariate GPs to those that contribute the 'most'
- Leads to a reduction in dimensionality of the output space (restrict to a linear subspace) – Higdon et al. (2008)
- The method relies on only principal component analysis
- Singular value decomposition of data matrix (or eigen-decomposition of sample covariance matrix) reorders data according to variance in the d dimensions with uncorrelated coefficients
- Natural basis (columns of A) for output space and expansion in terms of coefficients (assumed to be GPs)

GP emulation for PDE outputs using DR

- Orthonormal basis $\boldsymbol{p}_j, i = 1, \ldots, m$ for $\mathbb{R}^d$. Select linear subspace: $\text{span}\{\boldsymbol{p}_1, \ldots, \boldsymbol{p}_r\}, r \ll d$
- Coefficients in this basis $c_j^{(i)}, j = 1, \ldots, d, i = 1, \ldots, m$
- Expansion (restrict to subspace)

$$\boldsymbol{y} = \boldsymbol{\psi}(\boldsymbol{x}) \approx \sum_{j=1}^{r} c_j \boldsymbol{p}_j$$

Select a test point $\boldsymbol{x}$ for prediction

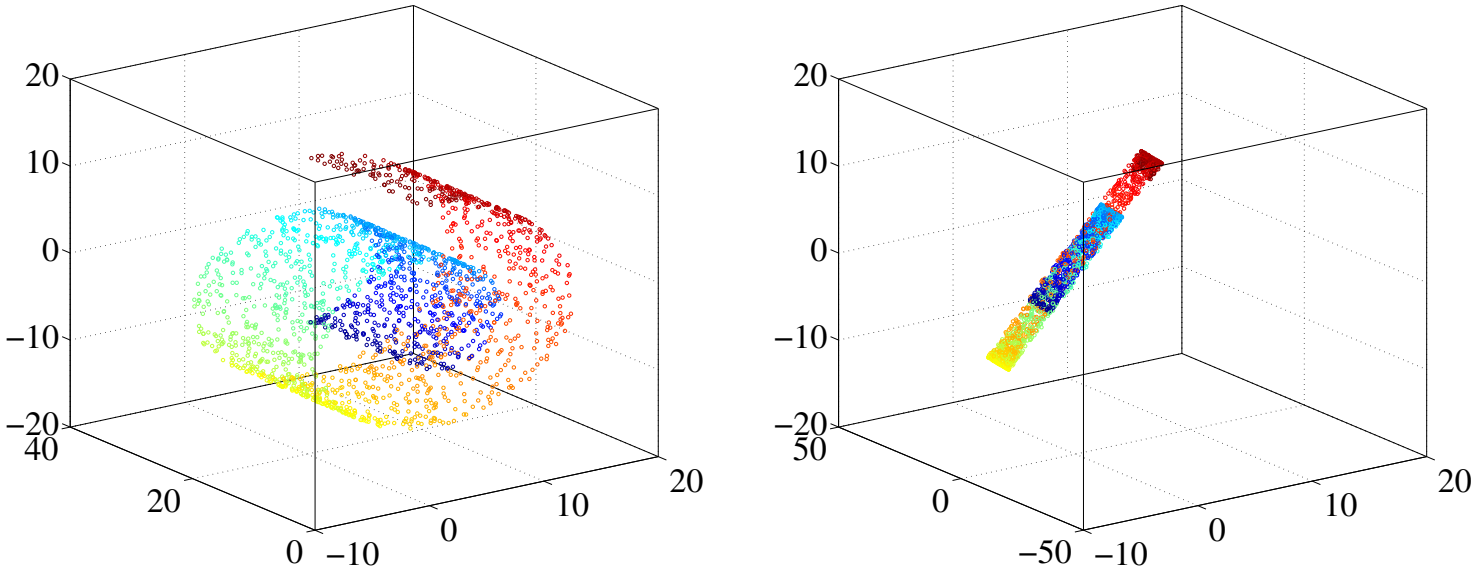
for $j = 1 : r$

scalar GPE on training set $c_j^{(i)}$, $\boldsymbol{x}^{(i)}$, $i = 1, \ldots, m$, prediction c_j

end

Approximate $\boldsymbol{y} = \boldsymbol{\psi}(\boldsymbol{x}) \approx \sum_{j=1}^{r} c_j \boldsymbol{p}_j$

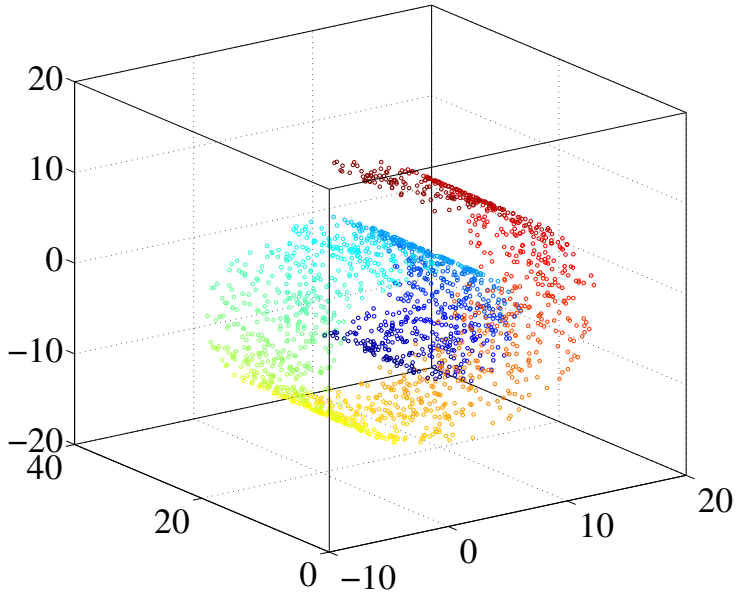
Limitations of linear DR methods



- Fails in many cases to provide an accurate representation of a response surface
- Informally, works with a relatively 'flat' surface
- For highly nonlinear response surfaces with abrupt changes it could fail completely

Nonlinear Dimensionality Reduction

- There are other methods for dimensionality reduction
- Linear methods
  - PCA
  - Multi-dimensional scaling
  - Independent component analysis
- **Nonlinear methods**
  - Kernel PCA
  - Isomap/kernel Isomap
  - Diffusion maps
  - Laplacian eigenmaps
  - Local linear embedding



Nonlinear Dimensionality Reduction

- Manifold assumption: the input data resides on or close to a 'low-dimensional' manifold embedded in the ambient space – informally the dimension is the number of parameters needed to specify a point, e.g., a surface of a sphere has dimension 2
- Learning/characterizing such manifolds from given data is called manifold learning
- Approaches are characterized in a number of ways, e.g., spectral, kernel-based, embeddings
- Each have their own advantages and disadvantages – no universal technique
- Performance on toy data sets can be misleading

Using kernel PCA for field emulation

- Mapped to a higher dimensional (possibly infinite) feature space and apply LPCA to mapped data (Scholkopf et al. (1998))
- Transform data in such a way that it lies in (or near) a linear subspace of the feature space
- Feature space can be very high dimensional (possibly infinite)
- Feature map $\boldsymbol{\phi} : \mathbb{R}^d \rightarrow \mathcal{F}$ is implicitly specified via kernel function

$$\widetilde{\boldsymbol{\phi}}(\boldsymbol{y}^{(i)})^T \widetilde{\boldsymbol{\phi}}(\boldsymbol{y}^{(j)}) = \widetilde{k}(\boldsymbol{y}^{(i)}, \boldsymbol{y}^{(j)}) = \widetilde{K}_{ij}$$

$$\widetilde{\boldsymbol{\phi}}(\boldsymbol{y}^{(i)}) = \boldsymbol{\phi}(\boldsymbol{y}^{(i)}) - \overline{\boldsymbol{\phi}}$$

Using kernel PCA for field emulation

- In general the map is not known explicitly
- Recast eigenproblem for sample covariance matrix (in feature space) as a eigenproblem for kernel matrix $\widetilde{\boldsymbol{\alpha}}_i$
- Eigenvectors of sample covariance matrix $\widetilde{\boldsymbol{v}}_i$ are not known but the coefficients in an expansion are known

$$z_i^{(j)} = \widetilde{\boldsymbol{v}}_i^T \widetilde{\boldsymbol{\phi}}_j = \sum_{k=1}^{m} \widetilde{\alpha}_{ki} \widetilde{\boldsymbol{\phi}}_k^T \widetilde{\boldsymbol{\phi}}_j = \sum_{k=1}^{m} \widetilde{\alpha}_{ki} \widetilde{K}_{kj}$$

- Do GPE on first r coefficients (test input $\boldsymbol{x}$) to approximate projection of $\boldsymbol{\phi}(\boldsymbol{y}(\boldsymbol{x}))$ onto $\mathrm{span}\{\widetilde{\boldsymbol{v}}_1, \ldots, \widetilde{\boldsymbol{v}}_r\}$

$$\widehat{\boldsymbol{\phi}}(\boldsymbol{y}(\boldsymbol{x})) = \sum_{i=1}^{r} z_i(\boldsymbol{x}) \widetilde{\boldsymbol{v}}_i + \overline{\boldsymbol{\phi}}$$

Using kernel PCA for field emulation

- Examples of a kernel functions

Gaussian kernel	$e^{-\frac{1}{2s^2}\|\|\boldsymbol{y}^{(i)}-\boldsymbol{y}^{(j)}\|\|^2}$
Polynomial (order n)	$((\boldsymbol{y}^{(i)})^T\boldsymbol{y}^{(j)}+s)^n$
Multiquadric	$\sqrt{1+s\|\|\boldsymbol{y}^{(i)}-\boldsymbol{y}^{(j)}\|\|^2}$
Sigmoid	$\tanh(s(\boldsymbol{y}^{(i)})^T\boldsymbol{y}^{(j)}+s')$

- Since the map is not known (nor the basis vectors in feature space), a pre-image problem has to be solved – approximate the inverse map
- It turns out this is possible to do in 3 main ways for most standard kernels

Using kernel PCA for field emulation

- Least squares approximation is possible by expressing distances between points in physical space to distances between points in feature space via kernel function. Method can suffer from numerical instabilities if $m < d$.
- A fixed-point iterative algorithm (Mika et al. (1999)) can be used but is again prone to instability
- Local linear interpolation (Ma & Zabaras (2011)) is the third method (again based on distance information) – use a weighted sum of known data point values. This gives stable results

Summary of method

1. Select design points $\mathbf{x}^{(j)} \in \mathcal{X} \subset \mathbb{R}^l$, $j = 1, \ldots, m$, using DOE
2. Collect outputs $\mathbf{y}^{(j)} = \boldsymbol{\eta}(\mathbf{x}^{(j)}) \in \mathbb{R}^d$ from the simulator
3. Perform kPCA on $\mathbf{y}^{(j)}$ $\Rightarrow$ $z_i^{(j)}$, $j = 1, \ldots, m$, $i = 1, \ldots, d$
4. Select a test point $\mathbf{x}$ for prediction

   for $i = 1 : r$

      perform scalar GPE on $\mathcal{D}_i = \{\mathbf{x}^{(j)}, z_i^{(j)}\}_{j=1}^m$ $\Rightarrow$ z_i

   end

Reconstruct $\Rightarrow$ $\widehat{\mathbf{y}} = \boldsymbol{\phi}^{-1}(\widehat{\boldsymbol{\phi}}) \approx \boldsymbol{\eta}(\mathbf{x})$

Example 1

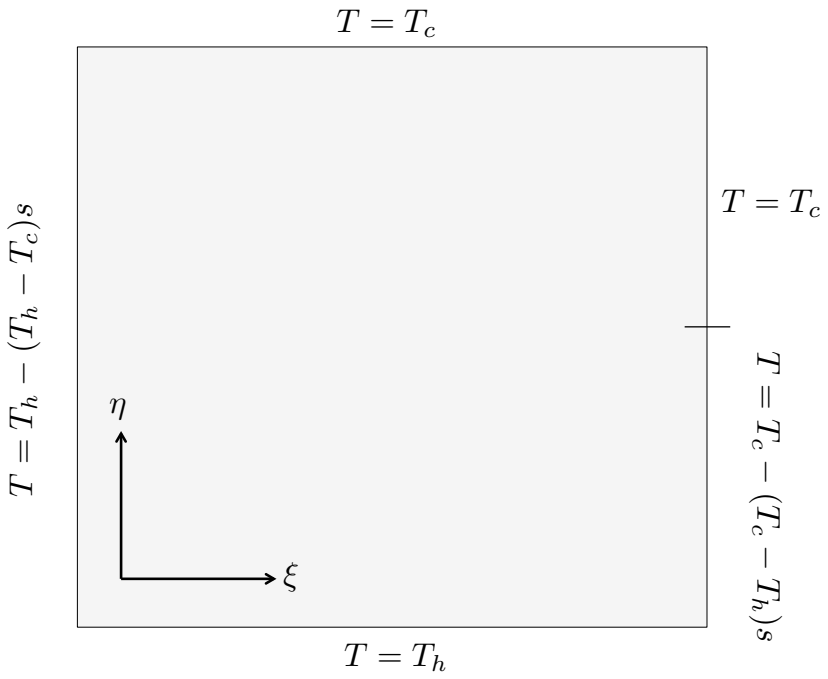
Subsurface flow in porous media driven by density variations

$$\frac{\mu}{\kappa}\mathbf{u} + \nabla p - \nabla \cdot \frac{\mu}{\epsilon}\left(\nabla \mathbf{u} + \nabla \mathbf{u}^T\right) = \rho \mathbf{g} \beta (T - T_c)$$

$$\nabla \cdot \mathbf{u} = 0$$

$$\rho C_p \mathbf{u} \cdot \nabla T - \nabla \cdot (k_m \nabla T) = 0$$

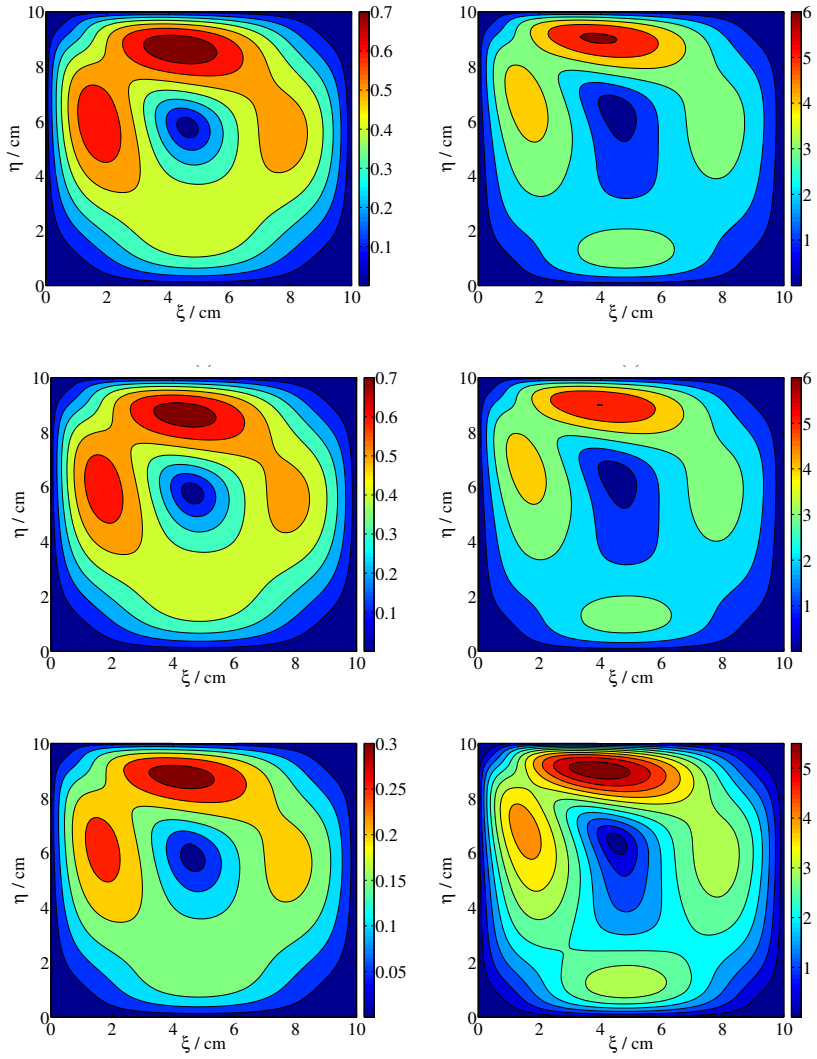
- Use Brinkman's equation with Boussinesq buoyancy term
- Temperature varies from high T_h to T_c along outer edges
- Initially water stagnant but temperature gradients alter fluid density and buoyant flow generated

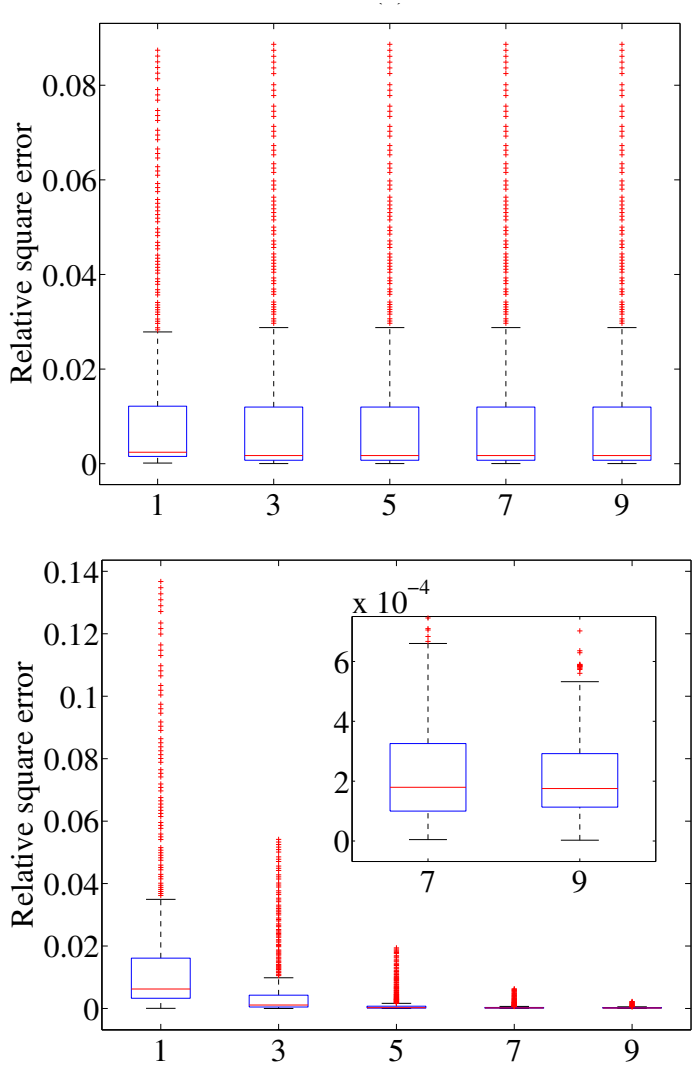


Example 1: Training and Testing

- Two input parameters varied: coefficient of volumetric thermal expansion β and the high temperature T_h
- A total of 500 numerical experiments were performed, with inputs selected using a Sobol sequence
- For each simulation, magnitude of the velocity |u| was recorded on regular 100 × 100 square spatial grid
- The 10000 points in the 2D spatial domain re-ordered into vector form in
- 400 samples reserved for testing

Example 1: Results





Example 2

- CSTR used to produce propylene glycol (PrOH) from the propylene oxide (PrO) with water in the presence of H_2SO_4
- Mass Balances

$$V_r \frac{dc_i}{dt} = v_f(c_{f,i} - c_i) + \nu_i V_r r$$

- Heat balance

$$\sum_i c_i C_{p,i} \frac{dT}{dt} = -Hr + \frac{F_x C_{p,x}(T_x - T)}{V_r}\left(1 - e^{UA/(F_x C_{p,x})}\right) + \sum_i \frac{v_f c_{f,i}(h_{f,i} - h_i)}{V_r}$$

Heat of reaction — Heat loss to HX — Convective heat flow

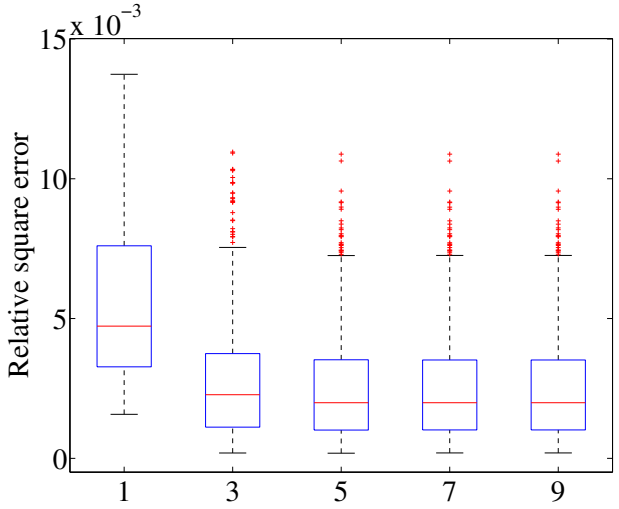
- Molar enthalpy of species i

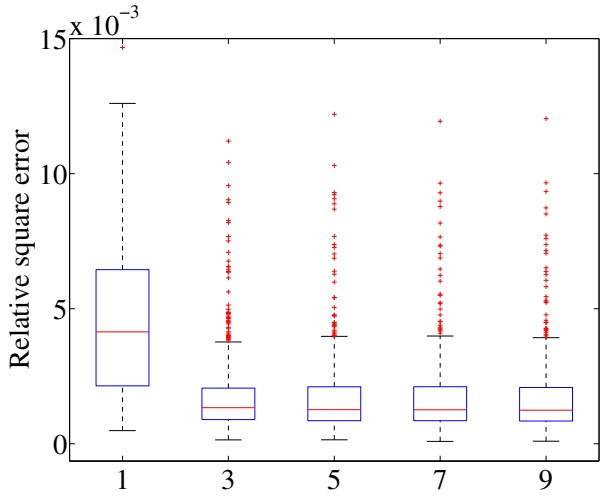
$$h_i = C_{p,i}(T - T_{ref}) + h_{i,ref}$$

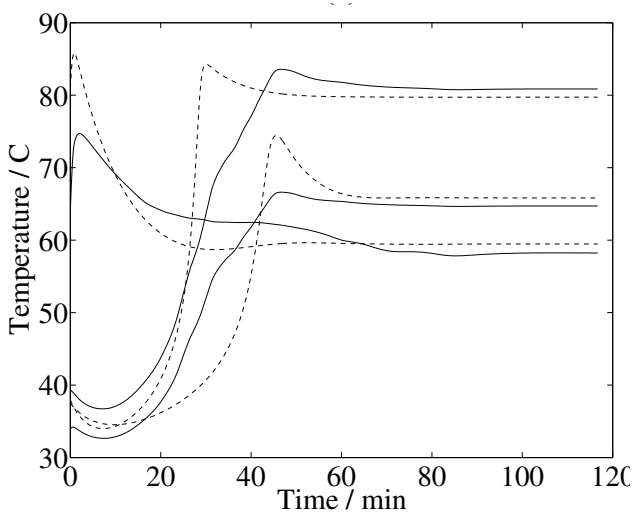
Example 2: Training and Testing

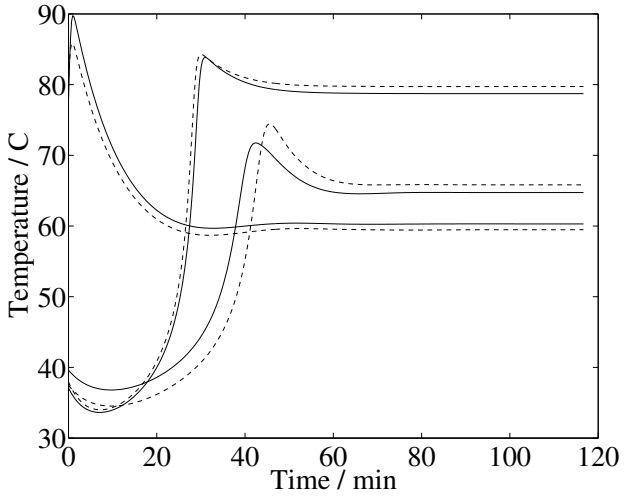
- The model was solved in COMSOL Multiphysics 4.3b ('Free convection in porous media')
- Three input parameters varied: initial temperatures, initial concentrations of PrO, heat exchange parameter, *UA*
- A total of 500 numerical experiments were performed, with inputs selected using a Sobol sequence
- Temperature recorded every 14 s up to 7000 s
- The 501 points re-ordered into vector form
- 400 samples reserved for testing

Example 2: Results





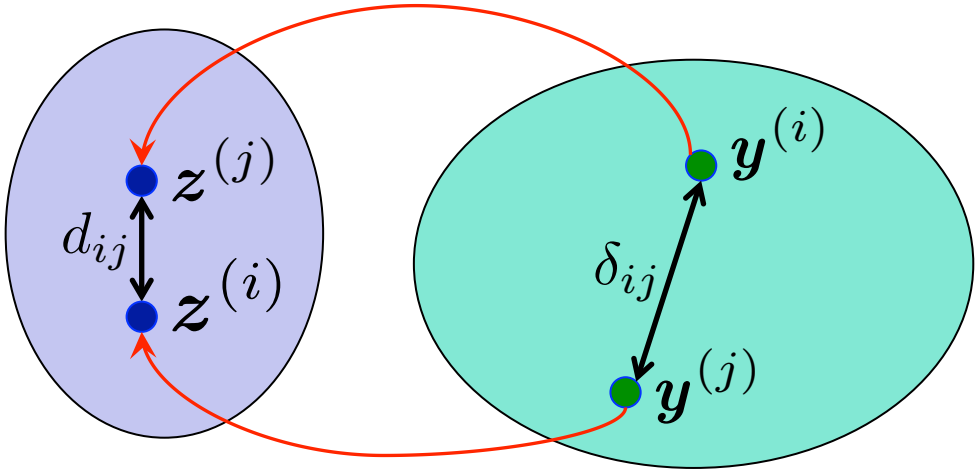




Isomap

- Classical multidimensional scaling provides a low-dimensional Euclidean space representation of data that lies on a manifold in a high dimensional ambient space
- It relates 'dissimilarities' $\boldsymbol{d_{ij}}$ between points i and j to Euclidean distances $\boldsymbol{\delta_{ij}}$ in the low-dimensional space
- Classical scaling is an isometric embedding

$$\delta_{ij} = ||\boldsymbol{z}^{(i)} - \boldsymbol{z}^{(j)}|| = d_{ij}$$



Dissimilarity matrix

$$D = [d_{ij}]$$

Isomap

- When dissimilarities are defined by Euclidean distance MDS is equivalent to PCA (easily seen from least-squares optimality of PCA)
- Method is also spectral: eigenvectors of a centred kernel matrix $K = -(1/2)H(D \circ D)H$
- Idea generalised by Tenenbaum et al. using geodesic distances for d_{ij} (e.g. shortest path distance) – Isomap
- Can be considered equivalent to kPCA: Dissimilarity matrix defines distances between points in feature space and leads to a centred kernel matrix
- Coordinates obtained from a spectral decomposition same as before, provided the kernel matrix is p.s.d.

Isomap

- Kernel ISOMAP guarantees p.s.d. kernel matrices and therefore the existence of a feature space
- Can use same procedure as before on coefficients learned by ISOMAP
- Pre-image problem can be solved similarly by relating distances between points in the low-dimensional space to dissimilarities (= Euclidean distances for ‘neighbours’)

Example

Metal melting front: a square cavity containing solid and liquid submitted to a temperature difference between the left and right boundaries

$$\rho_0 \frac{\partial \mathbf{u}}{\partial t} + \rho_0 (\mathbf{u} \cdot \nabla)\mathbf{u} + \nabla p - \nabla \cdot \mu \left(\nabla \mathbf{u} + \nabla \mathbf{u}^T \right) - \rho \mathbf{g} = 0$$

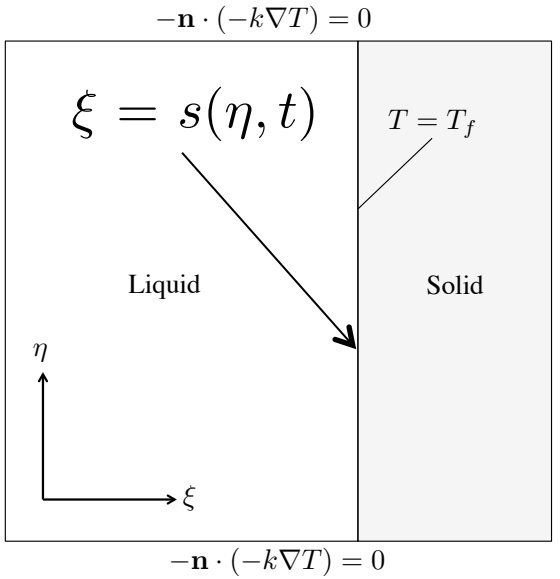
$$\nabla \cdot \mathbf{u} = 0$$

$$\rho C_p \frac{\partial T_l}{\partial t} + \rho C_p \mathbf{u} \cdot \nabla T_l - \nabla \cdot (k \nabla T_l) = 0$$

$$\rho = \rho_0 \beta (T_l - T_f)$$

$$\rho C_p \frac{\partial T_s}{\partial t} - \nabla \cdot (k \nabla T) = 0$$

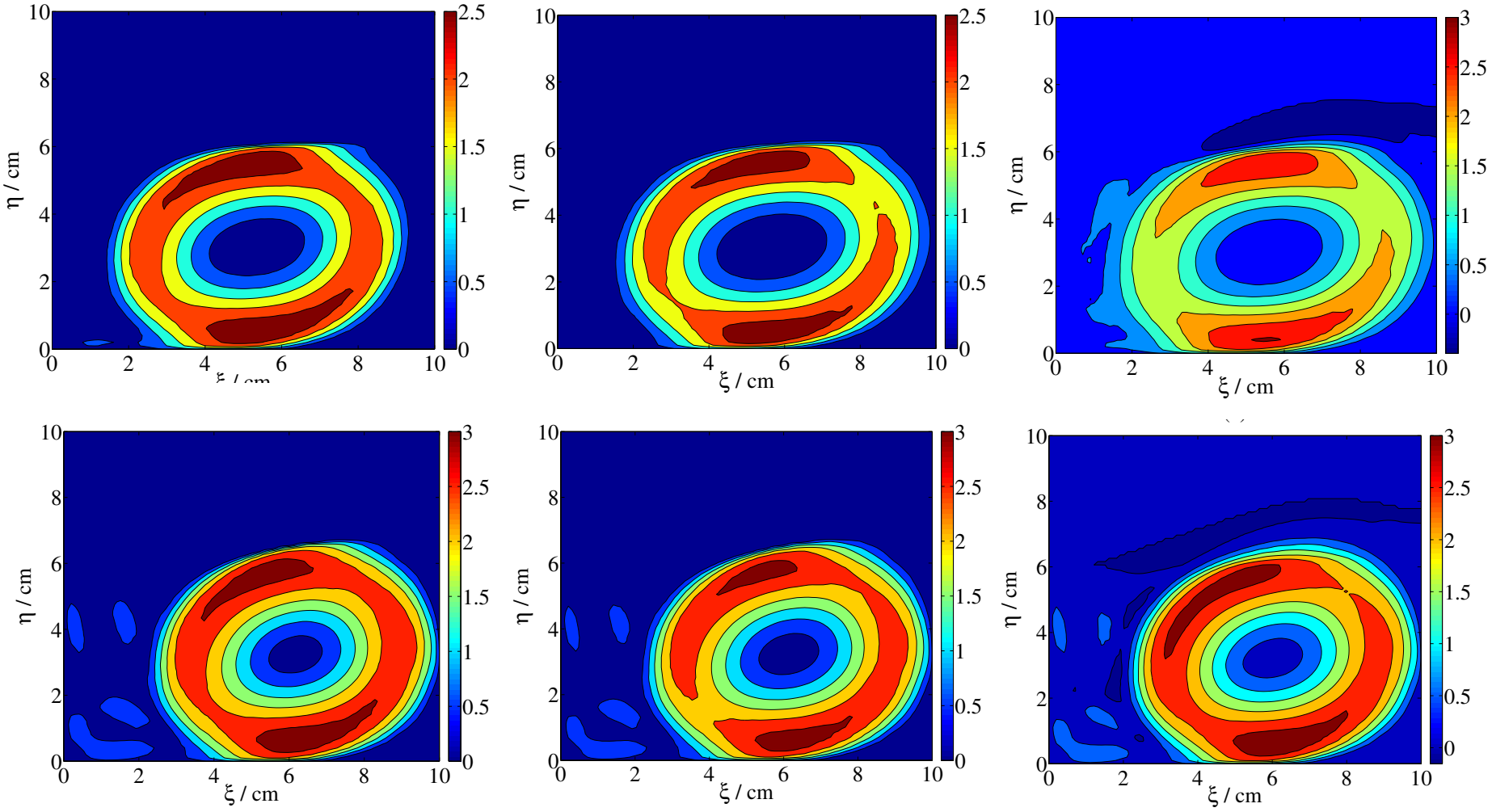
$$\rho_0 \Delta h_f \frac{\partial s}{\partial t} = \left(1 + \left(\frac{\partial s}{\partial y} \right)^2 \right) \left(k \frac{\partial T_s}{\partial y} - k \frac{\partial T_l}{\partial x} \right)$$



Example: Training and Testing

- The model was solved in COMSOL Multiphysics 4.3b
- Two input parameters varied: latent heat of fusion Δh_f and thermal conductivity k
- 50 numerical experiments were performed, with inputs selected using a Sobol sequence.
- For each set of parameters, 10 snapshots of the velocity field were recorded for $t = 50, 100, \ldots, 500$ s.
- For each simulation, magnitude of the velocity $|u|$ was recorded on regular 100×100 square spatial grid
- 400 samples reserved for testing

Example: Results



Emulating multiple fields

- We can emulate multiple fields or vector fields by combining data sets or separate emulation assuming independence
- Could lead to problems with scaling (e.g., temperature variations vs. velocity variations) or ignores correlations between outputs (e.g., electric potential and current)
- Multiple outputs types (fields) $\boldsymbol{y}_j^{(i)}, j = 1, \ldots, J$
- Perform NDR for each output type and extract coefficients $z_{k,j}^{(i)}$ where k indexes the coefficient number, j indexes the output type and i indexes inputs
- For a fixed k, define $\boldsymbol{Z}_k^{(i)} = (z_{k,1}^{(i)}, \ldots, z_{k,J}^{(i)})^T$

Emulating multiple fields

- Use LMC to infer coefficients simultaneously for test inputs: a *J*-variate GP

$$\boldsymbol{\mathcal{Z}}_k(\cdot) = F_k \boldsymbol{\mathcal{W}}_k(\cdot)$$

- $\boldsymbol{\mathcal{W}}_k = (\mathcal{W}_{1,k} \ldots, \mathcal{W}_{J,k})^T$ where coordinates are independent GPs with zero mean and correlation functions $c_{i,k}(\cdot, \cdot, \boldsymbol{\theta}_{i,k})$

$$\boldsymbol{\mathcal{Z}}_k(\cdot) | \text{ parameters, data} \sim \mathcal{GP}\left(\boldsymbol{\mathcal{M}}_k(\cdot), \mathcal{K}_k(\cdot, \cdot)\right)$$

- Explicit formulae for mean function $\boldsymbol{\mathcal{M}}_k(x)$ and variance-covariance matrix $\mathcal{K}_k(x, x)$ are given
- Parameters determined via likelihood or MCMC
- Repeat for each *k* and solve pre-image problem

Other work

- Diffusion maps, with a new method for solving pre-image problem based on an extended diffusion matrix and local interpolation
- Physics based approaches using NDR (direct approach?)
- Large scale problems (more complex data sets) including issues with DOE, number of samples and dimensionality of input space

References

1) M. Kennedy, A. O'Hagan, J. R. Stat. Soc. Ser. B Stat. Methodol. 63 (2001) 425–464
2) R. Paulo, G. Garcia-Donato, J. Palomo, Comput. Stat. Data Analysis 56 (2012) 3959–3974
3) H. Wackernagel, Multivariate Geostatistics, Springer, Berlin, 1995.
4) S. Conti, A. O'Hagan, J. Statistical Planning and Inference 140 (2010) 640–651
5) J. Rougier, J. Computational and Graphical Statistics 17 (2008) 827–843
6) D. Higdon et al., J. Am. Stat. Association 103 (482) (2008) 570–583
7) B. Scholkopf, A. Smola, K.-R. Muller, Neural Computation 10 (1998) 1299–1319
8) S. Mika et al., Advances in neural information processing systems II 11 (1999) 536–542
9) B. Ganapathysubramanian, N. Zabaras, J. Comput. Phys. 227 (2008) 6612–6637
10) X. Ma, N. Zabaras, J. Comput. Phys. 230 (2011) 7311-7331
11) J.B. Tenenbaum, V. De Silva, J.C. Langford, Science 290 (2000) 2319–2323