50 Ways with GPs

Richard Wilkinson

School of Maths and Statistics
University of Sheffield

Emulator workshop
June 2017

Recap

A Gaussian process is a random process indexed by some variable ($x \in \mathcal{X}$ say), such that for every finite set of indices, $x_1, \ldots, x_n$, then

$$\mathbf{f} = (f(x_1), \ldots, f(x_n))$$

has a multivariate Gaussian distribution.

Recap

A Gaussian process is a random process indexed by some variable ($x \in \mathcal{X}$ say), such that for every finite set of indices, $x_1, \ldots, x_n$, then

$$\mathbf{f} = (f(x_1), \ldots, f(x_n))$$

has a multivariate Gaussian distribution.

Why would we want to use this very restricted model?

Answer 1

Class of models is closed under various operations.

Answer 1

Class of models is closed under various operations.

- Closed under addition

$$f_1(\cdot), f_2(\cdot) \sim GP \quad \text{then} \quad (f_1 + f_2)(\cdot) \sim GP$$

Answer 1

Class of models is closed under various operations.

- Closed under addition

$$f_1(\cdot), f_2(\cdot) \sim GP \quad \text{then} \quad (f_1 + f_2)(\cdot) \sim GP$$

- Closed under Bayesian conditioning, i.e., if we observe

$$\mathbf{D} = (f(x_1), \ldots, f(x_n))$$

then

$$f|D \sim GP$$

but with updated mean and covariance functions.

Answer 1

Class of models is closed under various operations.

- Closed under addition

$$f_1(\cdot), f_2(\cdot) \sim GP \quad \text{then} \quad (f_1 + f_2)(\cdot) \sim GP$$

- Closed under Bayesian conditioning, i.e., if we observe

$$\mathbf{D} = (f(x_1), \ldots, f(x_n))$$

then

$$f|D \sim GP$$

but with updated mean and covariance functions.

- Closed under any linear operation. If $\mathcal{L}$ is a linear operator, then

$$\mathcal{L} \circ f \sim GP(\mathcal{L} \circ m, \mathcal{L}^2 \circ k)$$

e.g. $\frac{df}{dx}$, $\int f(x)dx$, Af are all GPs

Answer 2: non-parametric/kernel regression

k determines the space of functions that sample paths live in.

Answer 2: non-parametric/kernel regression

k determines the space of functions that sample paths live in.

- Linear regression $y = x^\top \beta + \epsilon$ can be written solely in terms of inner products $x^\top x$.

$$\hat{\beta} = \arg\min ||y - X\beta||_2^2 + \sigma^2 ||\beta||_2^2$$

Answer 2: non-parametric/kernel regression

k determines the space of functions that sample paths live in.

- Linear regression $y = x^\top \beta + \epsilon$ can be written solely in terms of inner products $x^\top x$.

$$\begin{aligned}\hat{\beta} &= \arg\min ||y - X\beta||_2^2 + \sigma^2 ||\beta||_2^2 \\ &= (X^\top X + \sigma^2 I) X^\top y \\ &= X^\top (XX^\top + \sigma^2 I)^{-1} y \quad \text{(the dual form)}\end{aligned}$$

Answer 2: non-parametric/kernel regression

k determines the space of functions that sample paths live in.

- Linear regression $y = x^\top \beta + \epsilon$ can be written solely in terms of inner products $x^\top x$.

$$\begin{aligned}\hat{\beta} &= \arg\min ||y - X\beta||_2^2 + \sigma^2 ||\beta||_2^2 \\ &= (X^\top X + \sigma^2 I) X^\top y \\ &= X^\top (XX^\top + \sigma^2 I)^{-1} y \quad \text{(the dual form)}\end{aligned}$$

So the prediction at a new location x' is

$$\begin{aligned}\hat{y}' = x'^\top \hat{\beta} &= x'^\top X^\top (XX^\top + \sigma^2 I)^{-1} y \\ &= k(x')(K + \sigma^2 I)^{-1} y\end{aligned}$$

where $k(x') := (x'^\top x_1, \ldots, x'^\top x_n)$ and $K_{ij} := x_i^\top x_j$

Answer 2: non-parametric/kernel regression

k determines the space of functions that sample paths live in.

- Linear regression $y = x^\top \beta + \epsilon$ can be written solely in terms of inner products $x^\top x$.

$$\begin{aligned}\hat{\beta} &= \arg\min ||y - X\beta||_2^2 + \sigma^2 ||\beta||_2^2 \\ &= (X^\top X + \sigma^2 I) X^\top y \\ &= X^\top (XX^\top + \sigma^2 I)^{-1} y \quad \text{(the dual form)}\end{aligned}$$

So the prediction at a new location x' is

$$\begin{aligned}\hat{y}' = x'^\top \hat{\beta} &= x'^\top X^\top (XX^\top + \sigma^2 I)^{-1} y \\ &= k(x')(K + \sigma^2 I)^{-1} y\end{aligned}$$

where $k(x') := (x'^\top x_1, \ldots, x'^\top x_n)$ and $K_{ij} := x_i^\top x_j$

- We know that we can replace x by a feature vector in linear regression, e.g., $\phi(x) = (1\ x\ x^2)$ etc.
  Then

$$K_{ij} = \phi(x_i)^\top \phi(x_j) \qquad \text{etc}$$

- For some sets of features, the inner product is equivalent to evaluating a kernel function

$$\phi(x)^\top \phi(x') \equiv k(x, x')$$

where

$$k : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$$

is a semi-positive definite function.

- For some sets of features, the inner product is equivalent to evaluating a kernel function

$$\phi(x)^\top \phi(x') \equiv k(x, x')$$

where

$$k : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$$

is a semi-positive definite function.

- We can use an infinite dimensional feature vector $\phi(x)$, and because linear regression can be done solely in terms of inner-products (inverting a $n \times n$ matrix in the dual form) we never need evaluate the feature vector, only the kernel.

Kernel trick: lift x into feature space by replacing inner products $x^\top x'$ by $k(x, x')$

- For some sets of features, the inner product is equivalent to evaluating a kernel function

$$\phi(x)^\top \phi(x') \equiv k(x, x')$$

where

$$k : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$$

is a semi-positive definite function.

- We can use an infinite dimensional feature vector $\phi(x)$, and because linear regression can be done solely in terms of inner-products (inverting a $n \times n$ matrix in the dual form) we never need evaluate the feature vector, only the kernel.

  Kernel trick: lift x into feature space by replacing inner products $x^\top x'$ by $k(x, x')$

Kernel regression/non-parametric regression/GP regression all closely related:

$$\hat{y}' = m(x') = \sum_{i=1}^{n} \alpha_i k(x, x_i)$$

Generally, we don't think about these features, we just choose a kernel. But any kernel is implicitly choosing a set of features, and our model only includes functions that are linear combinations of this set of features (this space is called the Reproducing Kernel Hilbert Space (RKHS) of k).

Generally, we don't think about these features, we just choose a kernel. But any kernel is implicitly choosing a set of features, and our model only includes functions that are linear combinations of this set of features (this space is called the Reproducing Kernel Hilbert Space (RKHS) of k).

Generally, we don't think about these features, we just choose a kernel. But any kernel is implicitly choosing a set of features, and our model only includes functions that are linear combinations of this set of features (this space is called the Reproducing Kernel Hilbert Space (RKHS) of k).

Example: If (modulo some detail)

$$\phi(x) = (e^{-\frac{(x-c_1)^2}{2\lambda^2}}, \ldots, e^{-\frac{(x-c_N)^2}{2\lambda^2}})$$

then as $N \to \infty$ then

$$\phi(x)^\top \phi(x) = \exp\left(-\frac{(x-x')^2}{2\lambda^2}\right)$$

Generally, we don't think about these features, we just choose a kernel. But any kernel is implicitly choosing a set of features, and our model only includes functions that are linear combinations of this set of features (this space is called the Reproducing Kernel Hilbert Space (RKHS) of k).

Example: If (modulo some detail)

$$\phi(x) = (e^{-\frac{(x-c_1)^2}{2\lambda^2}}, \ldots, e^{-\frac{(x-c_N)^2}{2\lambda^2}})$$

then as $N \to \infty$ then

$$\phi(x)^\top \phi(x) = \exp\left(-\frac{(x-x')^2}{2\lambda^2}\right)$$

Although our simulator may not lie in the RKHS defined by k, this space is much richer than any parametric regression model (and can be dense in some sets of continuous bounded functions), and is thus more likely to contain an element close to the simulator than any class of models that contains only a finite number of features.

This is the motivation for non-parametric methods.

Answer 3: Naturalness of GP framework

Why use **Gaussian** processes as non-parametric models?

1

Answer 3: Naturalness of GP framework

Why use **Gaussian** processes as non-parametric models?

One answer might come from Bayes linear methods[1].
If we only knew the expectation and variance of some random variables, X and Y, then how should we best do statistics?

[1] Some crazy cats think we should do statistics without probability

Answer 3: Naturalness of GP framework

Why use **Gaussian** processes as non-parametric models?

One answer might come from Bayes linear methods[1].
If we only knew the expectation and variance of some random variables, X and Y, then how should we best do statistics?

It has been shown, using coherency arguments, or geometric arguments, or..., that the best second-order inference we can do to update our beliefs about X given Y is

$$\mathbb{E}(X|Y) = \mathbb{E}(X) + \mathbb{C}\text{ov}(X, Y)\mathbb{V}\text{ar}(Y)^{-1}(Y - \mathbb{E}(Y))$$

i.e., exactly the Gaussian process update for the posterior mean.
So GPs are in some sense second-order optimal.

[1] Some crazy cats think we should do statistics without probability

Answer 4: Uncertainty estimates from emulators

We often think of our prediction as consisting of two parts

- point estimate
- uncertainty in that estimate

That GPs come equipped with the uncertainty in their prediction is seen as one of their main advantages.

Answer 4: Uncertainty estimates from emulators

We often think of our prediction as consisting of two parts

- point estimate
- uncertainty in that estimate

That GPs come equipped with the uncertainty in their prediction is seen as one of their main advantages.

It is important to check both aspects (see Lindsay's talk)

Answer 4: Uncertainty estimates from emulators

We often think of our prediction as consisting of two parts

- point estimate
- uncertainty in that estimate

That GPs come equipped with the uncertainty in their prediction is seen as one of their main advantages.

It is important to check both aspects (see Lindsay's talk)

Warning: the uncertainty estimates from a GP can be flawed. Note that given data $D = X, y$

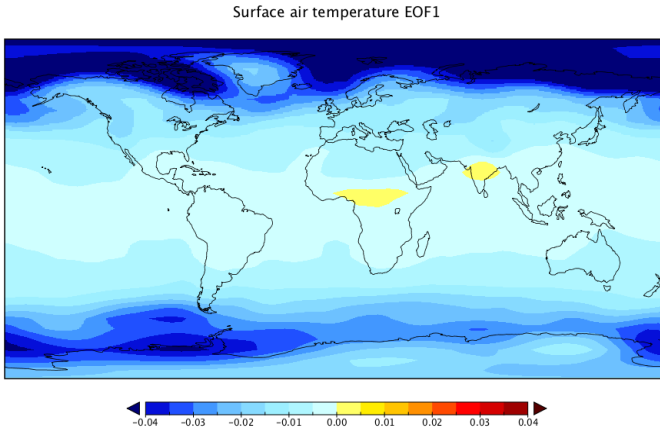
$$\mathbb{V}\text{ar}(f(x)|X, y) = k(x, x) - k(x, X)k(X, X)^{-1}k(X, x)$$

so that the posterior variance of $f(x)$ does not depend upon y!

The variance estimates are particularly sensitive to the hyper-parameter estimates.

Example 1: Easier regression

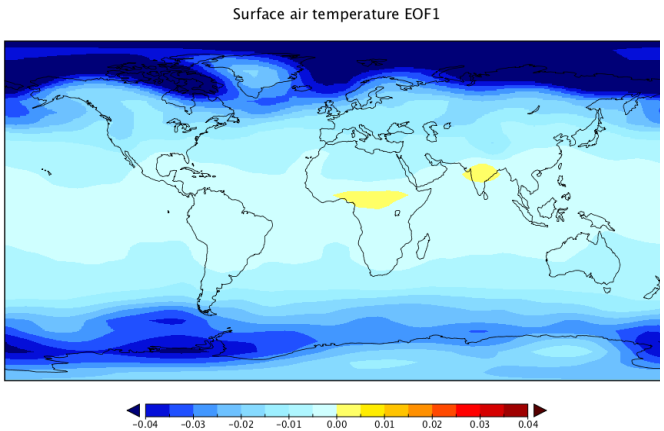
PLASIM-ENTS: Holden, Edwards, Garthwaite, W 2015



Emulate spatially resolved precipitation as a function of astronomical parameters: eccentricity, precession, obliquity.

Example 1: Easier regression

PLASIM-ENTS: Holden, Edwards, Garthwaite, W 2015

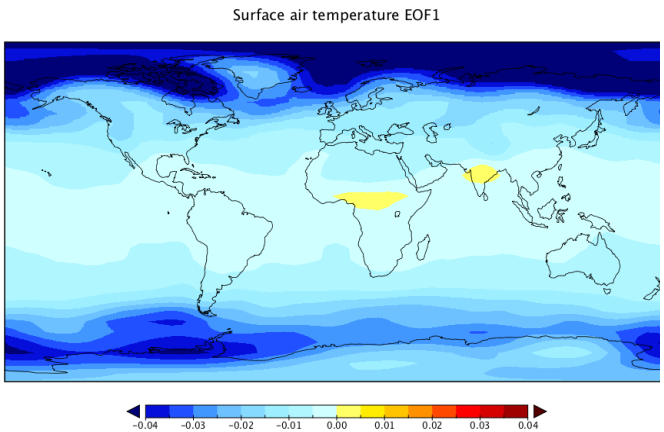


Emulate spatially resolved precipitation as a function of astronomical parameters: eccentricity, precession, obliquity.

- Using a linear regression emulator (on the EOFs/principal components), selecting terms using stepwise regression etc, we got an accuracy of 63%.

Example 1: Easier regression

PLASIM-ENTS: Holden, Edwards, Garthwaite, W 2015

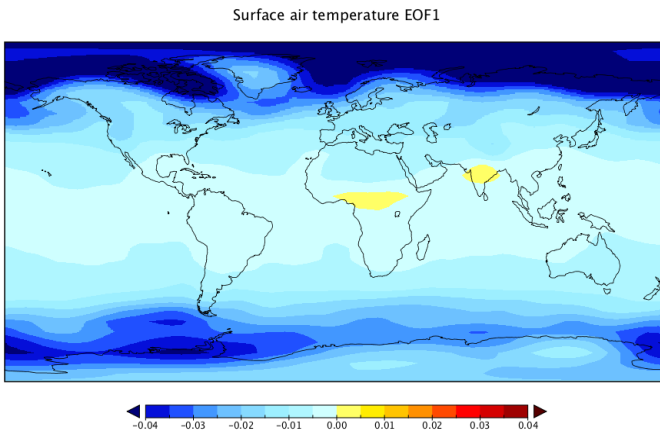


Emulate spatially resolved precipitation as a function of astronomical parameters: eccentricity, precession, obliquity.

- Using a linear regression emulator (on the EOFs/principal components), selecting terms using stepwise regression etc, we got an accuracy of 63%.
- After much thought and playing around, we realised we could improve the accuracy by using trigonometric transformations of the inputs. This gave an accuracy of 81%.

Example 1: Easier regression

PLASIM-ENTS: Holden, Edwards, Garthwaite, W 2015



Emulate spatially resolved precipitation as a function of astronomical parameters: eccentricity, precession, obliquity.

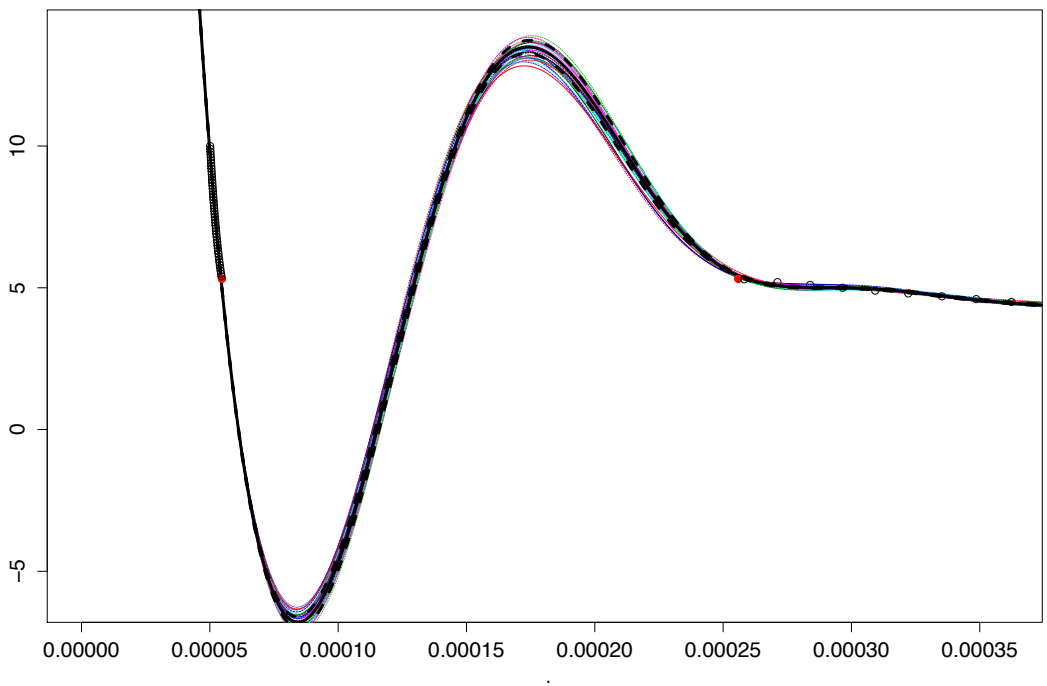
- Using a linear regression emulator (on the EOFs/principal components), selecting terms using stepwise regression etc, we got an accuracy of 63%.
- After much thought and playing around, we realised we could improve the accuracy by using trigonometric transformations of the inputs. This gave an accuracy of 81%.
- A GP gave us 82% accuracy (straight out of the box) with no need for transformations.

Example 2: Estimating gas laws for CCS

Cresswell, Wheatley, W., Graham 2016

$PV = nRT$ is an idealised law that holds in the limit.

- it doesn't apply when the gas is near its critical point
- gasses are most easily transported in the super-critical region.
- Impurities in the CO_2 (SO_2 etc) change the fluid behaviour.
- We only have a few measurements of fluid behaviour for impure CO2.



$$\int_{v_l}^{v_g} P(v)dv = P_s(v_g - v_l)$$

$$\text{and } \frac{\partial P}{\partial v}| = \frac{\partial P^2}{\partial v^2}| = 0$$

at $P = P_c, T = T_c$. By incorporating this information we were able to make more accurate predictions.

Example 3: Symmetry

Suppose we are modelling a function that is invariant under the single permutation σ, where $\sigma^2 = e$, e.g.,

$$f(x_1, x_2) = f(x_2, x_1)$$

Example 3: Symmetry

Suppose we are modelling a function that is invariant under the single permutation σ, where $\sigma^2 = e$, e.g.,

$$f(x_1, x_2) = f(x_2, x_1)$$

If we assume

$$f(x_1, x_2) = g(x_1, x_2) + g(x_2, x_1)$$

for some arbitrary function g, then f has the required symmetry.

Example 3: Symmetry

Suppose we are modelling a function that is invariant under the single permutation σ, where $\sigma^2 = e$, e.g.,

$$f(x_1, x_2) = f(x_2, x_1)$$

If we assume

$$f(x_1, x_2) = g(x_1, x_2) + g(x_2, x_1)$$

for some arbitrary function g, then f has the required symmetry.

If we model $g(\cdot) \sim GP(0, k(\cdot, \cdot))$, then the covariance function for f is

$$k_f = \mathbb{C}\text{ov}(f(x), f(x')) = k(x, x') + k(\sigma x, x') + k(x, \sigma x') + k(\sigma x, \sigma x')$$

Example 3: Symmetry

Suppose we are modelling a function that is invariant under the single permutation σ, where $\sigma^2 = e$, e.g.,

$$f(x_1, x_2) = f(x_2, x_1)$$

If we assume

$$f(x_1, x_2) = g(x_1, x_2) + g(x_2, x_1)$$

for some arbitrary function g, then f has the required symmetry.

If we model $g(\cdot) \sim GP(0, k(\cdot, \cdot))$, then the covariance function for f is

$$k_f = \mathbb{C}\text{ov}(f(x), f(x')) = k(x, x') + k(\sigma x, x') + k(x, \sigma x') + k(\sigma x, \sigma x')$$

If k is an isotropic kernel (we only actually require isotropy for each pair of vertices that swap in σ), then $k(x, x') = k(\sigma x, \sigma x')$ and $k(x, \sigma x') = k(\sigma x, x')$ as swaps only occur in pairs ($\sigma^2 = e$). So we can use

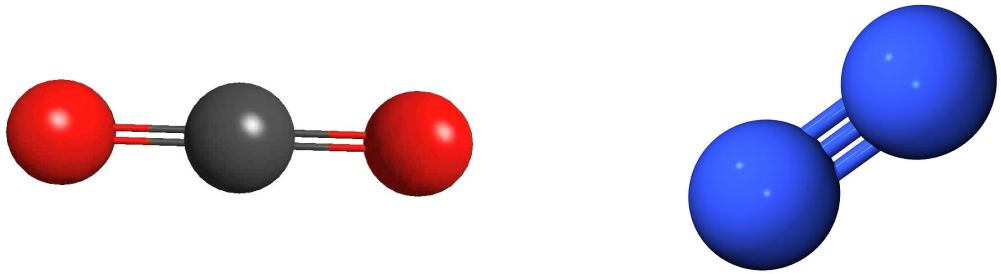
$$k_f(x, x') = k(x, x') + k(\sigma x, x')$$

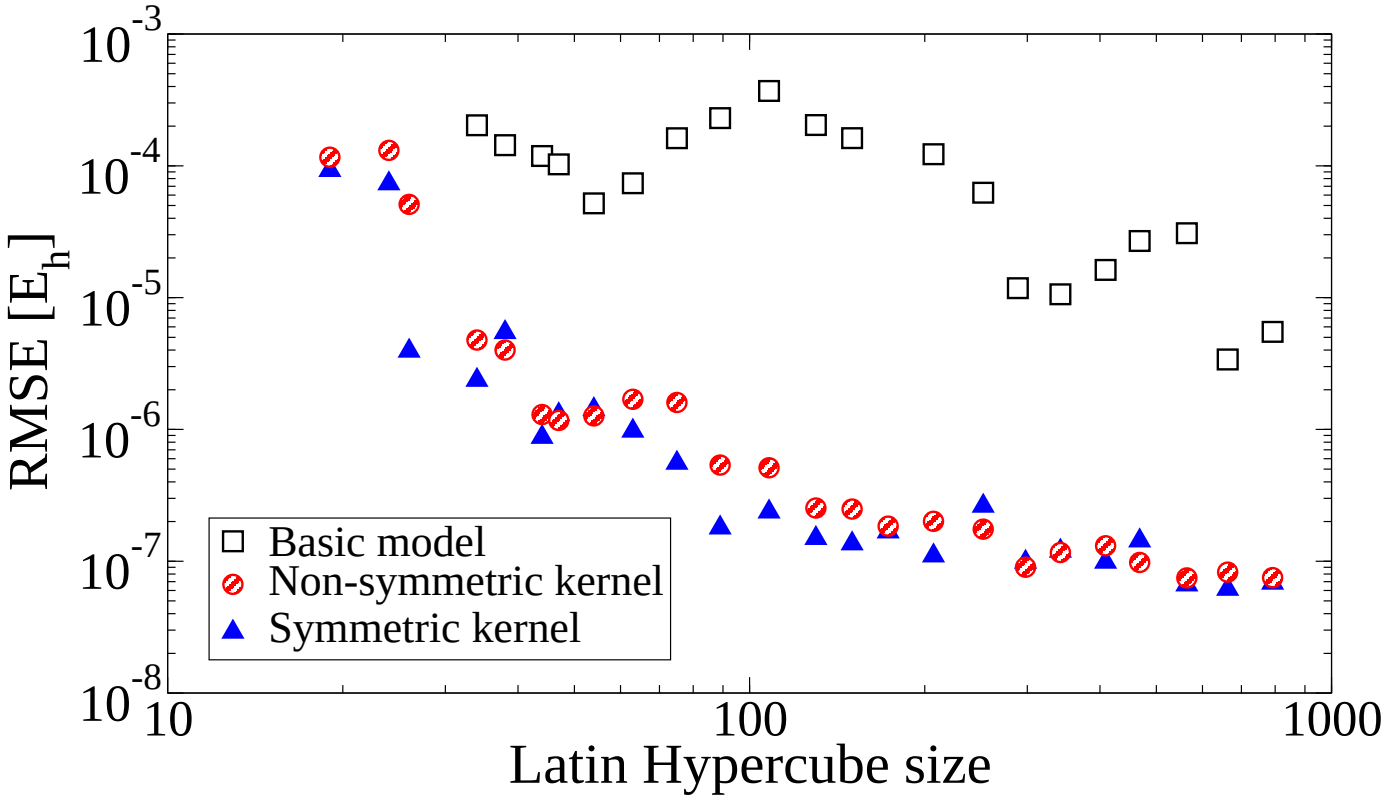
saving half the computation.

Example 3: Modelling intermolecular potentials: Ne-CO2

Uteva, Graham, W, Wheatley 2017

1294 cm-1



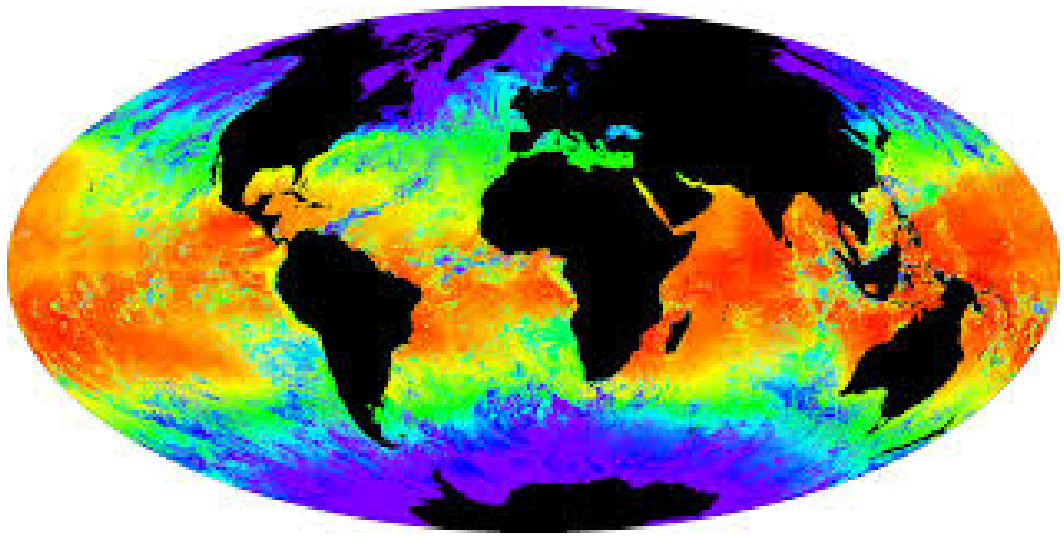


SPDE-INLA: Beyond GPs

Lindgren, Rue, Lindström 2011

The GP viewpoint is somewhat limited in that it relies upon us specifying a positive definite covariance function.

How can we build boutique covariance functions? E.g. emulating SST



The SPDE-INLA approach of Lindgren, Rue, Lindström shows how any Gauss Markov random field (somewhat like a GP) can be written as the solution to a SPDE, which we can solve on a finite mesh.

This gives us more modelling power, but at the cost of much more complex mathematics/algorithms.

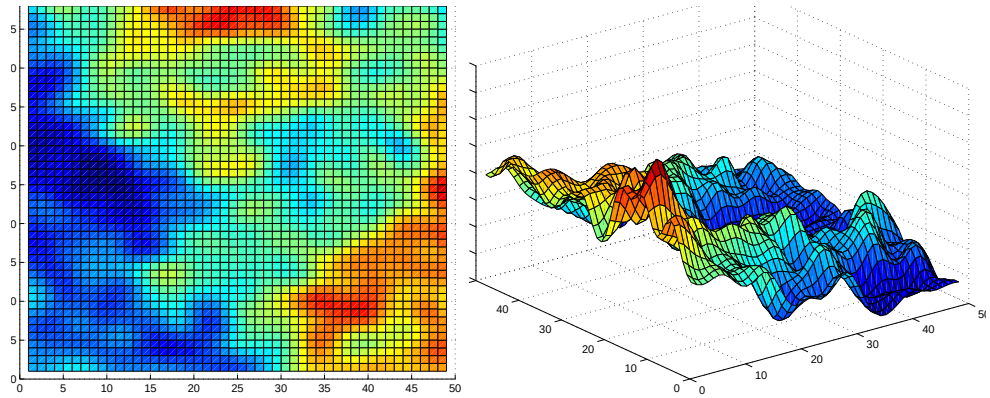
High dimensional problems

Carbon capture and storage

Knowledge of the physical problem is encoded in a simulator f

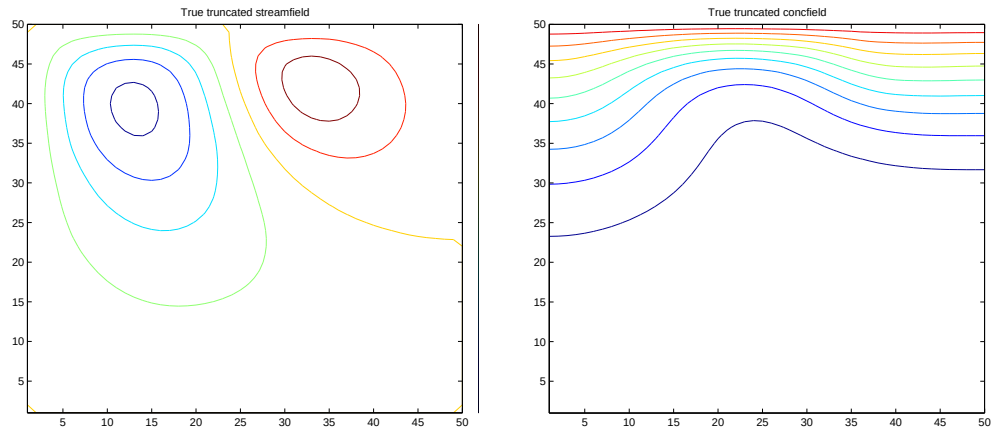
Inputs:

Permeability field, K
(2d field)



$\downarrow f(K)$

$f(K)$



Outputs:

Stream func. (2d field),
concentration (2d field),
surface flux (1d scalar),
$\vdots$

Surface Flux$= 6.43, \ldots$

Uncertainty quantification (UQ) for CCS

The simulator maps from permeability field K to outputs such as the surface flux $\mathcal{S}$. Let $f(K)$ denote this mapping

$$f : K \rightarrow \mathcal{S}$$

For most problems the permeability K is unknown.

Uncertainty quantification (UQ) for CCS

The simulator maps from permeability field K to outputs such as the surface flux $\mathcal{S}$. Let $f(K)$ denote this mapping

$$f : K \to \mathcal{S}$$

For most problems the permeability K is unknown.

If we assume a distribution for $K \sim \pi(K)$, we can quantify our uncertainty about $\mathcal{S} = f(K)$.

- e.g., by finding the cumulative distribution function (CDF) of $\mathcal{S}$:

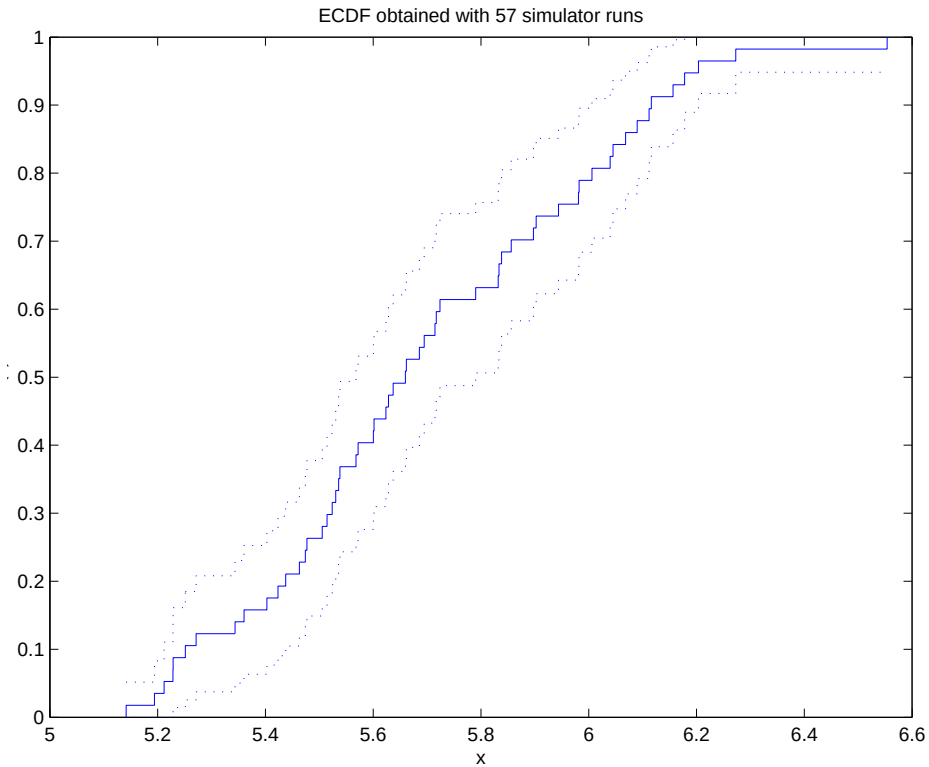
$$F(s) = \mathbb{P}(f(K) \leq s)$$

UQ for complex computer models

Gold standard approach: Monte Carlo simulation

- Draw $K_1, \ldots, K_N \sim \pi(K)$, and evaluate the simulator at each giving fluxes
  $s_1 = f(K_1), \ldots, s_N = f(K_N)$
- Estimate the empirical CDF

$$\widehat{F}(s) = \frac{1}{N} \sum_{i=1}^{N} \mathbb{I}_{s_i \leq s}$$

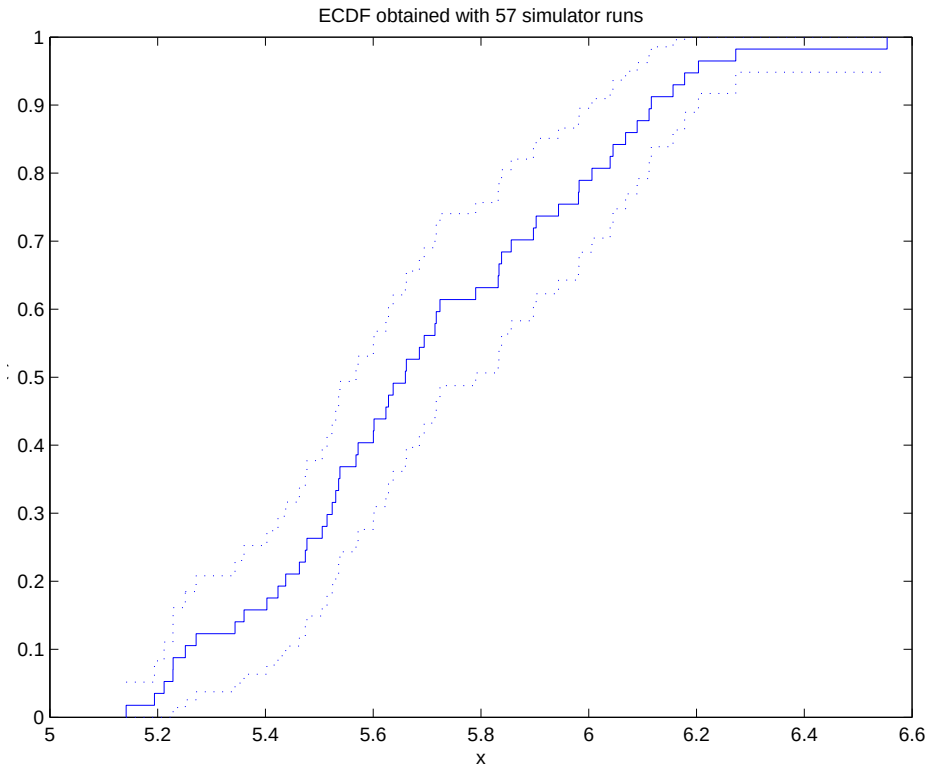


UQ for complex computer models

Gold standard approach: Monte Carlo simulation

- Draw $K_1, \ldots, K_N \sim \pi(K)$, and evaluate the simulator at each giving fluxes
  $s_1 = f(K_1), \ldots, s_N = f(K_N)$
- Estimate the empirical CDF

$$\widehat{F}(s) = \frac{1}{N} \sum_{i=1}^{N} \mathbb{I}_{s_i \leq s}$$



Note that $N = 10^3$ is not large if we want quantiles in the tail of the distribution

However the cost of the simulator means we are limited to $\sim$100 evaluations.

Multivariate Emulation

Wilkinson 2010

How can we deal with multivariate ouput?

- Build independent or separable multivariate emulators,
- Linear model of coregionalization?

Multivariate Emulation

Wilkinson 2010

How can we deal with multivariate ouput?

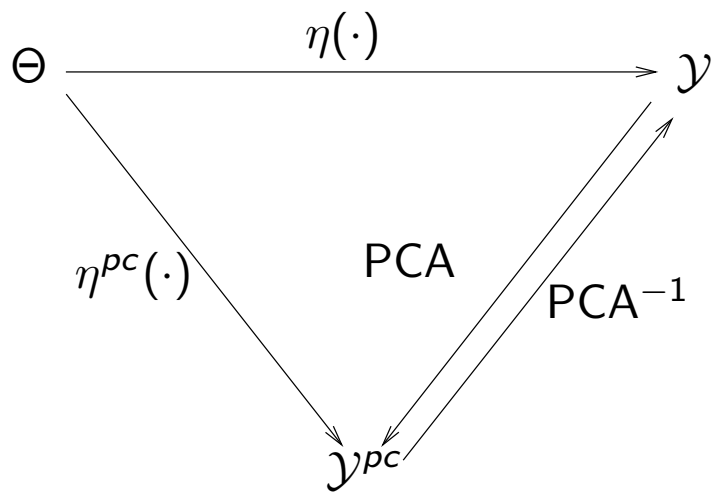
- Build independent or separable multivariate emulators,
- Linear model of coregionalization?

Instead, if the outputs are highly correlated we can reduce the dimension of the data by projecting the data into some lower dimensional space $\mathcal{Y}^{pc}$, i.e., assume

$$y = Wy^{pc} + e$$

where $dim(y) >> dim(y^{pc})$

Emulate from Θ to the reduced dimensional output space $\mathcal{Y}^{pc}$



Principal Component Emulation (EOF)

1. Find the singular value decomposition of Y.

$$Y = U\Gamma V^*.$$

Γ contains the singular values (sqrt of the eigenvalues), and V the principal components (eigenvectors of $Y^\top Y$).

Principal Component Emulation (EOF)

1. Find the singular value decomposition of Y.

$$Y = U\Gamma V^*.$$

   Γ contains the singular values (sqrt of the eigenvalues), and V the principal components (eigenvectors of $Y^\top Y$).

2. Decide on the dimension of the principal subspace, n^* say, and throw away all but the n^* leading principal components. An orthonormal basis for the principal subspace is given by the first n^* columns of V, denoted V_1. Let V_2 be the matrix of discarded columns.

Principal Component Emulation (EOF)

1. Find the singular value decomposition of Y.

$$Y = U\Gamma V^*.$$

   Γ contains the singular values (sqrt of the eigenvalues), and V the principal components (eigenvectors of $Y^\top Y$).

2. Decide on the dimension of the principal subspace, n^* say, and throw away all but the n^* leading principal components. An orthonormal basis for the principal subspace is given by the first n^* columns of V, denoted V_1. Let V_2 be the matrix of discarded columns.

3. Project Y onto the principal subspace to find $Y^{pc} = YV_1$

Principal Component Emulation (EOF)

1. Find the singular value decomposition of Y.

$$Y = U\Gamma V^*.$$

   Γ contains the singular values (sqrt of the eigenvalues), and V the principal components (eigenvectors of $Y^\top Y$).
2. Decide on the dimension of the principal subspace, n^* say, and throw away all but the n^* leading principal components. An orthonormal basis for the principal subspace is given by the first n^* columns of V, denoted V_1. Let V_2 be the matrix of discarded columns.
3. Project Y onto the principal subspace to find $Y^{pc} = YV_1$

Why use PCA here?

- The n directions are chosen to maximize the variance captured
- The approximation is the best possible rank n approximation in terms of minimizing the reconstruction error (Frobenius/2-norm)

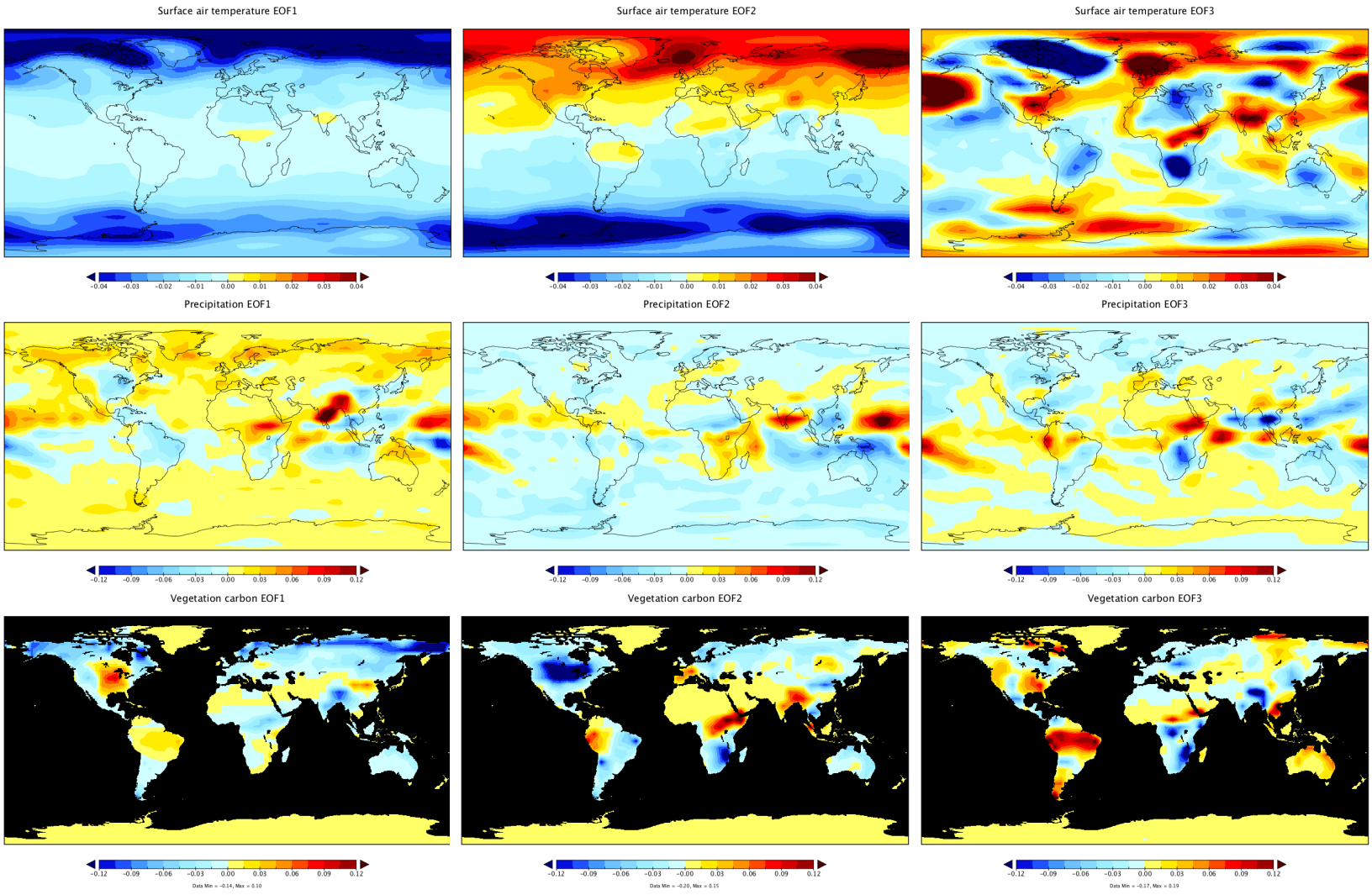
PLASIM-ENTS

Holden, Edwards, Garthwaite, Wilkinson 2015

- Planet Simulator coupled to the terrestrial carbon model ENTS
- Inputs are eccentricity, obliquity, precession describing Earth's orbit around the sun.
- Model climate (annual average surface temperature and rainfall) and vegetation (annual average vegetation carbon density) spatial fields (on a 64×32) grid.

We used an ensemble of 50 simulations

Principal components



PCA emulation

We then emulate the reduced dimension model

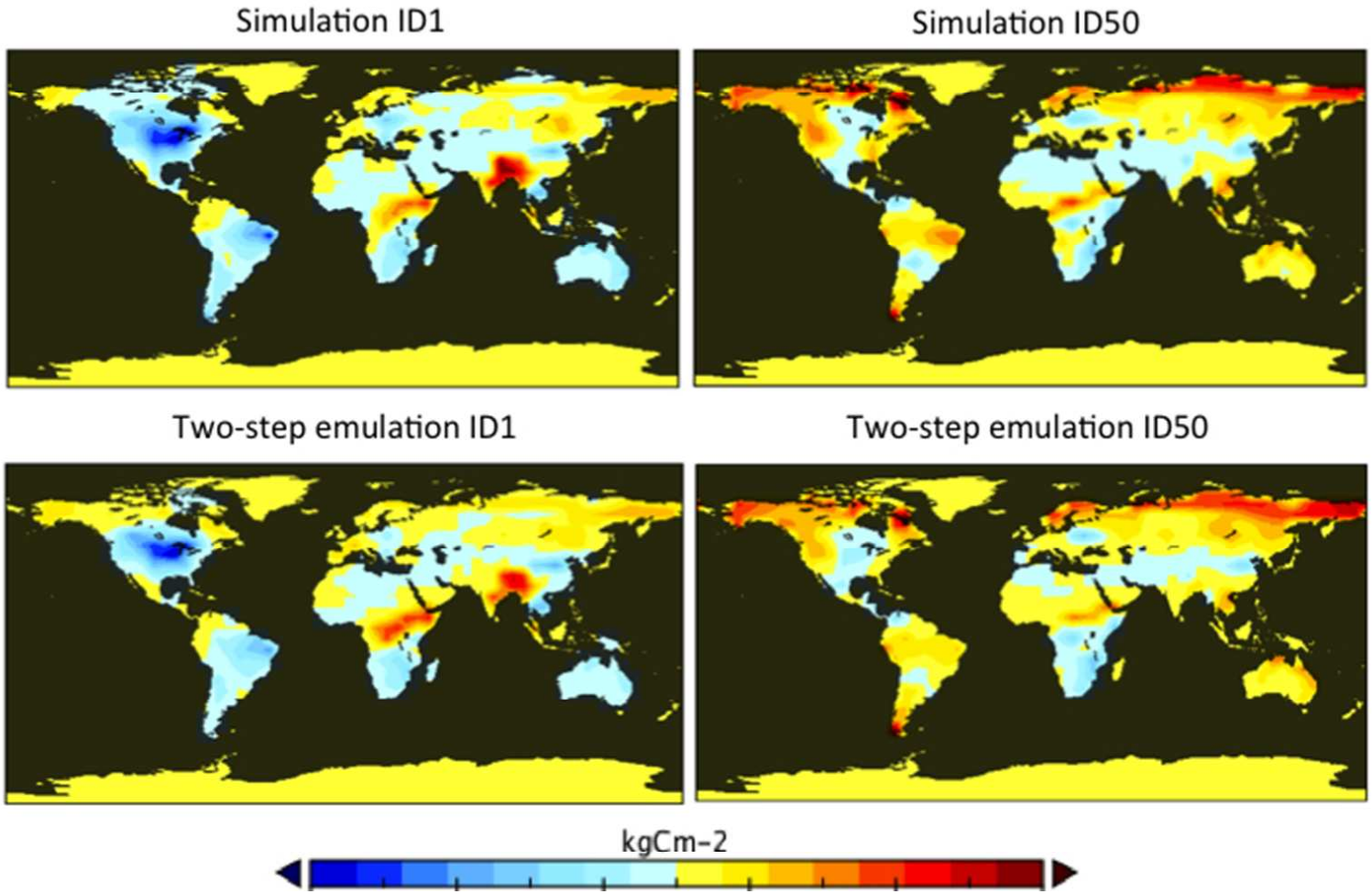
$$\eta_{pc}(\cdot) = (\eta_{pc}^1(\cdot), \ldots, \eta_{pc}^{n^*}(\cdot)).$$

- Each component η_{pc}^i will be uncorrelated (in the ensemble) but not necessarily independent. We use independent Gaussian processes for each component.
- The output can be reconstructed (accounting for reconstruction error) by modelling the discarded components as Gaussian noise with variance equal to the corresponding eigenvalue:

$$\eta(\theta) = V_1 \eta_{pc}(\theta) + V_2 \text{diag}(\Lambda)$$

where $\Lambda_i \sim N(0, \Gamma_{ii})$ ($\Gamma_{ii} = i^{th}$ eigenvalue).

Leave-one-out cross validation of the emulator



We can then use the PC-emulator to do sensitivity analysis.

Comments

- This approach (PCA emulation) requires that the outputs are highly correlated.
- We are assuming that the output $\mathcal{D}_{\text{sim}}$ is really a linear combination of a smaller number of variables,

$$\eta(\theta) = \mathbf{v}_1 \eta_{pc}^1(\theta) + \ldots + \mathbf{v}_{n^*} \eta_{pc}^{n^*}(\theta)$$

  which may be a reasonable assumption in many situations, eg, temporal spatial fields.
- Although PCA is a linear method (we could use kernel-PCA instead), the method can be used on highly non-linear models as we are still using non-linear Gaussian processes to map from Θ to $\mathcal{Y}^{pc}$ – the linear assumption applies only to the dimension reduction (and can be generalised).
- The method accounts for the reconstruction error from reducing the dimension of the data.

Emulating simulators with high dimensional input

Crevilln-Garca, W., Shah, Power, 2016

For the CCS simulator, the input is a permeability field which only needs to be known at a finite but large number of locations,

- e.g. if we use a 100×100 grid in the solver, K contains 10^4 entries
- Impossible to directly model $f : \mathbb{R}^{10,000} \to \mathbb{R}$

Emulating simulators with high dimensional input

Crevilln-Garca, W., Shah, Power, 2016

For the CCS simulator, the input is a permeability field which only needs to be known at a finite but large number of locations,

- e.g. if we use a 100×100 grid in the solver, K contains 10^4 entries
- Impossible to directly model $f : \mathbb{R}^{10,000} \to \mathbb{R}$

We can use the same idea to reduce the dimension of the inputs. However, because we know the distribution of K, it is more efficient to use the Karhunen-Loève (KL) expansion of K (rather than learn it empirically as in PCA)

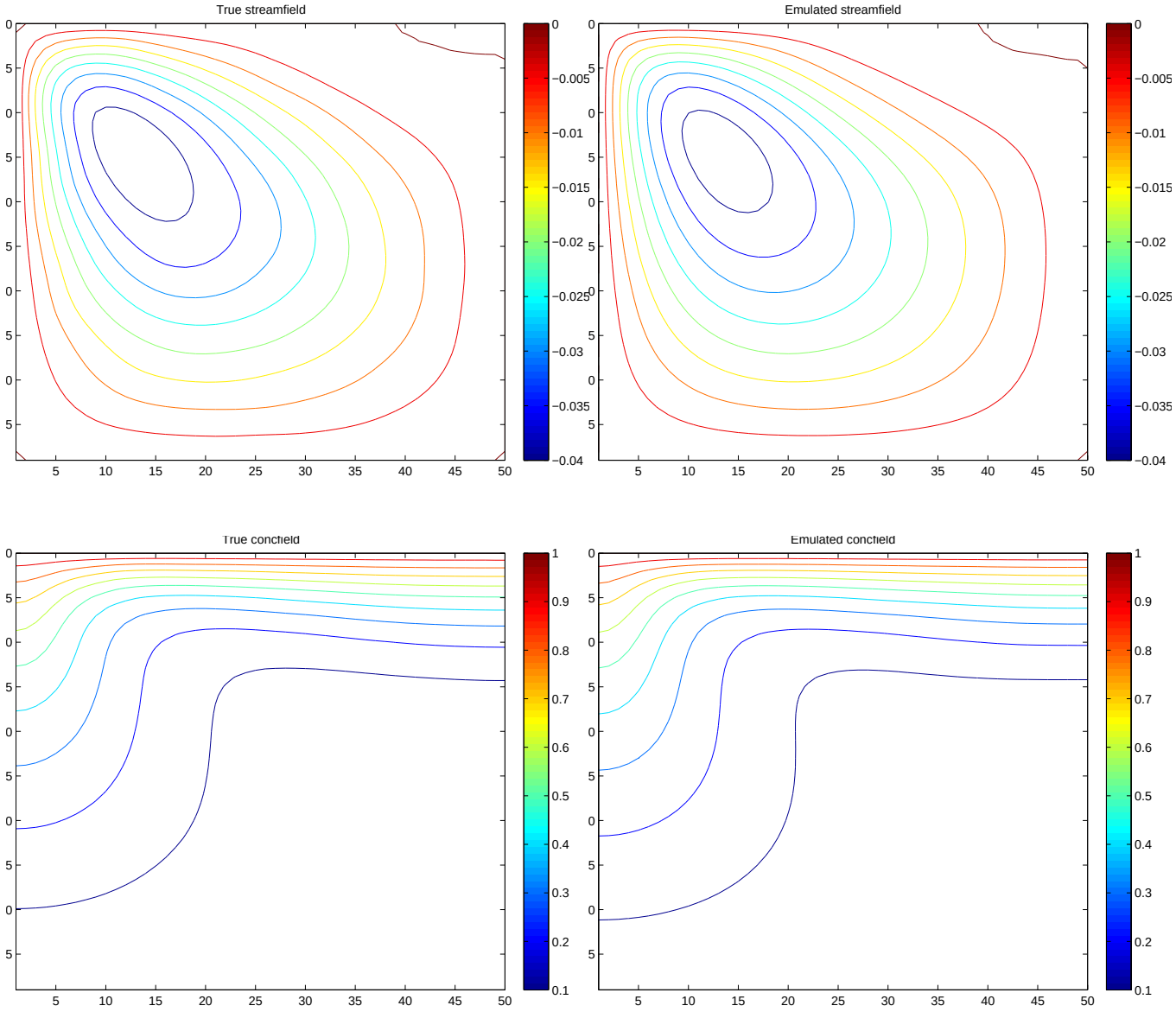
- $K = \exp(Z)$ where $Z \sim GP(m, C)$
- Z can be represented as

$$Z(\cdot) = \sum_{i=1}^{\infty} \lambda_i \xi_i \phi_i(\cdot)$$

where λ_i and ϕ_i are the eigenvalues and eigenfunctions of the covariance function of Z and $\xi_i \sim N(0, 1)$.

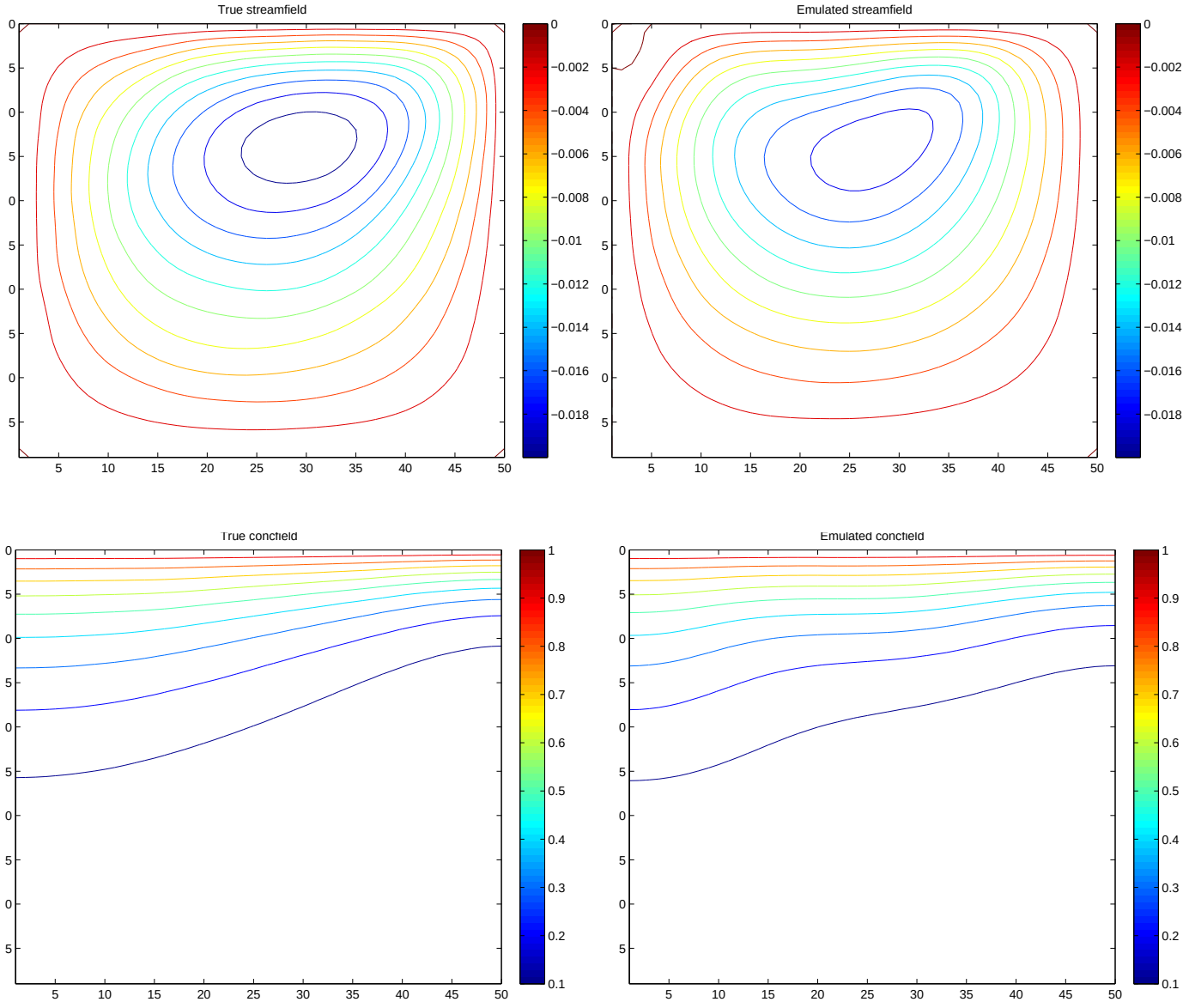
Emulating the stream function and concentration fields

Left=true, right = emulated, 118 training runs, held out test set.



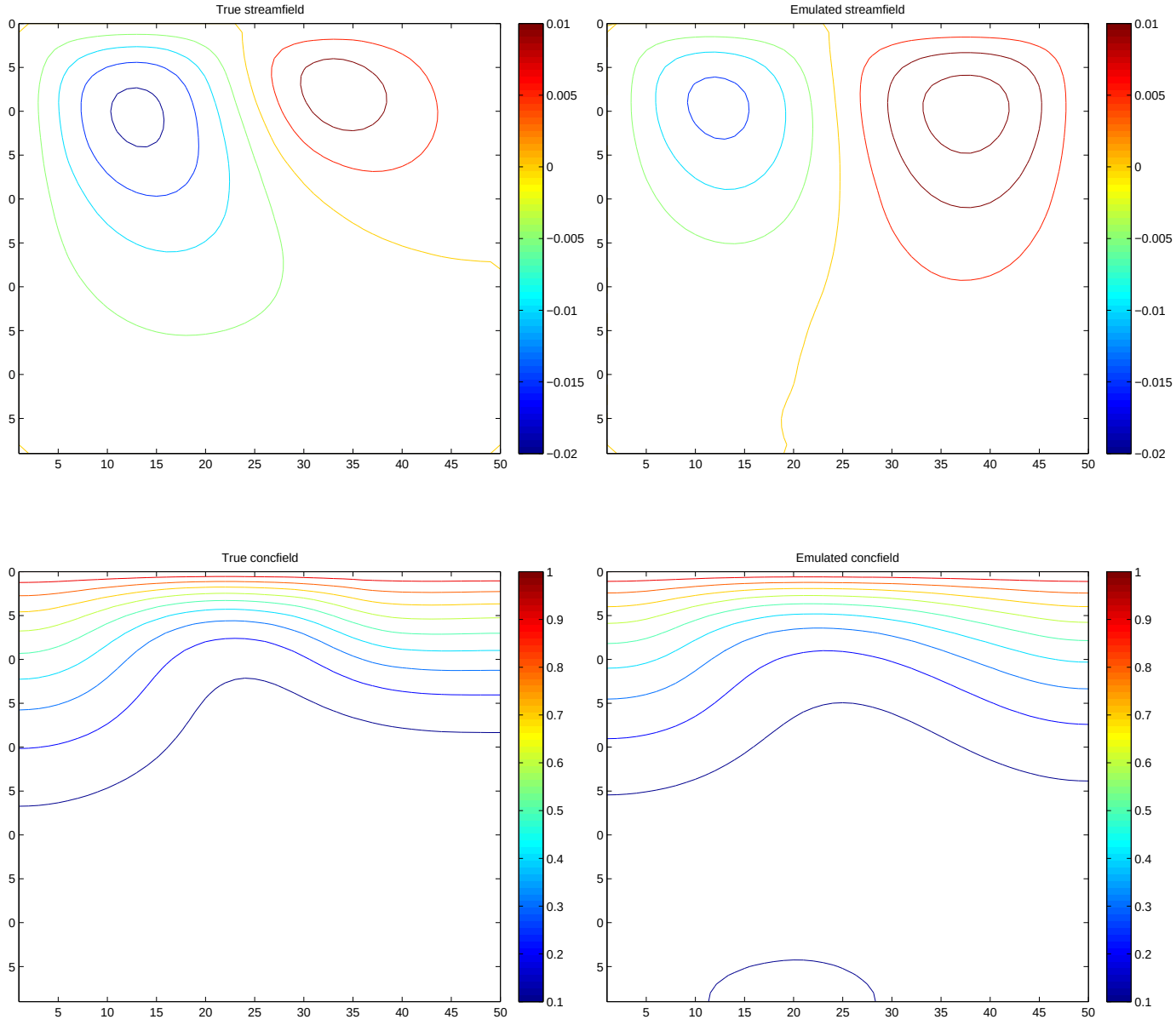
Emulating the stream function and concentration fields

Left=true, right = emulated, 118 training runs, held out test set.



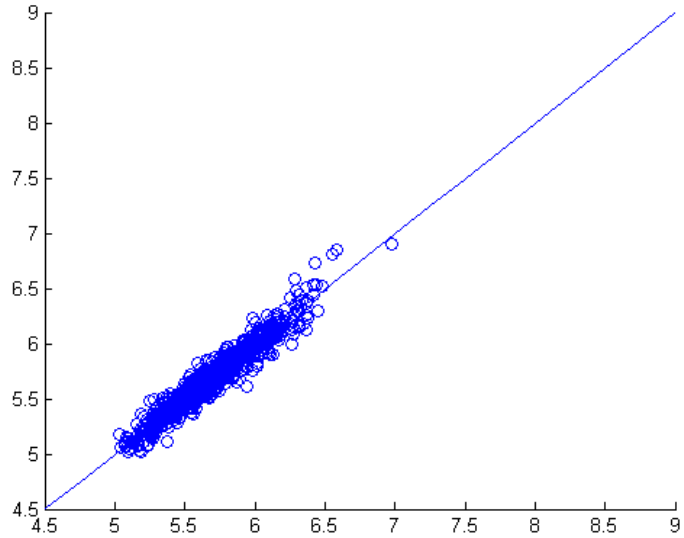
Emulating the stream function and concentration fields

Left=true, right = emulated, 118 training runs, held out test set.

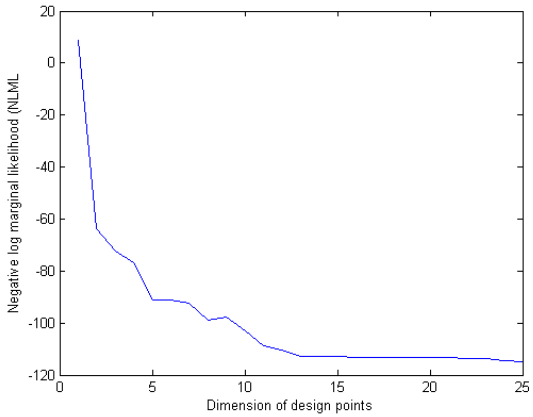


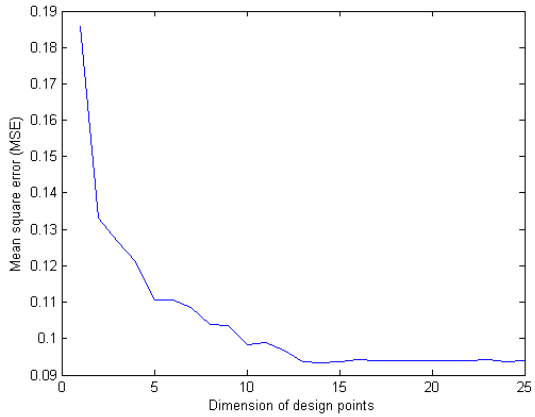
Predictive performance vs n = no. of KL components

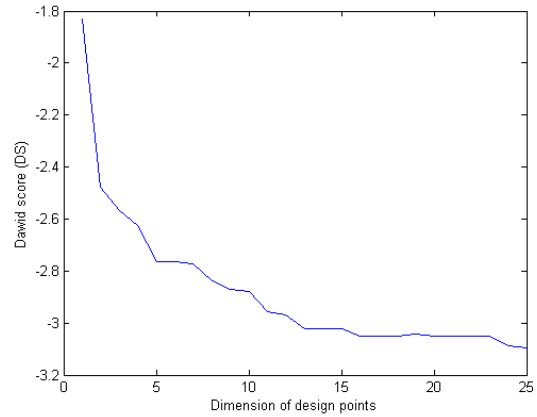
We can assess the accuracy of the emulator by examining the prediction error on a held-out test set. Plotting predicted vs true value indicates the accuracy the GP emulator.



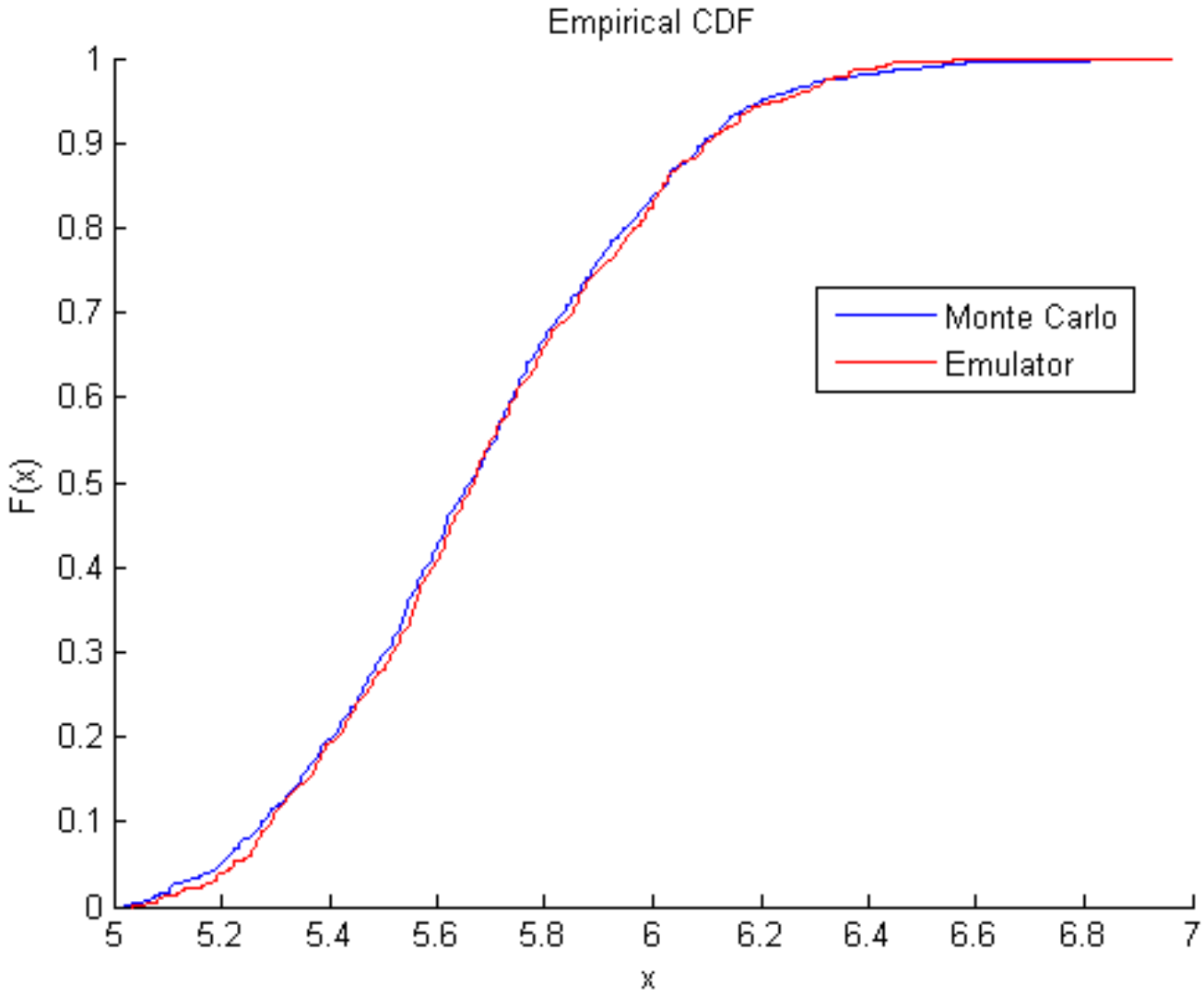
We can also choose the number of KL components to retain using numerical scores







CCS simulator results - 20 simulator training runs



Blue line = CDF from using 10^3 Monte Carlo samples from the simulator
Red line = CDF obtained using emulator (trained with 20 simulator runs, rational quadratic covariance function)

Comments

- The optimal output dimension reduction method is probably something like PCA, at least if what we care about is building a good global emulator.

Comments

- The optimal output dimension reduction method is probably something like PCA, at least if what we care about is building a good global emulator.
- PCA may be a poor dimension reduction for the inputs.
- Mathews and Vial 2017 describe a very interesting new approach for optimal dimension reduction when

$$d = f(x) \quad y = g(x)$$

where d are the observations, x the unknown (high dimensional) field, and y the quantity you want to predict.

Comments

- The optimal output dimension reduction method is probably something like PCA, at least if what we care about is building a good global emulator.
- PCA may be a poor dimension reduction for the inputs.
- Mathews and Vial 2017 describe a very interesting new approach for optimal dimension reduction when

$$d = f(x) \quad y = g(x)$$

  where d are the observations, x the unknown (high dimensional) field, and y the quantity you want to predict.
- There is a trade-off in the dimension reduction.
  - The more we reduce the dimension of the input the easier the regression becomes, but we lose more info in the compression.
  - Less dimension reduction leads to less information loss, but the regression becomes harder.

Comments

- The optimal output dimension reduction method is probably something like PCA, at least if what we care about is building a good global emulator.
- PCA may be a poor dimension reduction for the inputs.
- Mathews and Vial 2017 describe a very interesting new approach for optimal dimension reduction when

$$d = f(x) \quad y = g(x)$$

  where d are the observations, x the unknown (high dimensional) field, and y the quantity you want to predict.
- There is a trade-off in the dimension reduction.
  - The more we reduce the dimension of the input the easier the regression becomes, but we lose more info in the compression.
  - Less dimension reduction leads to less information loss, but the regression becomes harder.
- Using global sensitivity analysis to select the most influential inputs is a way of doing dimension reduction focused on the important information for regression. However, it is limited to projections onto the original coordinate axes.

Model discrepancy

An appealing idea

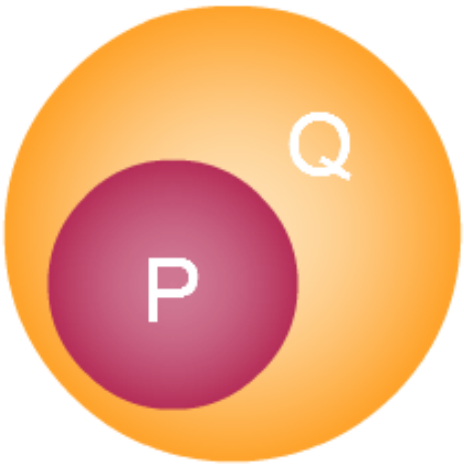
Kennedy and O'Hagan 2001

Lets acknowledge that most models are imperfect.

An appealing idea

Kennedy and O'Hagan 2001

Lets acknowledge that most models are imperfect.
Can we expand the class of models by adding a GP to our simulator?



If $f(x)$ is our simulator, d the observation, then perhaps we can correct f by modelling

$$y = f(x) + \delta(x) \quad \text{where} \quad \delta \sim GP$$

An appealing, but flawed, idea

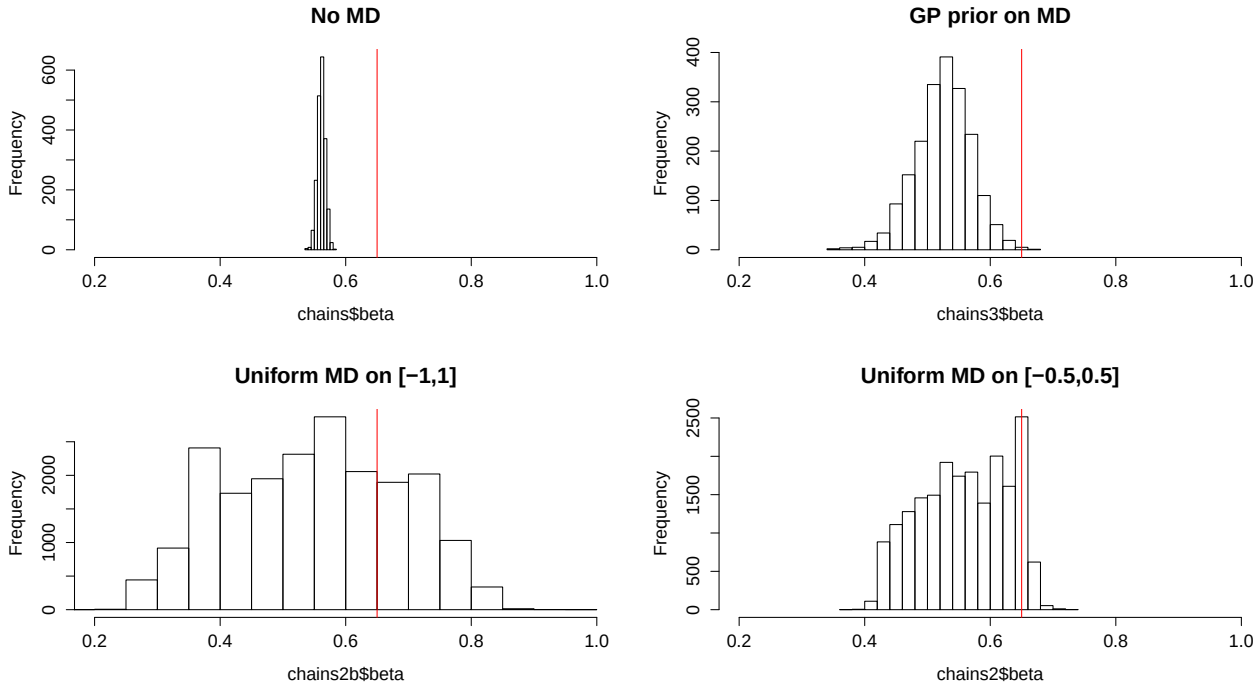
Kennedy and O'Hagan 2001, Brynjarsdottir and O'Hagan 2014

Simulator

$$f(x) = x\theta$$

Reality

$$g(x) = \frac{\theta x}{1 + \frac{x}{a}} \quad \theta = 0.65, a = 20$$



Bolting on a GP can correct your predictions, but won't necessarily fix your inference.

Design

Design

We build GPs using data $\{x_i, y_i\}_{i=1}^n$

- Call the collection $X_n = \{x_i\}_{i=1}^n \subset \mathbb{R}^d$ the **design**

For observational studies we have no control over the design, but we do for computer experiments!

- GP predictions made using a good design will be better than those using a poor design (Cf location of inducing points for sparse GPs)

What are we designing for?

- Global prediction
- Calibration
- Optimization - minimize the Expected Improvement (EI)?

Design for global prediction

e.g. Zhu and Stein 2006

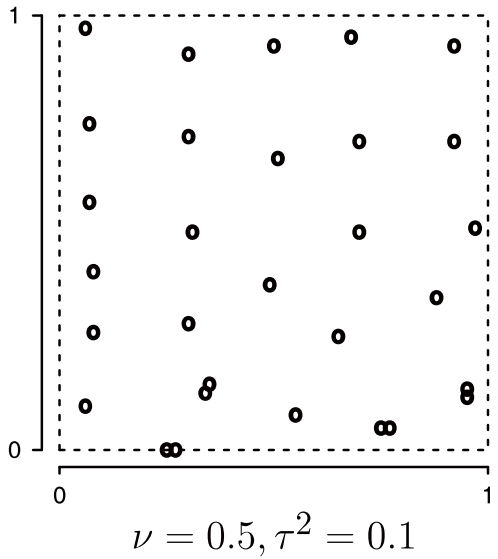
For a GP with known hyper parameters, space filling designs are good as the minimize the average prediction variance

- Latin hypercubes, maximin/minimax, max. entropy

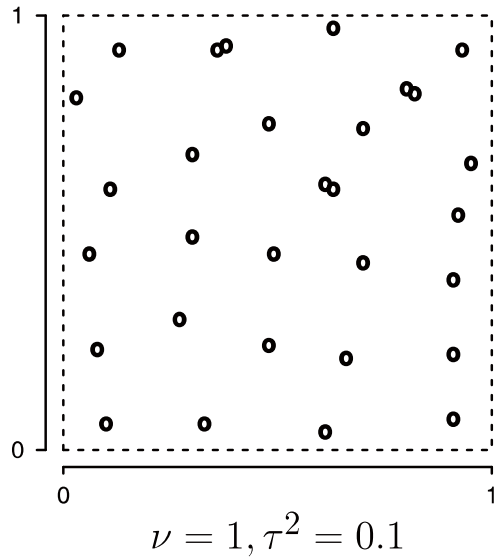
However, if we only want to estimate hyperparameters then maximize

$$\det \mathcal{I}(\theta) = -\det \mathbb{E}\left(\frac{\partial^2}{\partial \theta^2} f(X;\theta)\right)$$

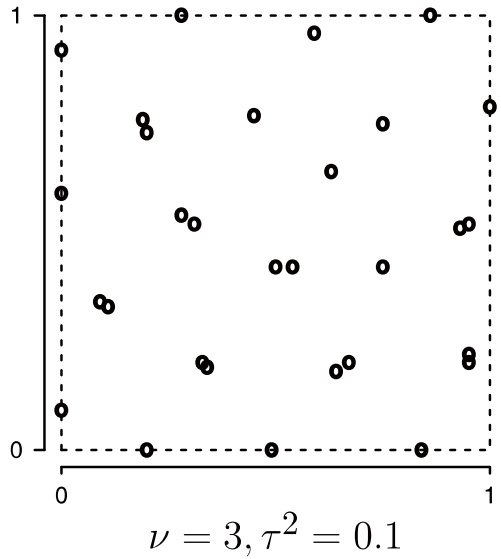
Usually, we want to make good predictions after having estimated parameters, and a trade-off between these two criteria has been proposed.



$\nu = 0.5, \tau^2 = 0.1$



$\nu = 1, \tau^2 = 0.1$



$\nu = 3, \tau^2 = 0.1$

Sequential design

The designs above are all 'one-shot' designs and can be wasteful. Instead we can use adaptive/sequential designs/active learning and add a point at a time:

- Choose location x_{n+1} to maximize some criterion/acquisition rule

$$C(x) \equiv C(x \mid \{x_i, y_i\}_{i=1}^n)$$

- Generate $y_{n+1} = f(x_{n+1})$

For optimization, we've seen that a good criterion for minimizing $f(x)$ is to choose x to maximize the expected improvement criterion

$$C(x) = \mathbb{E}[(\min_{i=1,\ldots,n} y_i - f(x))_+]$$

Sequential design for global prediction

Gramacy and Lee 2009, Beck and Guillas 2015

Many designs work on minimizing some function of the predictive variance/MSE

$$s_n^2(x) = \mathbb{V}\text{ar}(f(x)|D_n)$$

- Active learning MacKay (ALM): choose x at the point with largest predictive variance

$$C_n(x) = s_n^2(x)$$

Sequential design for global prediction

Gramacy and Lee 2009, Beck and Guillas 2015

Many designs work on minimizing some function of the predictive variance/MSE

$$s_n^2(x) = \mathbb{V}\text{ar}(f(x)|D_n)$$

- Active learning MacKay (ALM): choose x at the point with largest predictive variance

  $$C_n(x) = s_n^2(x)$$

  This tends to locate points on the edge of the domain.

Sequential design for global prediction

Gramacy and Lee 2009, Beck and Guillas 2015

Many designs work on minimizing some function of the predictive variance/MSE

$$s_n^2(x) = \mathbb{V}\text{ar}(f(x)|D_n)$$

- Active learning MacKay (ALM): choose x at the point with largest predictive variance

  $$C_n(x) = s_n^2(x)$$

  This tends to locate points on the edge of the domain.
- Active learning Cohn (ALC): choose x to give largest expected reduction in predictive variance

$$C_n(x) = \int s_n^2(x') - s_{n\cup x}^2(x')\mathrm{d}x'$$

Sequential design for global prediction

Gramacy and Lee 2009, Beck and Guillas 2015

Many designs work on minimizing some function of the predictive variance/MSE

$$s_n^2(x) = \mathbb{V}\text{ar}(f(x)|D_n)$$

- Active learning MacKay (ALM): choose x at the point with largest predictive variance

  $$C_n(x) = s_n^2(x)$$

  This tends to locate points on the edge of the domain.
- Active learning Cohn (ALC): choose x to give largest expected reduction in predictive variance

$$C_n(x) = \int s_n^2(x') - s_{n\cup x}^2(x')\mathrm{d}x'$$

ALC tends to give better designs than ALM, but has cost $O(n^3 + N_{ref}\,N_{cand}\,n^2)$ for each new design point

Sequential design for global prediction

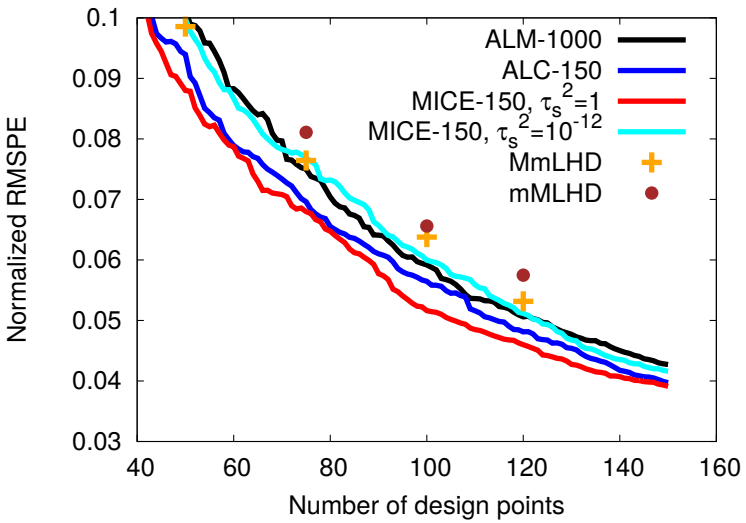
MICE: Beck and Guillas 2015

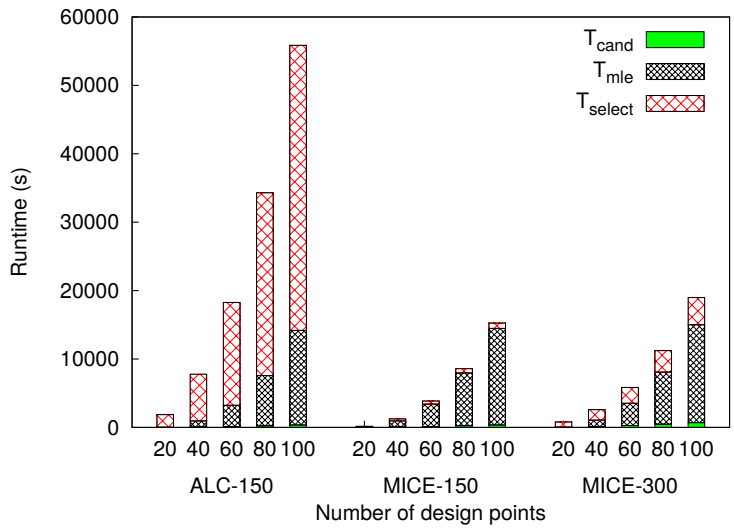
The Mutual Information between Y and Y' is

$$\mathcal{I}(Y; Y') = \mathcal{H}(Y) - \mathcal{H}(Y \mid Y') = KL(p_{y,y'} || p_y p_{y'})$$

Choose design X_n to maximize mutual information between $f(X_n)$ and $f(X_{cand} \setminus X_n)$ where X_{cand} is a set of candidate design points.
A sequential version for GPs reduces to choosing x to maximize

$$C_n(x) = \frac{s_n^2(x)}{s_{cand \setminus (n \cup x)}(x, \tau^2)}$$





Conclusions

Conclusions

You can do lots of stuff with GPs.

Conclusions

You can do lots of stuff with GPs.

Thank you for listening!