Known Boundary Emulation

Ian Vernon

Department of Mathematical Sciences, Durham University

Information and Statistics for Nuclear Experiment and Theory workshop (ISNET-9)

Joint work with: Sam Jackson, Jonathan Cumming.

Overview

- Incorporating Known Boundaries into a GP emulator.
- Designing runs in the presence of known boundaries.
- Application to Systems Biology model of Arabidopsis.

Motivation for use of GP Emulators

- I am a Bayesian Statistician, interested in using GP style emulation within various Uncertainty Quantification analyses (Model Calibration, History matching, decision support etc).

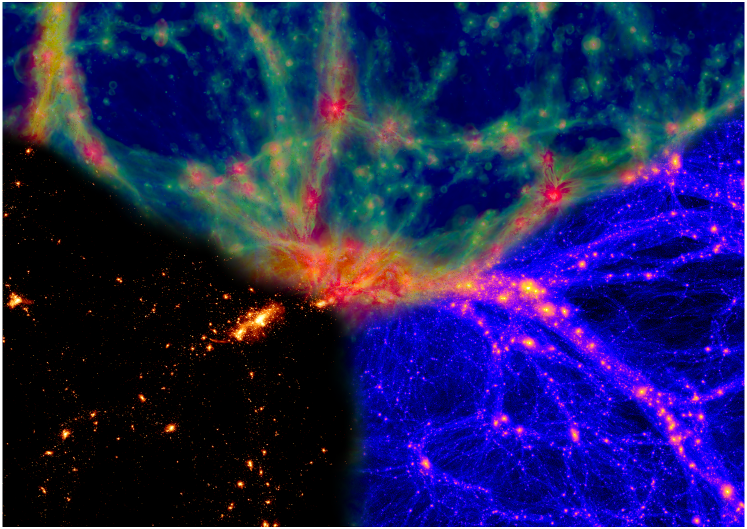
- GP emulators can mimic complex scientific models but are extremely fast (and provide uncertainties).

- We at Durham University, UK, have applied these emulators to multiple scientific areas including:

  - Cosmology: galaxy formation simulations, very expensive.
  - Epidemiology: large stochastic agent based models of HIV, Typhoid, TB, Covid, HPV.
  - Systems biology models: metabolic networks, plant root models.
  - Geology models.
  - Environmental models.
  - Climate models.

EAGLE Outputs

Gas Temperature



Visual spectrum

Dark Matter density

Emulation Basic Structure

- We consider an expensive computer model represented as a function $f(x) \in \mathbb{R}$, with d-dimensional input vector $x \in \mathcal{X}$, where $\mathcal{X} \subset \mathbb{R}^d$ is a pre-specified input parameter space of interest.
- We represent our beliefs about the unknown $f(x)$ at unevaluated input x via an emulator $u(x)$. This maybe a Gaussian process (or in a less fully specified version, a weakly second order stationary stochastic process):

$$f(x) = u(x)$$

- We make the judgement, consistent with most of the computer model literature, that the $u(x)$ have a product correlation structure:

$$\mathrm{Cov}[u(x), u(x')] = \sigma^2 r(x - x') = \sigma^2 \prod_{i=1}^{d} r_i(x_i - x_i')$$

with $r_i(0) = 1$, corresponding to deterministic $f(x)$.

Emulation Basic Structure

- We consider an expensive computer model represented as a function $f(x) \in \mathbb{R}$, with d-dimensional input vector $x \in \mathcal{X}$, where $\mathcal{X} \subset \mathbb{R}^d$ is a pre-specified input parameter space of interest.

- We represent our beliefs about the unknown $f(x)$ at unevaluated input x via an emulator $u(x)$. This maybe a Gaussian process (or in a less fully specified version, a weakly second order stationary stochastic process):

$$f(x) = u(x)$$

- We make the judgement, consistent with most of the computer model literature, that the $u(x)$ have a product correlation structure:

$$\mathrm{Cov}[u(x), u(x')] = \sigma^2 r(x - x') = \sigma^2 \prod_{i=1}^{d} r_i(x_i - x_i')$$

with $r_i(0) = 1$, corresponding to deterministic $f(x)$.

Emulation Basic Structure

- We consider an expensive computer model represented as a function $f(x) \in \mathbb{R}$, with d-dimensional input vector $x \in \mathcal{X}$, where $\mathcal{X} \subset \mathbb{R}^d$ is a pre-specified input parameter space of interest.
- We represent our beliefs about the unknown $f(x)$ at unevaluated input x via an emulator $u(x)$. This maybe a Gaussian process (or in a less fully specified version, a weakly second order stationary stochastic process):

$$f(x) \;=\; u(x)$$

- We make the judgement, consistent with most of the computer model literature, that the $u(x)$ have a product correlation structure:

$$\mathrm{Cov}[u(x), u(x')] \;=\; \sigma^2 r(x - x') \;=\; \sigma^2 \prod_{i=1}^{d} r_i(x_i - x_i')$$

with $r_i(0) = 1$, corresponding to deterministic $f(x)$.

Emulation Basic Structure

- Product correlation structures are very common, e.g. the Gaussian structure:

$$r(x - x') \;=\; \prod_{i=1}^{d} \exp\{-|x_i - x_i'|^2/\theta_i^2\}.$$

- As usual, we will also assume stationarity, but do not require it for the following.

- If we perform a set of runs at locations $x_D = (x^{(1)}, \ldots, x^{(n)})$ over $\mathcal{X}$, giving model outputs $D = (f(x^{(1)}), \ldots, f(x^{(n)}))^T$, then we can update our beliefs about the computer model $f(x)$ in light of D.

- This can be done either using Bayes theorem (if $u(x)$ is assumed to be a full Gaussian process) or using the Bayes linear update, which following DeFinetti, treats expectation as primitive, and requires only a second order specification:

$$\begin{aligned}
\mathrm{E}_D[f(x)] &= \mathrm{E}[f(x)] + \mathrm{Cov}[f(x), D]\mathrm{Var}[D]^{-1}(D - \mathrm{E}[D]) \\
\mathrm{Var}_D[f(x)] &= \mathrm{Var}[f(x)] - \mathrm{Cov}[f(x), D]\mathrm{Var}[D]^{-1}\mathrm{Cov}[D, f(x)] \\
\mathrm{Cov}_D[f(x), f(x')] &= \mathrm{Cov}[f(x), f(x')] - \mathrm{Cov}[f(x), D]\mathrm{Var}[D]^{-1}\mathrm{Cov}[D, f(x')]
\end{aligned}$$

where $\mathrm{E}_D[f(x)]$, $\mathrm{Var}_D[f(x)]$ and $\mathrm{Cov}_D[f(x), f(x')]$ are the expectation, variance and covariance of $f(x)$ adjusted by D.

Emulation Basic Structure

- Product correlation structures are very common, e.g. the Gaussian structure:

$$r(x - x') \;=\; \prod_{i=1}^{d} \exp\{-|x_i - x'_i|^2/\theta_i^2\}.$$

- As usual, we will also assume stationarity, but do not require it for the following.
- If we perform a set of runs at locations $x_D = (x^{(1)}, \dots, x^{(n)})$ over $\mathcal{X}$, giving model outputs $D = (f(x^{(1)}), \dots, f(x^{(n)}))^T$, then we can update our beliefs about the computer model $f(x)$ in light of D.
- This can be done either using Bayes theorem (if $u(x)$ is assumed to be a full Gaussian process) or using the Bayes linear update, which following DeFinetti, treats expectation as primitive, and requires only a second order specification:

$$\begin{aligned} \mathrm{E}_D[f(x)] &= \mathrm{E}[f(x)] + \mathrm{Cov}[f(x), D]\mathrm{Var}[D]^{-1}(D - \mathrm{E}[D]) \\ \mathrm{Var}_D[f(x)] &= \mathrm{Var}[f(x)] - \mathrm{Cov}[f(x), D]\mathrm{Var}[D]^{-1}\mathrm{Cov}[D, f(x)] \\ \mathrm{Cov}_D[f(x), f(x')] &= \mathrm{Cov}[f(x), f(x')] - \mathrm{Cov}[f(x), D]\mathrm{Var}[D]^{-1}\mathrm{Cov}[D, f(x')] \end{aligned}$$

where $\mathrm{E}_D[f(x)]$, $\mathrm{Var}_D[f(x)]$ and $\mathrm{Cov}_D[f(x), f(x')]$ are the expectation, variance and covariance of $f(x)$ adjusted by D.

Emulation Basic Structure

- Product correlation structures are very common, e.g. the Gaussian structure:

$$r(x - x') = \prod_{i=1}^{d} \exp\{-|x_i - x_i'|^2/\theta_i^2\}.$$

- As usual, we will also assume stationarity, but do not require it for the following.

- If we perform a set of runs at locations $x_D = (x^{(1)}, \ldots, x^{(n)})$ over $\mathcal{X}$, giving model outputs $D = (f(x^{(1)}), \ldots, f(x^{(n)}))^T$, then we can update our beliefs about the computer model $f(x)$ in light of D.

- This can be done either using Bayes theorem (if $u(x)$ is assumed to be a full Gaussian process) or using the Bayes linear update, which following DeFinetti, treats expectation as primitive, and requires only a second order specification:

$$\begin{aligned} \mathrm{E}_D[f(x)] &= \mathrm{E}[f(x)] + \mathrm{Cov}[f(x), D]\mathrm{Var}[D]^{-1}(D - \mathrm{E}[D]) \\ \mathrm{Var}_D[f(x)] &= \mathrm{Var}[f(x)] - \mathrm{Cov}[f(x), D]\mathrm{Var}[D]^{-1}\mathrm{Cov}[D, f(x)] \\ \mathrm{Cov}_D[f(x), f(x')] &= \mathrm{Cov}[f(x), f(x')] - \mathrm{Cov}[f(x), D]\mathrm{Var}[D]^{-1}\mathrm{Cov}[D, f(x')] \end{aligned}$$

where $\mathrm{E}_D[f(x)]$, $\mathrm{Var}_D[f(x)]$ and $\mathrm{Cov}_D[f(x), f(x')]$ are the expectation, variance and covariance of $f(x)$ adjusted by D.

Emulation Basic Structure

- Although we will work within the Bayes linear formalism, the derived results will apply directly to the fully Bayesian case, were one willing to make the additional assumption of full normality that use of a Gaussian process entails.
- In this case all Bayes linear adjusted quantities can be directly mapped to the corresponding posterior versions e.g.

$$\mathrm{E}_D[f(x)] \to \mathrm{E}[f(x)|D] \quad \text{and} \quad \mathrm{Var}_D[f(x)] \to \mathrm{Var}[f(x)|D].$$

- See Goldstein and Wooff (2007), for discussion of the benefits of using a Bayes linear approach.

Known Boundary Emulation: Motivation

- This is all fine, however, for some simulators, there exist input parameter settings, lying possibly on boundaries or hyperplanes in the input parameter space, where the simulator can be solved analytically (or just significantly faster).
- This may be due to the system in question, or at least a subset of the system outputs, behaving in a much simpler way for particular input settings.
- This is possibly due for example to various modules decoupling from more complex parts of the model (possibly when certain inputs are set to zero, switching some processes off).
- Note that this leads to Dirichlet boundary conditions, i.e. known simulator behaviour on various hyperplanes within $\mathcal{X}$, that impose constraints on the emulator itself, not the same as Dirichlet boundary conditions on the physical model (which we do not require here).
- The goal then, is to incorporate these known boundaries, situated where we essentially know the function output, into the Bayesian emulation process which should lead to significantly improved emulators.
- We do this by formally updating the emulators by the information contained on the known boundaries, obtaining analytic results, then updating by runs D as usual.

Known Boundary Emulation: Motivation

- This is all fine, however, for some simulators, there exist input parameter settings, lying possibly on boundaries or hyperplanes in the input parameter space, where the simulator can be solved analytically (or just significantly faster).
- This may be due to the system in question, or at least a subset of the system outputs, behaving in a much simpler way for particular input settings.
- This is possibly due for example to various modules decoupling from more complex parts of the model (possibly when certain inputs are set to zero, switching some processes off).
- Note that this leads to Dirichlet boundary conditions, i.e. known simulator behaviour on various hyperplanes within $\mathcal{X}$, that impose constraints on the emulator itself, not the same as Dirichlet boundary conditions on the physical model (which we do not require here).
- The goal then, is to incorporate these known boundaries, situated where we essentially know the function output, into the Bayesian emulation process which should lead to significantly improved emulators.
- We do this by formally updating the emulators by the information contained on the known boundaries, obtaining analytic results, then updating by runs D as usual.

Known Boundary Emulation: Motivation

- This is all fine, however, for some simulators, there exist input parameter settings, lying possibly on boundaries or hyperplanes in the input parameter space, where the simulator can be solved analytically (or just significantly faster).
- This may be due to the system in question, or at least a subset of the system outputs, behaving in a much simpler way for particular input settings.
- This is possibly due for example to various modules decoupling from more complex parts of the model (possibly when certain inputs are set to zero, switching some processes off).
- Note that this leads to Dirichlet boundary conditions, i.e. known simulator behaviour on various hyperplanes within $\mathcal{X}$, that impose constraints on the emulator itself, not the same as Dirichlet boundary conditions on the physical model (which we do not require here).
- The goal then, is to incorporate these known boundaries, situated where we essentially know the function output, into the Bayesian emulation process which should lead to significantly improved emulators.
- We do this by formally updating the emulators by the information contained on the known boundaries, obtaining analytic results, then updating by runs D as usual.

Known Boundary Emulation: Motivation

- This is all fine, however, for some simulators, there exist input parameter settings, lying possibly on boundaries or hyperplanes in the input parameter space, where the simulator can be solved analytically (or just significantly faster).
- This may be due to the system in question, or at least a subset of the system outputs, behaving in a much simpler way for particular input settings.
- This is possibly due for example to various modules decoupling from more complex parts of the model (possibly when certain inputs are set to zero, switching some processes off).
- Note that this leads to Dirichlet boundary conditions, i.e. known simulator behaviour on various hyperplanes within $\mathcal{X}$, that impose constraints on the emulator itself, not the same as Dirichlet boundary conditions on the physical model (which we do not require here).
- The goal then, is to incorporate these known boundaries, situated where we essentially know the function output, into the Bayesian emulation process which should lead to significantly improved emulators.
- We do this by formally updating the emulators by the information contained on the known boundaries, obtaining analytic results, then updating by runs D as usual.

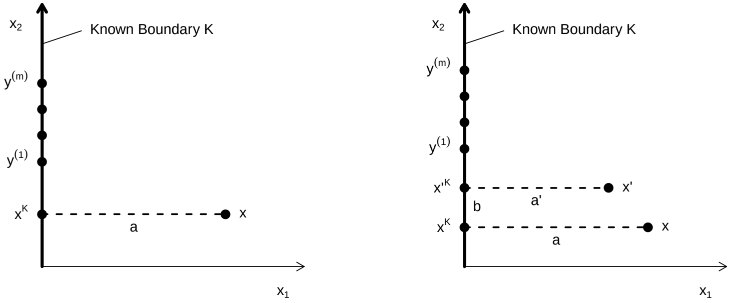
Known Boundary Emulation: Motivation

- This is all fine, however, for some simulators, there exist input parameter settings, lying possibly on boundaries or hyperplanes in the input parameter space, where the simulator can be solved analytically (or just significantly faster).
- This may be due to the system in question, or at least a subset of the system outputs, behaving in a much simpler way for particular input settings.
- This is possibly due for example to various modules decoupling from more complex parts of the model (possibly when certain inputs are set to zero, switching some processes off).
- Note that this leads to Dirichlet boundary conditions, i.e. known simulator behaviour on various hyperplanes within $\mathcal{X}$, that impose constraints on the emulator itself, not the same as Dirichlet boundary conditions on the physical model (which we do not require here).
- The goal then, is to incorporate these known boundaries, situated where we essentially know the function output, into the Bayesian emulation process which should lead to significantly improved emulators.
- We do this by formally updating the emulators by the information contained on the known boundaries, obtaining analytic results, then updating by runs D as usual.

Known Boundary Emulation: Motivation

- This is all fine, however, for some simulators, there exist input parameter settings, lying possibly on boundaries or hyperplanes in the input parameter space, where the simulator can be solved analytically (or just significantly faster).
- This may be due to the system in question, or at least a subset of the system outputs, behaving in a much simpler way for particular input settings.
- This is possibly due for example to various modules decoupling from more complex parts of the model (possibly when certain inputs are set to zero, switching some processes off).
- Note that this leads to Dirichlet boundary conditions, i.e. known simulator behaviour on various hyperplanes within $\mathcal{X}$, that impose constraints on the emulator itself, not the same as Dirichlet boundary conditions on the physical model (which we do not require here).
- The goal then, is to incorporate these known boundaries, situated where we essentially know the function output, into the Bayesian emulation process which should lead to significantly improved emulators.
- We do this by formally updating the emulators by the information contained on the known boundaries, obtaining analytic results, then updating by runs D as usual.

Single Known Boundary



The single known boundary case. Left panel: the points required for the $\mathrm{E}_K[f(x)]$ and $\mathrm{Var}_K[f(x)]$ calculation. x is the point we wish to emulate at, x^K its orthogonal projection onto the known boundary $\mathcal{K}$ at distance a. Right panel: the points required for the $\mathrm{Cov}_K[f(x), f(x')]$ calculation. x and x' are points we wish to update the covariance at, while x^K and x'^K are their orthogonal projection onto the known boundary $\mathcal{K}$, at distances a and a' respectively. In both panels, the $y^{(i)}$ represent a large number of points for which we can evaluate $f(y^{(i)})$ analytically (or at least very quickly).

How to update by knowledge of a single boundary 1

- We assume $\mathcal{K}$ is a $d-1$ dimensional hyperplane perpendicular to the x_1 direction.

- We wish to update our beliefs about $f(x)$, at the input point $x \in \mathcal{X}$.

- We can evaluate $f(x)$ at a large number, m, of points on $\mathcal{K}$ which we denote $y^{(1)}, \ldots, y^{(m)}$, and at the projection of x onto the boundary $\mathcal{K}$ denoted x^K, giving the $m+1$ column vector:

$$K = (f(x^K), f(y^{(1)}), \ldots, f(y^{(m)}))^T$$

- We wish to evaluate the expression for $\mathrm{E}_K[f(x)]$,

$$\mathrm{E}_K[f(x)] \;=\; \mathrm{E}[f(x)] + \mathrm{Cov}[f(x), K]\mathrm{Var}[K]^{-1}(K - \mathrm{E}[K])$$

  however due to m being very large (billions or trillions say depending on d) we cannot evaluate this directly as we cannot evaluate the $\mathrm{Var}[K]^{-1}$ term.

- However, if $x = x^K$ we know that trivially $\mathrm{E}_K[f(x^K)] = f(x^K)$ and $\mathrm{Var}_K[f(x^K)] = 0$, as $f(x)$ is deterministic.

How to update by knowledge of a single boundary 1

- We assume $\mathcal{K}$ is a $d-1$ dimensional hyperplane perpendicular to the x_1 direction.

- We wish to update our beliefs about $f(x)$, at the input point $x \in \mathcal{X}$.

- We can evaluate $f(x)$ at a large number, m, of points on $\mathcal{K}$ which we denote $y^{(1)}, \dots, y^{(m)}$, and at the projection of x onto the boundary $\mathcal{K}$ denoted x^K, giving the $m+1$ column vector:

$$K = (f(x^K), f(y^{(1)}), \dots, f(y^{(m)}))^T$$

- We wish to evaluate the expression for $\mathrm{E}_K[f(x)]$,

$$\mathrm{E}_K[f(x)] = \mathrm{E}[f(x)] + \mathrm{Cov}[f(x), K]\mathrm{Var}[K]^{-1}(K - \mathrm{E}[K])$$

  however due to m being very large (billions or trillions say depending on d) we cannot evaluate this directly as we cannot evaluate the $\mathrm{Var}[K]^{-1}$ term.

- However, if $x = x^K$ we know that trivially $\mathrm{E}_K[f(x^K)] = f(x^K)$ and $\mathrm{Var}_K[f(x^K)] = 0$, as $f(x)$ is deterministic.

How to update by knowledge of a single boundary 1

- We assume $\mathcal{K}$ is a $d-1$ dimensional hyperplane perpendicular to the x_1 direction.

- We wish to update our beliefs about $f(x)$, at the input point $x \in \mathcal{X}$.

- We can evaluate $f(x)$ at a large number, m, of points on $\mathcal{K}$ which we denote $y^{(1)}, \dots, y^{(m)}$, and at the projection of x onto the boundary $\mathcal{K}$ denoted x^K, giving the $m+1$ column vector:

$$K = (f(x^K), f(y^{(1)}), \dots, f(y^{(m)}))^T$$

- We wish to evaluate the expression for $\mathrm{E}_K[f(x)]$,

$$\mathrm{E}_K[f(x)] \;=\; \mathrm{E}[f(x)] + \mathrm{Cov}[f(x), K]\mathrm{Var}[K]^{-1}(K - \mathrm{E}[K])$$

  however due to m being very large (billions or trillions say depending on d) we cannot evaluate this directly as we cannot evaluate the $\mathrm{Var}[K]^{-1}$ term.

- However, if $x = x^K$ we know that trivially $\mathrm{E}_K[f(x^K)] = f(x^K)$ and $\mathrm{Var}_K[f(x^K)] = 0$, as $f(x)$ is deterministic.

How to update by knowledge of a single boundary 1

- We assume $\mathcal{K}$ is a $d-1$ dimensional hyperplane perpendicular to the x_1 direction.
- We wish to update our beliefs about $f(x)$, at the input point $x \in \mathcal{X}$.
- We can evaluate $f(x)$ at a large number, m, of points on $\mathcal{K}$ which we denote $y^{(1)}, \ldots, y^{(m)}$, and at the projection of x onto the boundary $\mathcal{K}$ denoted x^K, giving the $m+1$ column vector:

$$K = (f(x^K), f(y^{(1)}), \ldots, f(y^{(m)}))^T$$

- We wish to evaluate the expression for $\mathrm{E}_K[f(x)]$,

$$\mathrm{E}_K[f(x)] \;=\; \mathrm{E}[f(x)] + \mathrm{Cov}[f(x), K]\mathrm{Var}[K]^{-1}(K - \mathrm{E}[K])$$

  however due to m being very large (billions or trillions say depending on d) we cannot evaluate this directly as we cannot evaluate the $\mathrm{Var}[K]^{-1}$ term.
- However, if $x = x^K$ we know that trivially $\mathrm{E}_K[f(x^K)] = f(x^K)$ and $\mathrm{Var}_K[f(x^K)] = 0$, as $f(x)$ is deterministic.

How to update by knowledge of a single boundary 1

- We assume $\mathcal{K}$ is a $d-1$ dimensional hyperplane perpendicular to the x_1 direction.
- We wish to update our beliefs about $f(x)$, at the input point $x \in \mathcal{X}$.
- We can evaluate $f(x)$ at a large number, m, of points on $\mathcal{K}$ which we denote $y^{(1)}, \ldots, y^{(m)}$, and at the projection of x onto the boundary $\mathcal{K}$ denoted x^K, giving the $m+1$ column vector:

$$K = (f(x^K), f(y^{(1)}), \ldots, f(y^{(m)}))^T$$

- We wish to evaluate the expression for $\mathrm{E}_K[f(x)]$,

$$\mathrm{E}_K[f(x)] \;=\; \mathrm{E}[f(x)] + \mathrm{Cov}[f(x), K]\mathrm{Var}[K]^{-1}(K - \mathrm{E}[K])$$

  however due to m being very large (billions or trillions say depending on d) we cannot evaluate this directly as we cannot evaluate the $\mathrm{Var}[K]^{-1}$ term.
- However, if $x = x^K$ we know that trivially $\mathrm{E}_K[f(x^K)] = f(x^K)$ and $\mathrm{Var}_K[f(x^K)] = 0$, as $f(x)$ is deterministic.

How to update by knowledge of a single boundary 2

- As $f(x^K)$ is included as the first element of K, we note that

$$\begin{aligned} I_{(m+1)} &= \text{Var}[K]\text{Var}[K]^{-1} \\ &= \begin{pmatrix} \text{Cov}[f(x^K), K] \\ \text{Cov}[f(y^{(1)}), K] \\ \vdots \\ \text{Cov}[f(y^{(m)}), K] \end{pmatrix} \text{Var}[K]^{-1} \end{aligned}$$

- Taking the first row gives

$$\text{Cov}[f(x^K), K]\text{Var}[K]^{-1} = (1, 0, \cdots, 0)$$

- As we have defined x^K as the perpendicular projection of x onto $\mathcal{K}$, we can write

$$x = x^K + (a, 0, \ldots, 0)$$

How to update by knowledge of a single boundary 2

- As $f(x^K)$ is included as the first element of K, we note that

$$\begin{aligned} I_{(m+1)} &= \mathrm{Var}[K]\mathrm{Var}[K]^{-1} \\ &= \begin{pmatrix} \mathrm{Cov}[f(x^K), K] \\ \mathrm{Cov}[f(y^{(1)}), K] \\ \vdots \\ \mathrm{Cov}[f(y^{(m)}), K] \end{pmatrix} \mathrm{Var}[K]^{-1} \end{aligned}$$

- Taking the first row gives

$$\mathrm{Cov}[f(x^K), K]\mathrm{Var}[K]^{-1} = (1, 0, \cdots, 0)$$

- As we have defined x^K as the perpendicular projection of x onto $\mathcal{K}$, we can write

$$x = x^K + (a, 0, \ldots, 0)$$

How to update by knowledge of a single boundary 2

- As $f(x^K)$ is included as the first element of K, we note that

$$
\begin{aligned}
I_{(m+1)} &= \mathrm{Var}[K]\mathrm{Var}[K]^{-1} \\
&= \begin{pmatrix} \mathrm{Cov}[f(x^K), K] \\ \mathrm{Cov}[f(y^{(1)}), K] \\ \vdots \\ \mathrm{Cov}[f(y^{(m)}), K] \end{pmatrix} \mathrm{Var}[K]^{-1}
\end{aligned}
$$

- Taking the first row gives

$$\mathrm{Cov}[f(x^K), K]\mathrm{Var}[K]^{-1} = (1, 0, \cdots, 0)$$

- As we have defined x^K as the perpendicular projection of x onto $\mathcal{K}$, we can write

$$x = x^K + (a, 0, \ldots, 0)$$

How to update by knowledge of a single boundary 2

- Now we can exploit the symmetry of the product correlation structure:

$$\begin{aligned}\mathrm{Cov}[f(x), f(x^K)] &= \sigma^2 \prod_{i=1}^{d} r_i(x_i - x_i^K) = \sigma^2 r_1(x_1 - x_1^K) = \sigma^2 r_1(a) \\ &= r_1(a)\,\mathrm{Cov}[f(x^K), f(x^K)]\end{aligned}$$

  since $x_i = x_i^K$ for $i = 2, \ldots, d$ and $r_i(0) = 1$. Furthermore,

$$\begin{aligned}\mathrm{Cov}[f(x), f(y^{(j)})] &= \sigma^2 \prod_{i=1}^{d} r_i(x_i - y_i^{(j)}) = \sigma^2 r_1(x_1 - x_1^K) \prod_{i=2}^{d} r_i(x_i - y_i^{(j)}) \\ &= \sigma^2 r_1(a) \prod_{i=2}^{d} r_i(x_i^K - y_i^{(j)}) = r_1(a)\,\mathrm{Cov}[f(x^K), f(y^{(j)})]\end{aligned}$$

  since the first component of x^K and $y^{(j)}$ must be equal as they all lie on $\mathcal{K}$.

- Combining we obtain the covariance between $f(x)$ and the set K

$$\begin{aligned}\mathrm{Cov}[f(x), K] &= \left(\mathrm{Cov}[f(x), f(x^K)], \mathrm{Cov}[f(x), f(y^{(1)})], \cdots, \mathrm{Cov}[f(x), f(y^{(m)})]\right) \\ &= r_1(a)\,\mathrm{Cov}[f(x^K), K]\end{aligned}$$

How to update by knowledge of a single boundary 2

- Now we can exploit the symmetry of the product correlation structure:

$$\begin{aligned}\mathrm{Cov}[f(x), f(x^K)] &= \sigma^2 \prod_{i=1}^{d} r_i(x_i - x_i^K) = \sigma^2 r_1(x_1 - x_1^K) = \sigma^2 r_1(a) \\ &= r_1(a)\,\mathrm{Cov}[f(x^K), f(x^K)]\end{aligned}$$

since $x_i = x_i^K$ for $i = 2, \ldots, d$ and $r_i(0) = 1$. Furthermore,

$$\begin{aligned}\mathrm{Cov}[f(x), f(y^{(j)})] &= \sigma^2 \prod_{i=1}^{d} r_i(x_i - y_i^{(j)}) = \sigma^2 r_1(x_1 - x_1^K) \prod_{i=2}^{d} r_i(x_i - y_i^{(j)}) \\ &= \sigma^2 r_1(a) \prod_{i=2}^{d} r_i(x_i^K - y_i^{(j)}) = r_1(a)\,\mathrm{Cov}[f(x^K), f(y^{(j)})]\end{aligned}$$

since the first component of x^K and $y^{(j)}$ must be equal as they all lie on $\mathcal{K}$.

- Combining we obtain the covariance between $f(x)$ and the set K

$$\begin{aligned}\mathrm{Cov}[f(x), K] &= \Big(\mathrm{Cov}[f(x), f(x^K)], \mathrm{Cov}[f(x), f(y^{(1)})], \cdots, \mathrm{Cov}[f(x), f(y^{(m)})]\Big) \\ &= r_1(a)\,\mathrm{Cov}[f(x^K), K]\end{aligned}$$

How to update by knowledge of a single boundary 3

- We can use these results to analytically solve the adjusted emulator expectation

$$\begin{aligned}
\mathrm{E}_K[f(x)] &= \mathrm{E}[f(x)] + \mathrm{Cov}[f(x), K]\mathrm{Var}[K]^{-1}(K - \mathrm{E}[K]) \\
&= \mathrm{E}[f(x)] + r_1(a)\,\mathrm{Cov}[f(x^K), K]\mathrm{Var}[K]^{-1}(K - \mathrm{E}[K]) \\
&= \mathrm{E}[f(x)] + r_1(a)(1, 0, \cdots, 0)(K - \mathrm{E}[K]) \\
&= \mathrm{E}[f(x)] + r_1(a)(f(x^K) - \mathrm{E}[f(x^K)])
\end{aligned}$$

  Thus we have eliminated the need to explicitly invert the large matrix $\mathrm{Var}[K]$.

- Similarly, we find the adjusted variance to be

$$\begin{aligned}
\mathrm{Var}_K[f(x)] &= \mathrm{Var}[f(x)] - \mathrm{Cov}[f(x), K]\mathrm{Var}[K]^{-1}\mathrm{Cov}[K, f(x)] \\
&= \mathrm{Var}[f(x)] - r_1(a)(1, 0, \cdots, 0)\mathrm{Cov}[K, f(x)] \\
&= \mathrm{Var}[f(x)] - r_1(a)\mathrm{Cov}[f(x^K), f(x)] \\
&= \sigma^2(1 - r_1(a)^2)
\end{aligned}$$

- As these results require only evaluations of the analytic boundary function and the correlation function they can be implemented with trivial computational cost in comparison to a direct update by K.

- Note that they critically rely on the projected point $f(x^K)$ being in K: we only require finite boundaries such that $P_r(\mathcal{X}) \subset \mathcal{K}$.

How to update by knowledge of a single boundary 3

- We can use these results to analytically solve the adjusted emulator expectation

$$\begin{aligned}
\mathrm{E}_K[f(x)] &= \mathrm{E}[f(x)] + \mathrm{Cov}[f(x), K]\mathrm{Var}[K]^{-1}(K - \mathrm{E}[K]) \\
&= \mathrm{E}[f(x)] + r_1(a)\,\mathrm{Cov}[f(x^K), K]\mathrm{Var}[K]^{-1}(K - \mathrm{E}[K]) \\
&= \mathrm{E}[f(x)] + r_1(a)(1, 0, \cdots, 0)(K - \mathrm{E}[K]) \\
&= \mathrm{E}[f(x)] + r_1(a)(f(x^K) - \mathrm{E}[f(x^K)])
\end{aligned}$$

  Thus we have eliminated the need to explicitly invert the large matrix $\mathrm{Var}[K]$.

- Similarly, we find the adjusted variance to be

$$\begin{aligned}
\mathrm{Var}_K[f(x)] &= \mathrm{Var}[f(x)] - \mathrm{Cov}[f(x), K]\mathrm{Var}[K]^{-1}\mathrm{Cov}[K, f(x)] \\
&= \mathrm{Var}[f(x)] - r_1(a)(1, 0, \cdots, 0)\mathrm{Cov}[K, f(x)] \\
&= \mathrm{Var}[f(x)] - r_1(a)\mathrm{Cov}[f(x^K), f(x)] \\
&= \sigma^2(1 - r_1(a)^2)
\end{aligned}$$

- As these results require only evaluations of the analytic boundary function and the correlation function they can be implemented with trivial computational cost in comparison to a direct update by K.

- Note that they critically rely on the projected point $f(x^K)$ being in K: we only require finite boundaries such that $P_r(\mathcal{X}) \subset \mathcal{K}$.

How to update by knowledge of a single boundary 3

- We can use these results to analytically solve the adjusted emulator expectation

$$\begin{aligned} \mathrm{E}_K[f(x)] &= \mathrm{E}[f(x)] + \mathrm{Cov}[f(x), K]\mathrm{Var}[K]^{-1}(K - \mathrm{E}[K]) \\ &= \mathrm{E}[f(x)] + r_1(a)\,\mathrm{Cov}[f(x^K), K]\mathrm{Var}[K]^{-1}(K - \mathrm{E}[K]) \\ &= \mathrm{E}[f(x)] + r_1(a)(1, 0, \cdots, 0)(K - \mathrm{E}[K]) \\ &= \mathrm{E}[f(x)] + r_1(a)(f(x^K) - \mathrm{E}[f(x^K)]) \end{aligned}$$

  Thus we have eliminated the need to explicitly invert the large matrix $\mathrm{Var}[K]$.

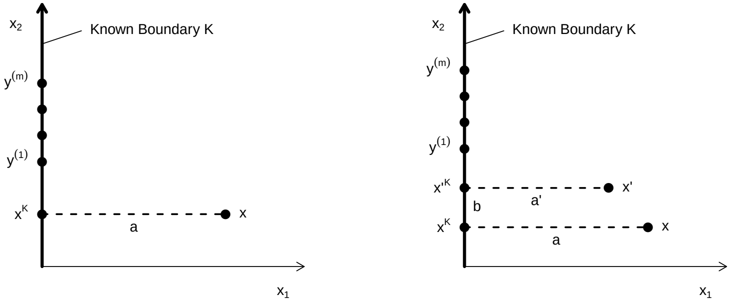
- Similarly, we find the adjusted variance to be

$$\begin{aligned} \mathrm{Var}_K[f(x)] &= \mathrm{Var}[f(x)] - \mathrm{Cov}[f(x), K]\mathrm{Var}[K]^{-1}\mathrm{Cov}[K, f(x)] \\ &= \mathrm{Var}[f(x)] - r_1(a)(1, 0, \cdots, 0)\mathrm{Cov}[K, f(x)] \\ &= \mathrm{Var}[f(x)] - r_1(a)\mathrm{Cov}[f(x^K), f(x)] \\ &= \sigma^2(1 - r_1(a)^2) \end{aligned}$$

- As these results require only evaluations of the analytic boundary function and the correlation function they can be implemented with trivial computational cost in comparison to a direct update by K.

- Note that they critically rely on the projected point $f(x^K)$ being in K: we only require finite boundaries such that $P_r(\mathcal{X}) \subset \mathcal{K}$.

Outline



The single known boundary case. Left panel: the points required for the $\mathrm{E}_K[f(x)]$ and $\mathrm{Var}_K[f(x)]$ calculation. x is the point we wish to emulate at, x^K its orthogonal projection onto the known boundary $\mathcal{K}$ at distance a. Right panel: the points required for the $\mathrm{Cov}_K[f(x), f(x')]$ calculation. x and x' are points we wish to update the covariance at, while x^K and x'^K are their orthogonal projection onto the known boundary $\mathcal{K}$, at distances a and a' respectively. In both panels, the $y^{(i)}$ represent a large number of points for which we can evaluate $f(y^{(i)})$ analytically (or at least very quickly).

How to update by knowledge of a single boundary 4

- For sequential emulation we also need the covariances:

$$\begin{aligned}
\mathrm{Cov}_K[f(x), f(x')] &= \mathrm{Cov}[f(x), f(x')] - \mathrm{Cov}[f(x), K]\mathrm{Var}[K]^{-1}\mathrm{Cov}[K, f(x')] \\
&= \mathrm{Cov}[f(x), f(x')] - r_1(a)(1, 0, \cdots, 0)\mathrm{Cov}[K, f(x')] \\
&= \mathrm{Cov}[f(x), f(x')] - r_1(a)\mathrm{Cov}[f(x^K), f(x')] \\
&= \mathrm{Cov}[f(x), f(x')] - r_1(a)\mathrm{Cov}[f(x^K), f(x'^K)]r_1(a') \\
&= \sigma^2 \prod_{i=1}^{d} r_i(x_i - x_i') - r_1(a)r_1(a')\sigma^2 \prod_{i=1}^{d} r_i(x_i^K - x_i'^K) \\
&= \sigma^2 \left(r_1(a - a') - r_1(a)r_1(a')\right) r_{-1}(x^K - x'^K) \\
&= \sigma^2 R_1(a, a')\, r_{-1}(x^K - x'^K)
\end{aligned}$$

where we have defined the correlation function of the projection of x and x' onto $\mathcal{K}$

$$r_{-1}(x^K - x'^K) = \prod_{i=2}^{d} r_i(x_i^K - x_i'^K) = \mathrm{Cov}[f(x^K), f(x'^K)]$$

and defined the ‘updated correlation component’ in the x_1 direction as

$$R_1(a, a') = r_1(a - a') - r_1(a)r_1(a')$$

Insights from the single boundary K update

$$\begin{aligned}
\mathrm{E}_K[f(x)] &= \mathrm{E}[f(x)] + r_1(a)(f(x^K) - \mathrm{E}[f(x^K)]) \\
\mathrm{Var}_K[f(x)] &= \sigma^2(1 - r_1(a)^2) \\
\mathrm{Cov}_K[f(x), f(x')] &= \sigma^2 R_1(a, a')\, r_{-1}(x^K - x'^K)
\end{aligned}$$

(a) *Sufficiency:* we see that for the emulator update $f(x^K)$ is sufficient for K. Can use this to include known boundaries directly in black box GP packages.

(b) *The correlation structure is still in product form:* therefore, we can update by further known boundaries either orthogonal to any of the remaining inputs x_i, with $i = 2, \ldots, d$, hence orthogonal to $\mathcal{K}$, or indeed by a second boundary parallel to $\mathcal{K}$.

(c) *Intuitive limiting behaviour:*

$$\begin{aligned}
\lim_{a\to 0} \mathrm{E}_K[f(x)] &= f(x^K), & \lim_{a\to 0} \mathrm{Var}_K[f(x)] &= 0, \\
\lim_{a\to \infty} \mathrm{E}_K[f(x)] &= \mathrm{E}[f(x)], & \lim_{a\to\infty} \mathrm{Var}_K[f(x)] &= \mathrm{Var}[f(x)],
\end{aligned}$$

$$\lim_{a\to 0} \mathrm{Cov}_K[f(x), f(x')] = \lim_{a'\to 0} \mathrm{Cov}_K[f(x), f(x')] = 0$$

$$\lim_{a,a'\to\infty} \mathrm{Cov}_K[f(x), f(x')] = \sigma^2 r(x - x') = \mathrm{Cov}[f(x), f(x')], \quad a - a' \text{ finite}$$

Insights from the single boundary K update

$$\begin{aligned}
\mathrm{E}_K[f(x)] &= \mathrm{E}[f(x)] + r_1(a)(f(x^K) - \mathrm{E}[f(x^K)]) \\
\mathrm{Var}_K[f(x)] &= \sigma^2(1 - r_1(a)^2) \\
\mathrm{Cov}_K[f(x), f(x')] &= \sigma^2 R_1(a, a')\, r_{-1}(x^K - x'^K)
\end{aligned}$$

(a) *Sufficiency:* we see that for the emulator update $f(x^K)$ is sufficient for K. Can use this to include known boundaries directly in black box GP packages.

(b) *The correlation structure is still in product form:* therefore, we can update by further known boundaries either orthogonal to any of the remaining inputs x_i, with $i = 2, \dots, d$, hence orthogonal to $\mathcal{K}$, or indeed by a second boundary parallel to $\mathcal{K}$.

(c) *Intuitive limiting behaviour:*

$$\begin{aligned}
\lim_{a\to 0} \mathrm{E}_K[f(x)] &= f(x^K), & \lim_{a\to 0} \mathrm{Var}_K[f(x)] &= 0, \\
\lim_{a\to\infty} \mathrm{E}_K[f(x)] &= \mathrm{E}[f(x)], & \lim_{a\to\infty} \mathrm{Var}_K[f(x)] &= \mathrm{Var}[f(x)],
\end{aligned}$$

$$\lim_{a\to 0} \mathrm{Cov}_K[f(x), f(x')] = \lim_{a'\to 0} \mathrm{Cov}_K[f(x), f(x')] = 0$$

$$\lim_{a,a'\to\infty} \mathrm{Cov}_K[f(x), f(x')] = \sigma^2 r(x - x') = \mathrm{Cov}[f(x), f(x')], \quad a - a' \text{ finite}$$

Insights from the single boundary K update

$$\begin{aligned} \mathrm{E}_K[f(x)] &= \mathrm{E}[f(x)] + r_1(a)(f(x^K) - \mathrm{E}[f(x^K)]) \\ \mathrm{Var}_K[f(x)] &= \sigma^2(1 - r_1(a)^2) \\ \mathrm{Cov}_K[f(x), f(x')] &= \sigma^2 R_1(a, a')\, r_{-1}(x^K - x'^K) \end{aligned}$$

(a) *Sufficiency:* we see that for the emulator update $f(x^K)$ is sufficient for K. Can use this to include known boundaries directly in black box GP packages.

(b) *The correlation structure is still in product form:* therefore, we can update by further known boundaries either orthogonal to any of the remaining inputs x_i, with $i = 2, \ldots, d$, hence orthogonal to $\mathcal{K}$, or indeed by a second boundary parallel to $\mathcal{K}$.

(c) *Intuitive limiting behaviour:*

$$\begin{aligned} \lim_{a \to 0} \mathrm{E}_K[f(x)] &= f(x^K), & \lim_{a \to 0} \mathrm{Var}_K[f(x)] &= 0, \\ \lim_{a \to \infty} \mathrm{E}_K[f(x)] &= \mathrm{E}[f(x)], & \lim_{a \to \infty} \mathrm{Var}_K[f(x)] &= \mathrm{Var}[f(x)], \end{aligned}$$

$$\lim_{a \to 0} \mathrm{Cov}_K[f(x), f(x')] = \lim_{a' \to 0} \mathrm{Cov}_K[f(x), f(x')] = 0$$

$$\lim_{a, a' \to \infty} \mathrm{Cov}_K[f(x), f(x')] = \sigma^2 r(x - x') = \mathrm{Cov}[f(x), f(x')], \quad a - a' \text{ finite}$$

Application to 2-dimensional Model

- Consider the problem of emulating the 2-dimensional function

$$f(x) \;=\; -\sin(2\pi x_2) + 0.9\sin(2\pi(1-x_1)(1-x_2))$$

defined over the region $\mathcal{X}$ given by $0 < x_1 < 1$, $0 < x_2 < 1$, where we assume a known boundary $\mathcal{K}$ at $x_1 = 0$, and hence have that

$$f(x^K) = f(0, x_2) = -1.9\sin(2\pi x_2)$$

- Using a prior expectation $\mathrm{E}[f(x)] = 0$, and a product Gaussian covariance structure with parameters $\theta = 0.4$ and $\sigma = 1$, we have

$$\begin{aligned} \mathrm{E}_K[f(x)] &= -1.9\exp\{-x_1^2/\theta^2\}\sin(2\pi x_2) \\ \mathrm{Var}_K[f(x)] &= 1 - \exp\{-2x_1^2/\theta^2\} \end{aligned}$$

- We can assess the emulator behaviour using simple emulator diagnostics over $\mathcal{X}$ of the form of the standardised values $S_K(x) = (\mathrm{E}_K[f(x)] - f(x))/\sqrt{\mathrm{Var}_K[f(x)]}$.

Application to 2-dimensional Model

- Consider the problem of emulating the 2-dimensional function

$$f(x) \;=\; -\sin(2\pi x_2) + 0.9\sin(2\pi(1-x_1)(1-x_2))$$

defined over the region $\mathcal{X}$ given by $0 < x_1 < 1$, $0 < x_2 < 1$, where we assume a known boundary $\mathcal{K}$ at $x_1 = 0$, and hence have that

$$f(x^K) = f(0, x_2) = -1.9\sin(2\pi x_2)$$

- Using a prior expectation $\mathrm{E}[f(x)] = 0$, and a product Gaussian covariance structure with parameters $\theta = 0.4$ and $\sigma = 1$, we have

$$\begin{aligned} \mathrm{E}_K[f(x)] &= -1.9\exp\{-x_1^2/\theta^2\}\sin(2\pi x_2) \\ \mathrm{Var}_K[f(x)] &= 1-\exp\{-2x_1^2/\theta^2\} \end{aligned}$$

- We can assess the emulator behaviour using simple emulator diagnostics over $\mathcal{X}$ of the form of the standardised values $S_K(x) = (\mathrm{E}_K[f(x)] - f(x))/\sqrt{\mathrm{Var}_K[f(x)]}$.

Application to 2-dimensional Model

- Consider the problem of emulating the 2-dimensional function

$$f(x) = -\sin(2\pi x_2) + 0.9\sin(2\pi(1-x_1)(1-x_2))$$

defined over the region $\mathcal{X}$ given by $0 < x_1 < 1$, $0 < x_2 < 1$, where we assume a known boundary $\mathcal{K}$ at $x_1 = 0$, and hence have that

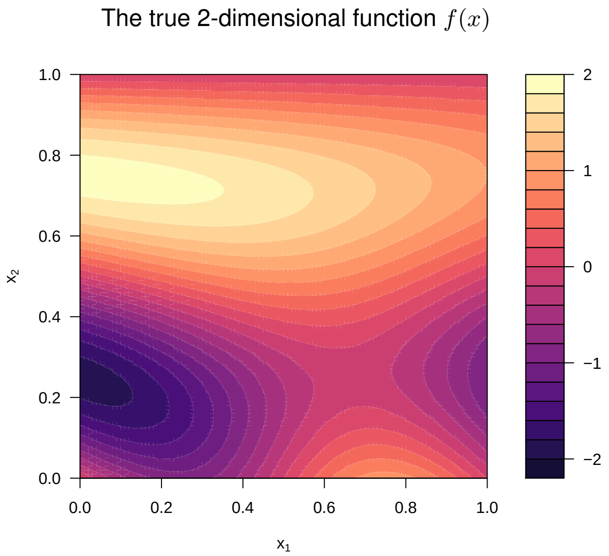
$$f(x^K) = f(0, x_2) = -1.9\sin(2\pi x_2)$$

- Using a prior expectation $\mathrm{E}[f(x)] = 0$, and a product Gaussian covariance structure with parameters $\theta = 0.4$ and $\sigma = 1$, we have

$$\begin{aligned} \mathrm{E}_K[f(x)] &= -1.9\exp\{-x_1^2/\theta^2\}\sin(2\pi x_2) \\ \mathrm{Var}_K[f(x)] &= 1 - \exp\{-2x_1^2/\theta^2\} \end{aligned}$$

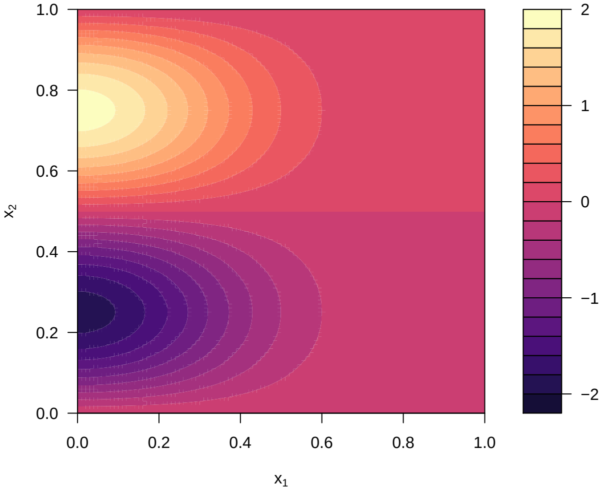
- We can assess the emulator behaviour using simple emulator diagnostics over $\mathcal{X}$ of the form of the standardised values $S_K(x) = (\mathrm{E}_K[f(x)] - f(x))/\sqrt{\mathrm{Var}_K[f(x)]}$.

Updating by a single known boundary $\mathcal{K}$ at $x_1 = 0$



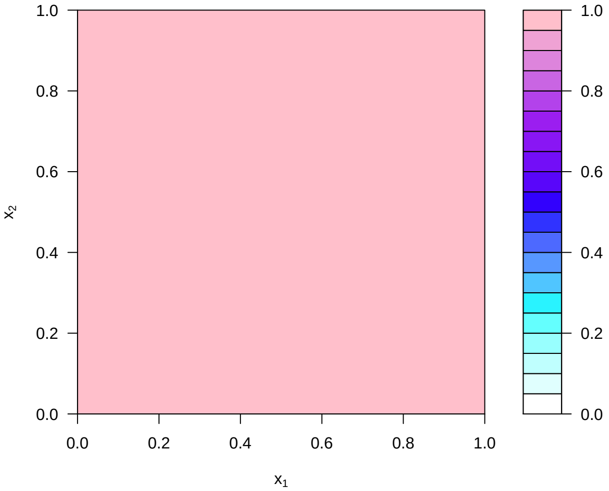
Updating by a single known boundary $\mathcal{K}$ at $x_1 = 0$

The emulator expectation $\mathrm{E}_K[f(x)]$



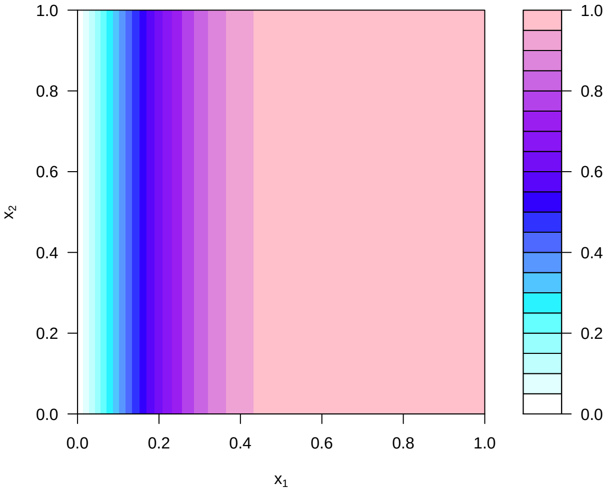
Updating by a single known boundary $\mathcal{K}$ at $x_1 = 0$

The emulator prior stan. dev. $\sqrt{\mathrm{Var}[f(x)]}$

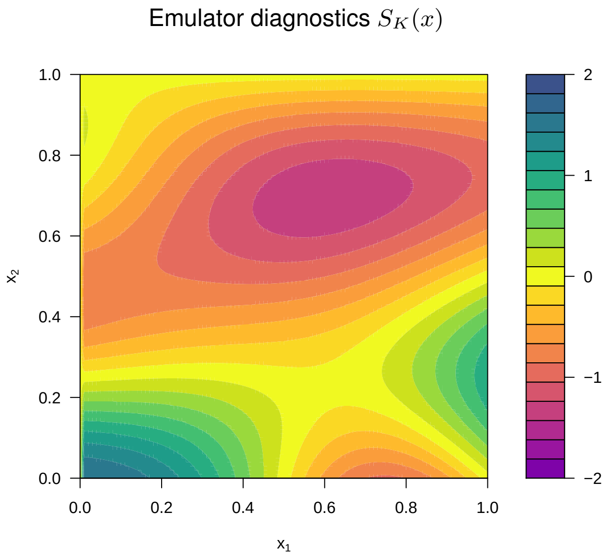


Updating by a single known boundary $\mathcal{K}$ at $x_1 = 0$

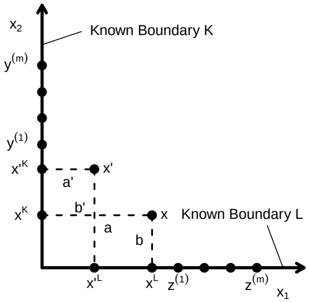
The emulator stan. dev. $\sqrt{\mathrm{Var}_K[f(x)]}$

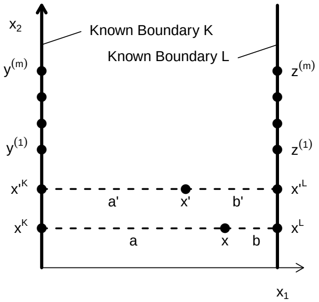


Updating by a single known boundary $\mathcal{K}$ at $x_1 = 0$



Updating by two known boundaries: perpendicular or parallel





Left panel: two perpendicular known boundaries. Right panel: two parallel known boundaries. In both cases x and x' are the points of interest for the emulation calculation, while x^K and x'^K are their orthogonal projection onto the known boundary $\mathcal{K}$, and x^L and x'^L their orthogonal projection onto the known boundary $\mathcal{L}$. The $y^{(i)}$ and $z^{(i)}$ represent a large number of points on the boundaries $\mathcal{K}$ and $\mathcal{L}$ respectively for which we can evaluate $f(y^{(i)})$ and $f(z^{(i)})$ analytically.

Updating by Two Perpendicular Known Boundaries

- We can update with respect to a second boundary $\mathcal{L}$, perpendicular to $\mathcal{K}$, containing points $x^L, z^{(1)}, \dots, z^{(m)}$ and vector of model evaluations L:

$$L \;=\; \left(f(x^L), f(z^{(1)}), \dots, f(z^{(m)})\right)^T ,$$

- An analogous proof to that seen before, now updated by K, gives

$$\mathrm{Cov}_K[f(x^L), L]\mathrm{Var}_K[L]^{-1} \;=\; (1, 0, \cdots, 0)$$

while as the product correlation structure is unperturbed by the K update, we have

$$\mathrm{Cov}_K[f(x), L] \;=\; r_2(b)\,\mathrm{Cov}_K[f(x^L), L]$$

- Hence, for example, the emulator expectation updated by $L \cup K$ is found to be

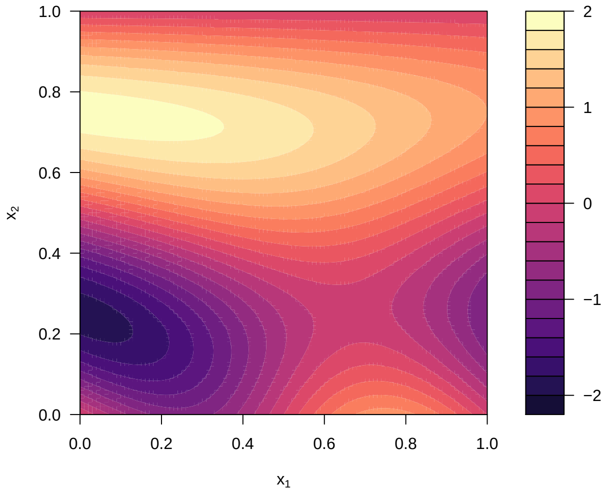
$$\begin{aligned}
\mathrm{E}_{L\cup K}[f(x)] \;&=\; \mathrm{E}_K[f(x)] + r_2(b)(1, 0, \cdots, 0)(L - \mathrm{E}_K[L]) \\
&=\; \mathrm{E}_K[f(x)] + r_2(b)(f(x^L) - \mathrm{E}_K[f(x^L)]) \\
&=\; \mathrm{E}[f(x)] + r_1(a)(f(x^K) - \mathrm{E}[f(x^K)]) \;+\; r_2(b)f(x^L) \\
&\qquad - r_2(b)(\mathrm{E}[f(x^L)] + r_1(a)(f(x^{LK}) - \mathrm{E}[f(x^{LK})])) \\
&=\; \mathrm{E}[f(x)] + r_1(a)\Delta f(x^K) + r_2(b)\Delta f(x^L) - r_1(a)r_2(b)\Delta f(x^{LK})
\end{aligned}$$

with $\Delta f(.) \equiv f(.) - \mathrm{E}[f(.)]$. Parallel boundary case a little trickier.

Emulators updated by two perpendicular boundaries $\mathcal{K}$ and $\mathcal{L}$.

Perpendicular boundaries, with $\mathcal{K} : x_1 = 0$ and $\mathcal{L} : x_2 = 0$

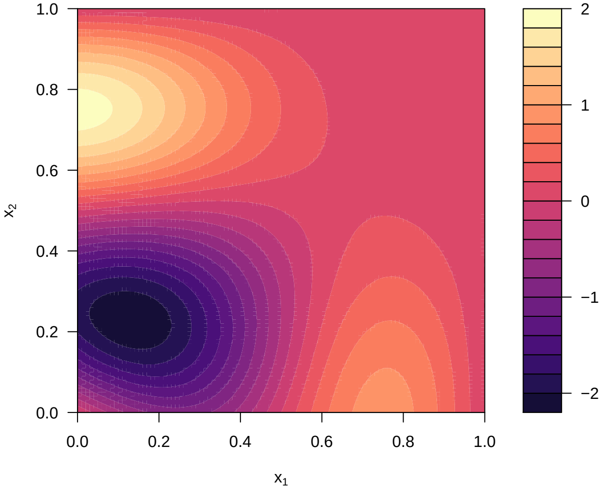
The true 2-dimensional function $f(x)$



Emulators updated by two perpendicular boundaries $\mathcal{K}$ and $\mathcal{L}$.

Perpendicular boundaries, with $\mathcal{K} : x_1 = 0$ and $\mathcal{L} : x_2 = 0$

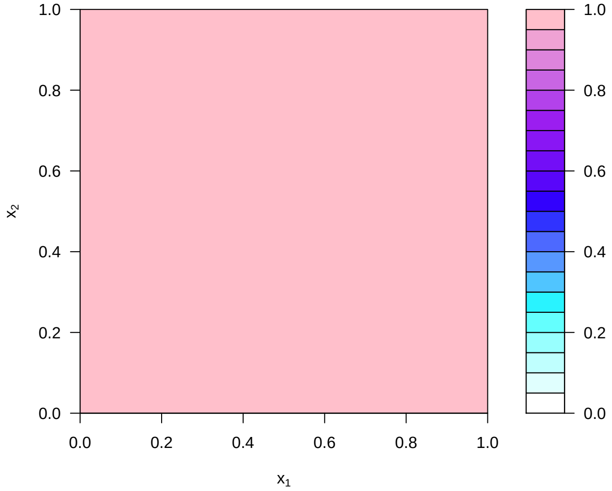
The emulator expectation $\mathrm{E}_{L \cup K}[f(x)]$



Emulators updated by two perpendicular boundaries $\mathcal{K}$ and $\mathcal{L}$.

Perpendicular boundaries, with $\mathcal{K} : x_1 = 0$ and $\mathcal{L} : x_2 = 0$

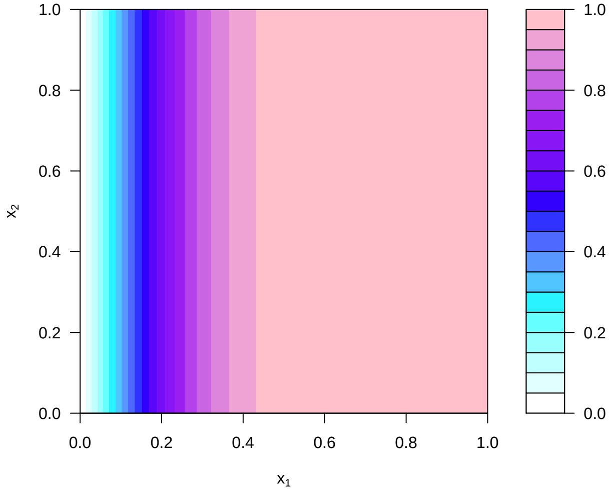
The emulator prior stan. dev. $\sqrt{\text{Var}[f(x)]}$



Emulators updated by two perpendicular boundaries $\mathcal{K}$ and $\mathcal{L}$.

Perpendicular boundaries, with $\mathcal{K} : x_1 = 0$ and $\mathcal{L} : x_2 = 0$

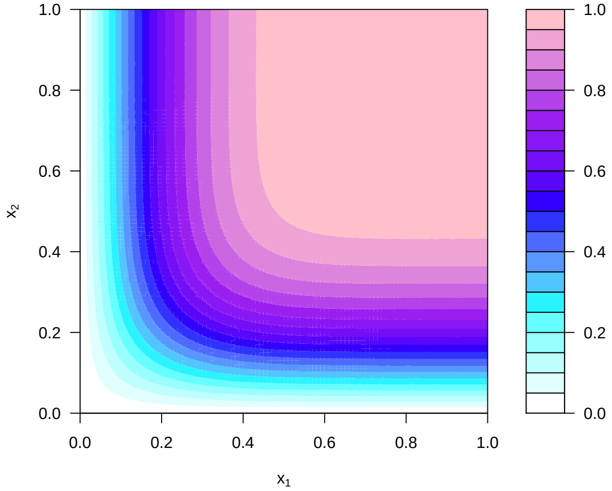
The emulator stan. dev. with a single boundary $\sqrt{\mathrm{Var}_K[f(x)]}$



Emulators updated by two perpendicular boundaries $\mathcal{K}$ and $\mathcal{L}$.

Perpendicular boundaries, with $\mathcal{K}: x_1 = 0$ and $\mathcal{L}: x_2 = 0$

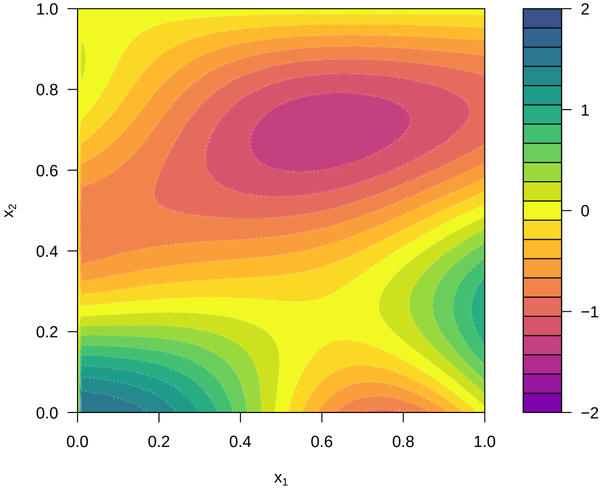
The emulator stan. dev. $\sqrt{\mathrm{Var}_{L \cup K}[f(x)]}$



Emulators updated by two perpendicular boundaries $\mathcal{K}$ and $\mathcal{L}$.

Perpendicular boundaries, with $\mathcal{K} : x_1 = 0$ and $\mathcal{L} : x_2 = 0$

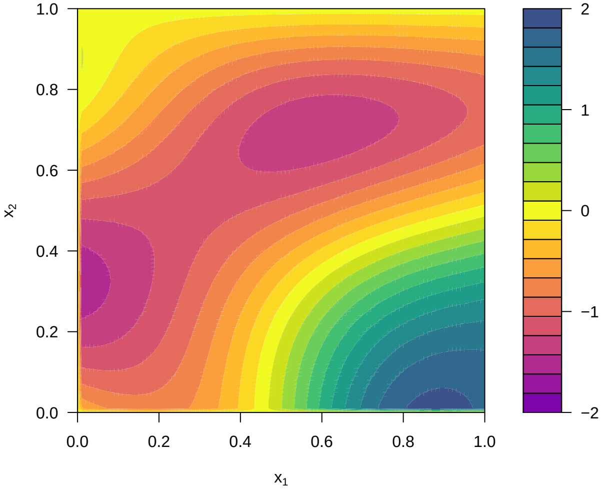
Emulator diagnostics with a single boundary $S_K(x)$



Emulators updated by two perpendicular boundaries $\mathcal{K}$ and $\mathcal{L}$.

Perpendicular boundaries, with $\mathcal{K} : x_1 = 0$ and $\mathcal{L} : x_2 = 0$

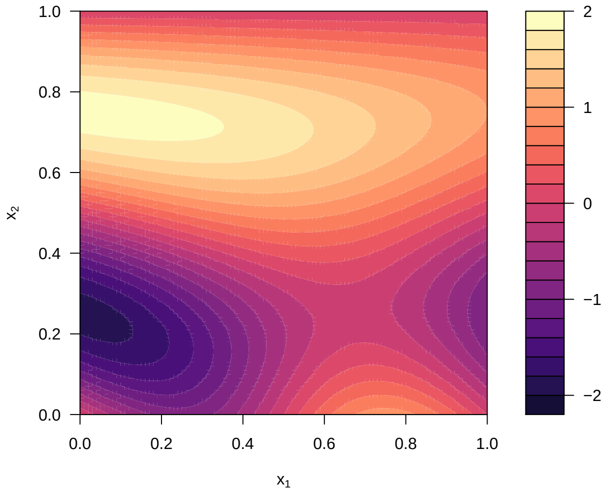
Emulator diagnostics $S_{L \cup K}(x)$



Emulators updated by two parallel boundaries $\mathcal{K}$ and $\mathcal{L}$.

Parallel boundaries, with $\mathcal{K} : x_1 = 0$ and $\mathcal{L} : x_1 = 1$

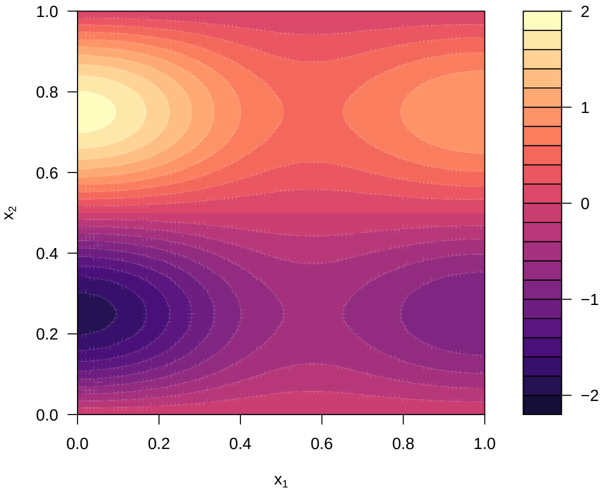
The true 2-dimensional function $f(x)$



Emulators updated by two parallel boundaries $\mathcal{K}$ and $\mathcal{L}$.

Parallel boundaries, with $\mathcal{K} : x_1 = 0$ and $\mathcal{L} : x_1 = 1$

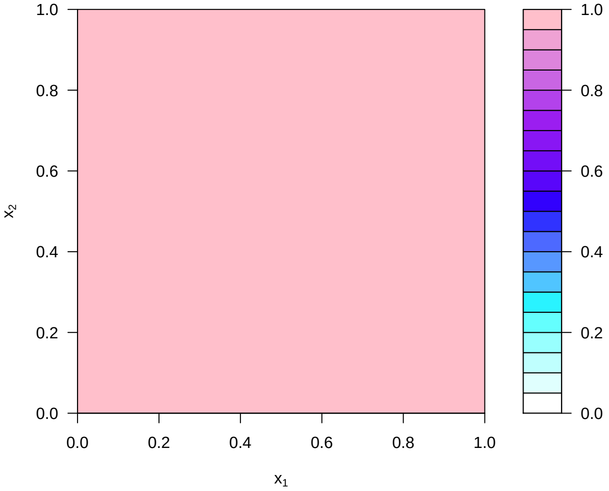
The emulator expectation $\mathrm{E}_{L \cup K}[f(x)]$



Emulators updated by two parallel boundaries $\mathcal{K}$ and $\mathcal{L}$.

Parallel boundaries, with $\mathcal{K}: x_1 = 0$ and $\mathcal{L}: x_1 = 1$

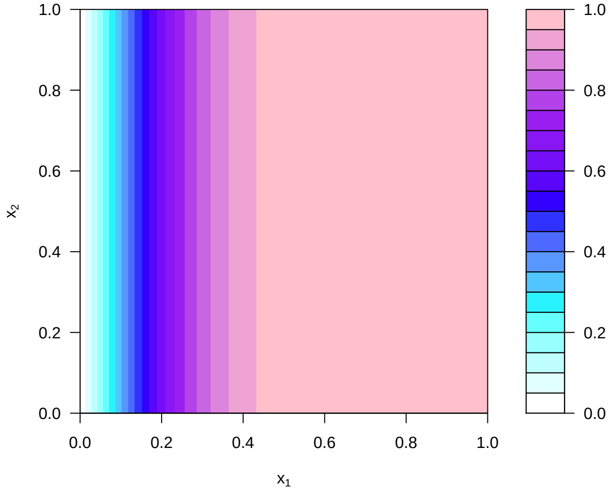
The emulator prior stan. dev. $\sqrt{\mathrm{Var}[f(x)]}$



Emulators updated by two parallel boundaries $\mathcal{K}$ and $\mathcal{L}$.

Parallel boundaries, with $\mathcal{K} : x_1 = 0$ and $\mathcal{L} : x_1 = 1$

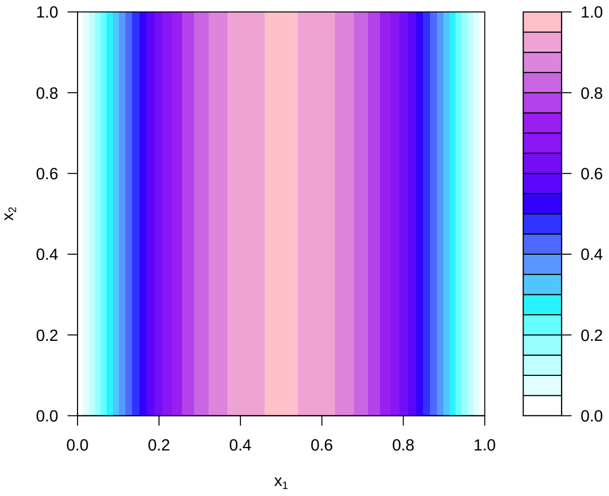
The emulator stan. dev. with a single boundary $\sqrt{\text{Var}_K[f(x)]}$



Emulators updated by two parallel boundaries $\mathcal{K}$ and $\mathcal{L}$.

Parallel boundaries, with $\mathcal{K} : x_1 = 0$ and $\mathcal{L} : x_1 = 1$

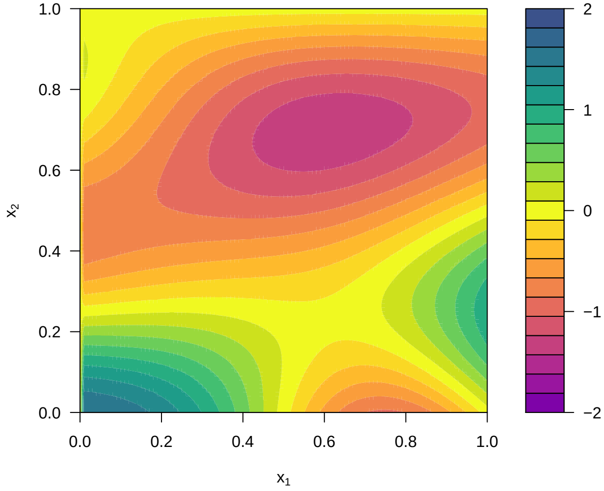
The emulator stan. dev. $\sqrt{\mathrm{Var}_{L \cup K}[f(x)]}$



Emulators updated by two parallel boundaries $\mathcal{K}$ and $\mathcal{L}$.

Parallel boundaries, with $\mathcal{K} : x_1 = 0$ and $\mathcal{L} : x_1 = 1$

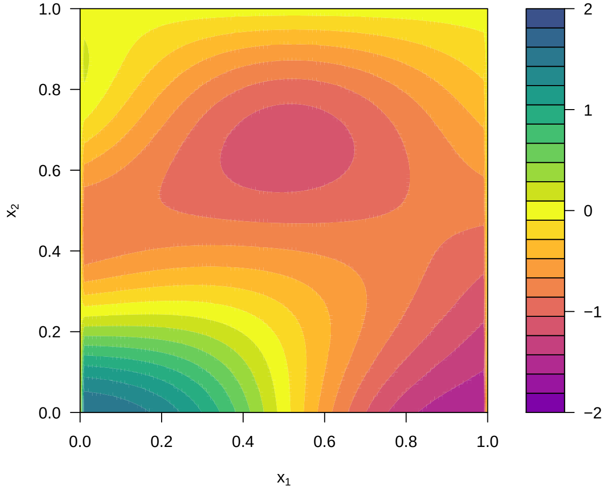
Emulator diagnostics with a single boundary $S_K(x)$



Emulators updated by two parallel boundaries $\mathcal{K}$ and $\mathcal{L}$.

Parallel boundaries, with $\mathcal{K} : x_1 = 0$ and $\mathcal{L} : x_1 = 1$

Emulator diagnostics $S_{L \cup K}(x)$



Summary of Analytic Results

Updating by one boundary $\mathcal{K}$, with $R_1(a,a') = r_1(a-a') - r_1(a)r_1(a')$ and $\Delta f(.) \equiv f(.) - \mathrm{E}[f(.)]$:

$$\begin{aligned}
\mathrm{E}_K[f(x)] &= \mathrm{E}[f(x)] + r_1(a)\Delta f(x^K) \\
\mathrm{Cov}_K[f(x), f(x')] &= \sigma^2 R_1(a,a')\, r_{-1}(x^K - x'^K) \\
\mathrm{Var}_K[f(x)] &= \sigma^2(1 - r_1(a)^2)
\end{aligned}$$

When updating by two perpendicular boundaries $\mathcal{K}$ and $\mathcal{L}$:

$$\begin{aligned}
\mathrm{E}_{L\cup K}[f(x)] &= \mathrm{E}[f(x)] + r_1(a)\Delta f(x^K) + r_2(b)\Delta f(x^L) - r_1(a)r_2(b)\Delta f(x^{LK}) \\
\mathrm{Cov}_{L\cup K}[f(x), f(x')] &= \sigma^2 R_1(a,a')\, R_2(b,b')\, r_{-1,-2}(x^{LK} - x'^{LK}) \\
\mathrm{Var}_{L\cup K}[f(x)] &= \sigma^2(1 - r_1^2(a))(1 - r_2^2(b))
\end{aligned}$$

When updating by two parallel boundaries $\mathcal{K}$ and $\mathcal{L}$, a distance $c = a + b$ apart:

$$\begin{aligned}
\mathrm{E}_{L\cup K}[f(x)] &= \mathrm{E}[f(x)] + \left[\frac{r_1(a) - r_1(b)r_1(c)}{1 - r_1^2(c)}\right]\Delta f(x^K) + \left[\frac{r_1(b) - r_1(a)r_1(c)}{1 - r_1^2(c)}\right]\Delta f(x^L) \\
\mathrm{Cov}_{L\cup K}[f(x), f(x')] &= \sigma^2\,\frac{r_{-1}(x^K - x'^K)}{1 - r_1^2(c)}\Big\{r_1(a-a')(1 - r_1^2(c)) - r_1(a)r_1(a') - r_1(b)r_1(b') \\
&\qquad + r_1(c)\left[r_1(a)r_1(b') + r_1(b)r_1(a')\right]\Big\} \\
\mathrm{Var}_{L\cup K}[f(x)] &= \sigma^2\,\frac{1}{1 - r_1^2(c)}\Big\{1 - r_1^2(c) - r_1^2(a) - r_1^2(b) + 2r_1(c)r_1(a)r_1(b)\Big\}
\end{aligned}$$

Aside: Continuous Known Boundaries

- Above we used a slightly artificial discrete and finite set of m known points on each boundary, which only requires a standard Bayes linear update.
- Here we generalise to a more natural continuum of known points $K = \{f(y) : y \in \mathcal{K}\}$ on a continuous boundary $\mathcal{K}$, which requires a generalised Bayes linear update.
- The adjusted expectation changes from the matrix equation,

$$\mathrm{E}_K[f(x)] \;=\; \mathrm{E}[f(x)] + \mathrm{Cov}[f(x), K]\mathrm{Var}[K]^{-1}(K - \mathrm{E}[K])$$

  to the integral equation

$$\mathrm{E}_K[f(x)] \;=\; \mathrm{E}[f(x)] + \int_{y\in\mathcal{K}}\int_{y'\in\mathcal{K}} \mathrm{Cov}[f(x), f(y)]\; s(y,y')\; (f(y') - \mathrm{E}[f(y')])dydy'$$

- Here $s(x,x')$ represents the infinite dimensional generalisation of $\mathrm{Var}[K]^{-1}$, and satisfies the equivalent inverse property:

$$\int_{y'\in\mathcal{K}} \mathrm{Cov}[f(y), f(y')]s(y',y'')\; dy' \;=\; \delta(y - y''), \qquad \text{for } y, y'' \in \mathcal{K}$$

  where $\delta(y - y'')$ is the Dirac delta function, the generalisation of the identity matrix.

Aside: Continuous Known Boundaries

- The derivations then take a similar structure to before. We have for $y \in \mathcal{K}$ that

$$\mathrm{Cov}[f(x), f(y)] = r_1(a)\,\mathrm{Cov}[f(x^K), f(y)],$$

- which on substitution into the generalised Bayes linear update gives

$$\begin{aligned}
\mathrm{E}_K[f(x)] &= \mathrm{E}[f(x)] + \int_{y\in\mathcal{K}}\int_{y'\in\mathcal{K}} r_1(a)\,\mathrm{Cov}[f(x^K), f(y)]\; s(y,y')\;(f(y') - \mathrm{E}[f(y')])dydy' \\
&= \mathrm{E}[f(x)] + r_1(a)\int_{y'\in\mathcal{K}} \delta(x^K - y')\;(f(y') - \mathrm{E}[f(y')])dy' \\
&= \mathrm{E}[f(x)] + r_1(a)\;(f(x^K) - \mathrm{E}[f(x^K)])
\end{aligned}$$

- Other results for $\mathrm{Var}_K[f(x)]$ and $\mathrm{Cov}_K[f(x), f(x')]$ are derived similarly, as are those for two boundaries.

- (End Aside!)

Updating by further model evaluations (the real point)

- Take the single boundary K case. As we have analytic expressions for $\mathrm{E}_K[f(x)]$, $\mathrm{Var}_K[f(x)]$ and $\mathrm{Cov}_K[f(x), f(x')]$ we are now easily able to include additional simulator evaluations into the emulation process.

- To do this, we perform n (expensive) evaluations, D, of the full simulator across $\mathcal{X}$, and use these to supplement the evaluations, K, available on the boundary.

- We want to update the emulator by the union of the evaluations D and K, that is to find $\mathrm{E}_{D\cup K}[f(x)]$, $\mathrm{Var}_{D\cup K}[f(x)]$ and $\mathrm{Cov}_{D\cup K}[f(x), f(x')]$. This can be achieved via a sequential Bayes Linear update:

$$\begin{aligned}
\mathrm{E}_{D\cup K}[f(x)] &= \mathrm{E}_K[f(x)] + \mathrm{Cov}_K[f(x), D]\mathrm{Var}_K[D]^{-1}(D - \mathrm{E}_K[D]) \\
\mathrm{Var}_{D\cup K}[f(x)] &= \mathrm{Var}_K[f(x)] - \mathrm{Cov}_K[f(x), D]\mathrm{Var}_K[D]^{-1}\mathrm{Cov}_K[D, f(x)] \\
\mathrm{Cov}_{D\cup K}[f(x), f(x')] &= \mathrm{Cov}_K[f(x), f(x')] - \mathrm{Cov}_K[f(x), D]\mathrm{Var}_K[D]^{-1}\mathrm{Cov}_K[D, f(x')]
\end{aligned}$$

- As typically n is small due to the relative expense of evaluating the full simulator, these calculations will remain tractable, as $\mathrm{Var}_K[D]^{-1}$ will be feasible for modest values of n.

Updating by further model evaluations (the real point)

- Take the single boundary K case. As we have analytic expressions for $\mathrm{E}_K[f(x)]$, $\mathrm{Var}_K[f(x)]$ and $\mathrm{Cov}_K[f(x), f(x')]$ we are now easily able to include additional simulator evaluations into the emulation process.

- To do this, we perform n (expensive) evaluations, D, of the full simulator across $\mathcal{X}$, and use these to supplement the evaluations, K, available on the boundary.

- We want to update the emulator by the union of the evaluations D and K, that is to find $\mathrm{E}_{D\cup K}[f(x)]$, $\mathrm{Var}_{D\cup K}[f(x)]$ and $\mathrm{Cov}_{D\cup K}[f(x), f(x')]$. This can be achieved via a sequential Bayes Linear update:

$$\begin{aligned}
\mathrm{E}_{D\cup K}[f(x)] &= \mathrm{E}_K[f(x)] + \mathrm{Cov}_K[f(x), D]\mathrm{Var}_K[D]^{-1}(D - \mathrm{E}_K[D]) \\
\mathrm{Var}_{D\cup K}[f(x)] &= \mathrm{Var}_K[f(x)] - \mathrm{Cov}_K[f(x), D]\mathrm{Var}_K[D]^{-1}\mathrm{Cov}_K[D, f(x)] \\
\mathrm{Cov}_{D\cup K}[f(x), f(x')] &= \mathrm{Cov}_K[f(x), f(x')] - \mathrm{Cov}_K[f(x), D]\mathrm{Var}_K[D]^{-1}\mathrm{Cov}_K[D, f(x')]
\end{aligned}$$

- As typically n is small due to the relative expense of evaluating the full simulator, these calculations will remain tractable, as $\mathrm{Var}_K[D]^{-1}$ will be feasible for modest values of n.

Updating by further model evaluations (the real point)

- Take the single boundary K case. As we have analytic expressions for $\mathrm{E}_K[f(x)]$, $\mathrm{Var}_K[f(x)]$ and $\mathrm{Cov}_K[f(x), f(x')]$ we are now easily able to include additional simulator evaluations into the emulation process.
- To do this, we perform n (expensive) evaluations, D, of the full simulator across $\mathcal{X}$, and use these to supplement the evaluations, K, available on the boundary.
- We want to update the emulator by the union of the evaluations D and K, that is to find $\mathrm{E}_{D\cup K}[f(x)]$, $\mathrm{Var}_{D\cup K}[f(x)]$ and $\mathrm{Cov}_{D\cup K}[f(x), f(x')]$. This can be achieved via a sequential Bayes Linear update:

$$\begin{aligned}
\mathrm{E}_{D\cup K}[f(x)] &= \mathrm{E}_K[f(x)] + \mathrm{Cov}_K[f(x), D]\mathrm{Var}_K[D]^{-1}(D - \mathrm{E}_K[D]) \\
\mathrm{Var}_{D\cup K}[f(x)] &= \mathrm{Var}_K[f(x)] - \mathrm{Cov}_K[f(x), D]\mathrm{Var}_K[D]^{-1}\mathrm{Cov}_K[D, f(x)] \\
\mathrm{Cov}_{D\cup K}[f(x), f(x')] &= \mathrm{Cov}_K[f(x), f(x')] - \mathrm{Cov}_K[f(x), D]\mathrm{Var}_K[D]^{-1}\mathrm{Cov}_K[D, f(x')]
\end{aligned}$$

- As typically n is small due to the relative expense of evaluating the full simulator, these calculations will remain tractable, as $\mathrm{Var}_K[D]^{-1}$ will be feasible for modest values of n.

Updating by further model evaluations (the real point)

- Take the single boundary K case. As we have analytic expressions for $\mathrm{E}_K[f(x)]$, $\mathrm{Var}_K[f(x)]$ and $\mathrm{Cov}_K[f(x), f(x')]$ we are now easily able to include additional simulator evaluations into the emulation process.

- To do this, we perform n (expensive) evaluations, D, of the full simulator across $\mathcal{X}$, and use these to supplement the evaluations, K, available on the boundary.

- We want to update the emulator by the union of the evaluations D and K, that is to find $\mathrm{E}_{D\cup K}[f(x)]$, $\mathrm{Var}_{D\cup K}[f(x)]$ and $\mathrm{Cov}_{D\cup K}[f(x), f(x')]$. This can be achieved via a sequential Bayes Linear update:

$$\begin{aligned}
\mathrm{E}_{D\cup K}[f(x)] &= \mathrm{E}_K[f(x)] + \mathrm{Cov}_K[f(x), D]\mathrm{Var}_K[D]^{-1}(D - \mathrm{E}_K[D]) \\
\mathrm{Var}_{D\cup K}[f(x)] &= \mathrm{Var}_K[f(x)] - \mathrm{Cov}_K[f(x), D]\mathrm{Var}_K[D]^{-1}\mathrm{Cov}_K[D, f(x)] \\
\mathrm{Cov}_{D\cup K}[f(x), f(x')] &= \mathrm{Cov}_K[f(x), f(x')] - \mathrm{Cov}_K[f(x), D]\mathrm{Var}_K[D]^{-1}\mathrm{Cov}_K[D, f(x')]
\end{aligned}$$

- As typically n is small due to the relative expense of evaluating the full simulator, these calculations will remain tractable, as $\mathrm{Var}_K[D]^{-1}$ will be feasible for modest values of n.

Design of Known Boundary Emulation Experiments

- The inclusion of known boundaries such as K will of course increase the accuracy of the emulator, for negligible computational cost.
- The extent of this increase depends upon the dimension of K and on the specifics of the prior correlation structure.
- In addition, if we are aware of the boundaries in advance, we can design a more informative set of runs D that exploit the known boundaries.
- We can examine the emulator variance $\text{Var}_{D\cup K}[f(x)]$ to choose D.
- *V-optimality:* An example of this is to choose a design x_D that minimises

$$c(x_D) = \text{trace}(\text{Var}_{D\cup K}[f(X_G)])$$

where X_G is a large set of points covering $\mathcal{X}$.

Design of Known Boundary Emulation Experiments

- The inclusion of known boundaries such as K will of course increase the accuracy of the emulator, for negligible computational cost.
- The extent of this increase depends upon the dimension of K and on the specifics of the prior correlation structure.
- In addition, if we are aware of the boundaries in advance, we can design a more informative set of runs D that exploit the known boundaries.
- We can examine the emulator variance $\mathrm{Var}_{D\cup K}[f(x)]$ to choose D.
- *V-optimality:* An example of this is to choose a design x_D that minimises

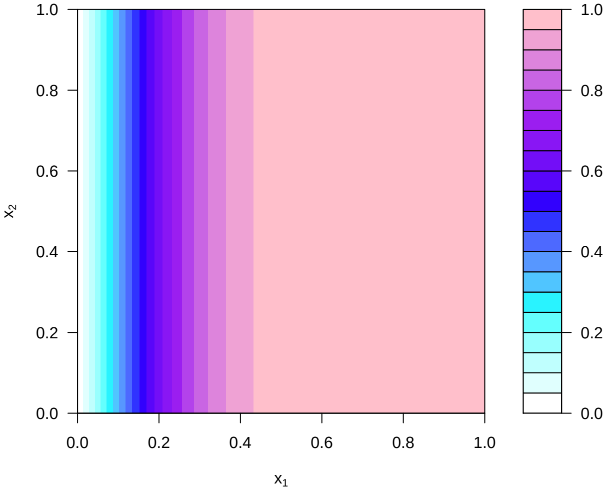
  $$c(x_D) = \mathrm{trace}(\mathrm{Var}_{D\cup K}[f(X_G)])$$

  where X_G is a large set of points covering $\mathcal{X}$.

Designing runs in the presence of a known boundary $\mathcal{K}$.

Single boundary, with $\mathcal{K} : x_1 = 0$

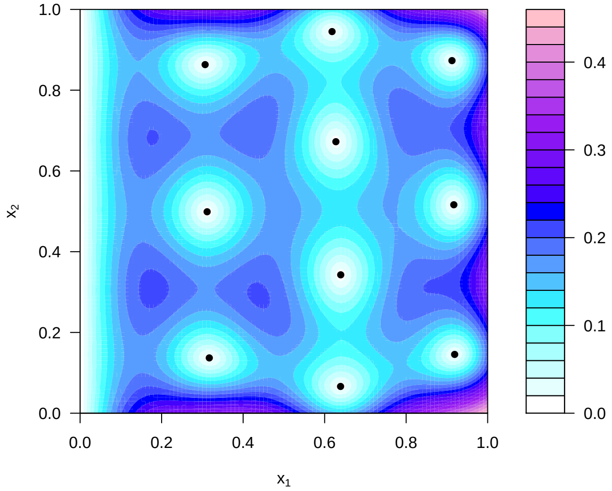
The emulator stan. dev. with a single boundary $\sqrt{\mathrm{Var}_K[f(x)]}$



Designing runs in the presence of a known boundary $\mathcal{K}$.

Single boundary, with $\mathcal{K} : x_1 = 0$

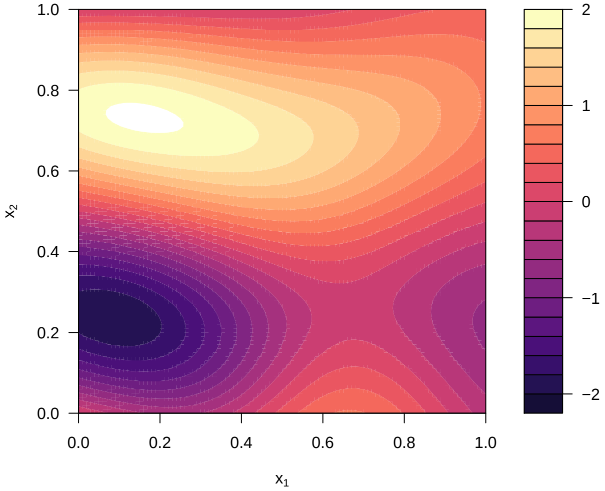
The emulator stan. dev. $\sqrt{\mathrm{Var}_{D\cup K}[f(x)]}$, with D a 10 point V-optimal design.



Designing runs in the presence of a known boundary $\mathcal{K}$.

Single boundary, with $\mathcal{K} : x_1 = 0$

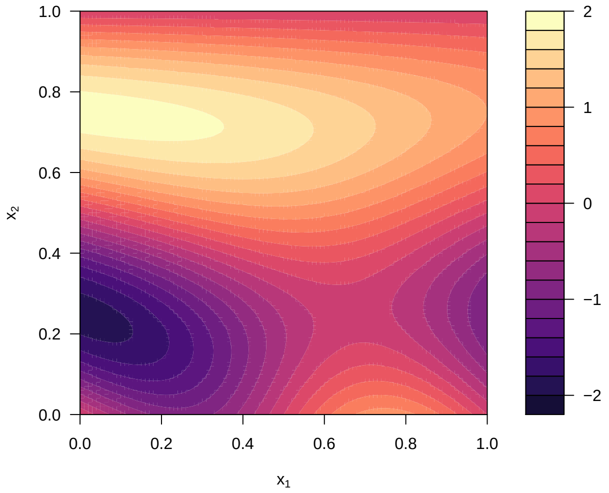
The emulator stan. dev. $\sqrt{\mathrm{Var}_{D \cup K}[f(x)]}$, with D a 10 point V-optimal design.



Designing runs in the presence of a known boundary $\mathcal{K}$.

Single boundary, with $\mathcal{K} : x_1 = 0$

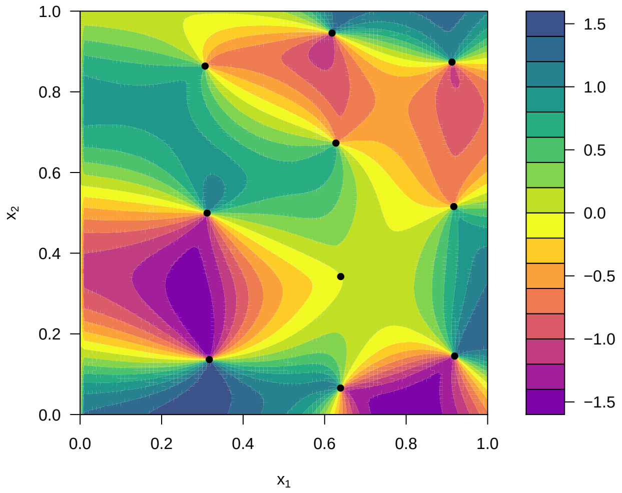
The true 2-dimensional function $f(x)$



Designing runs in the presence of a known boundary $\mathcal{K}$.

Single boundary, with $\mathcal{K} : x_1 = 0$

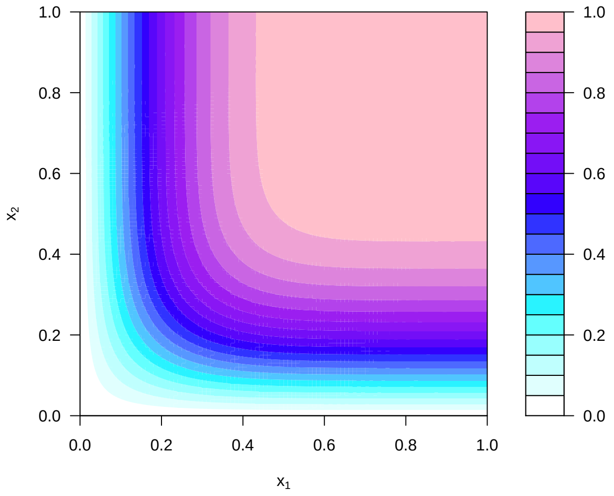
Emulator diagnostics with a single boundary $S_{D \cup K}(x)$



Designing runs in the presence of known boundaries $\mathcal{K}$ and $\mathcal{L}$.

Perpendicular boundaries, with $\mathcal{K} : x_1 = 0$ and $\mathcal{L} : x_2 = 0$

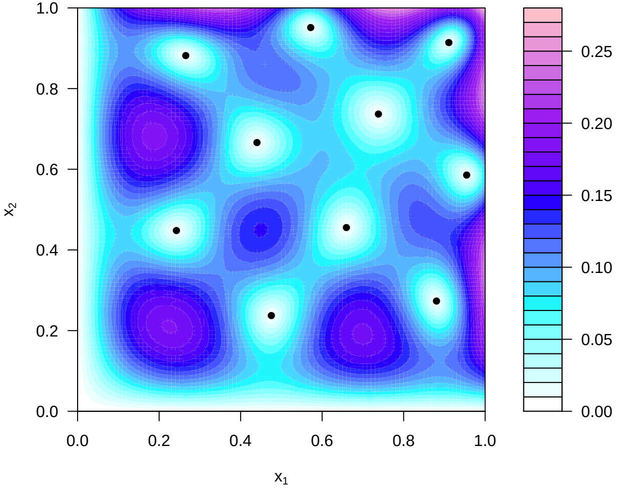
The emulator stan. dev. $\sqrt{\mathrm{Var}_{L \cup K}[f(x)]}$



Designing runs in the presence of known boundaries $\mathcal{K}$ and $\mathcal{L}$.

Perpendicular boundaries, with $\mathcal{K} : x_1 = 0$ and $\mathcal{L} : x_2 = 0$

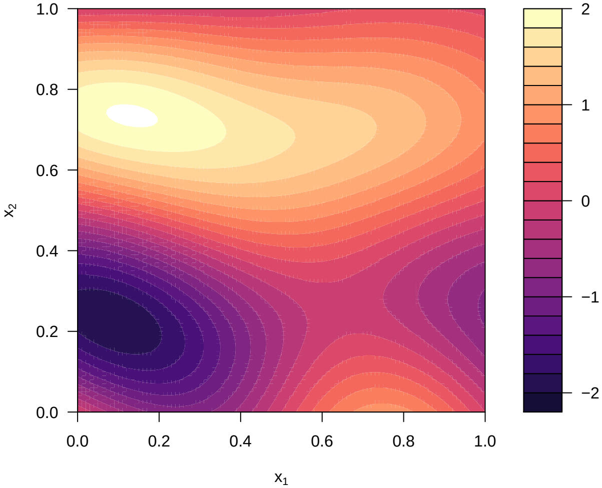
The emulator stan. dev. $\sqrt{\mathrm{Var}_{D \cup L \cup K}[f(x)]}$, with D a 10 pt V-optimal design



Designing runs in the presence of known boundaries $\mathcal{K}$ and $\mathcal{L}$.

Perpendicular boundaries, with $\mathcal{K} : x_1 = 0$ and $\mathcal{L} : x_2 = 0$

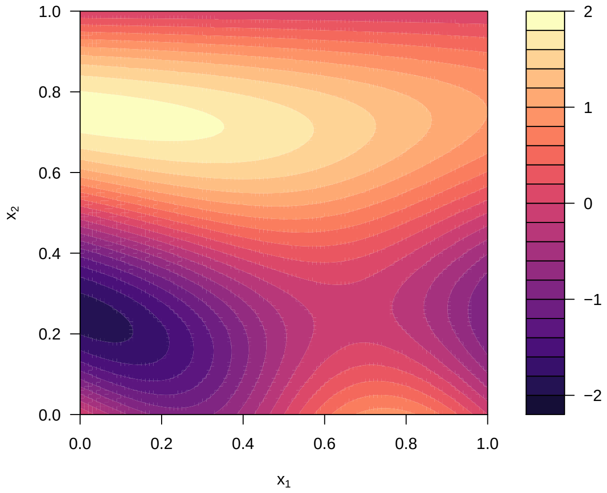
The emulator expectation $\mathrm{E}_{D \cup L \cup K}[f(x)]$, with D a 10 pt V-optimal design



Designing runs in the presence of known boundaries $\mathcal{K}$ and $\mathcal{L}$.

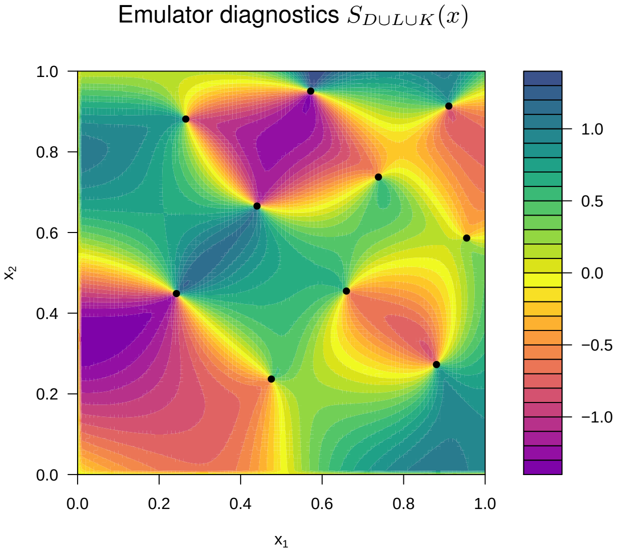
Perpendicular boundaries, with $\mathcal{K} : x_1 = 0$ and $\mathcal{L} : x_2 = 0$

The true 2-dimensional function $f(x)$



Designing runs in the presence of known boundaries $\mathcal{K}$ and $\mathcal{L}$.

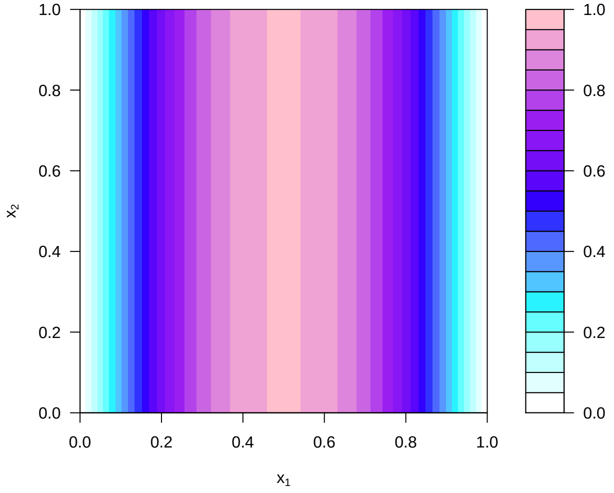
Perpendicular boundaries, with $\mathcal{K} : x_1 = 0$ and $\mathcal{L} : x_2 = 0$



Designing runs in the presence of known boundaries $\mathcal{K}$ and $\mathcal{L}$.

Parallel boundaries, with $\mathcal{K} : x_1 = 0$ and $\mathcal{L} : x_1 = 1$

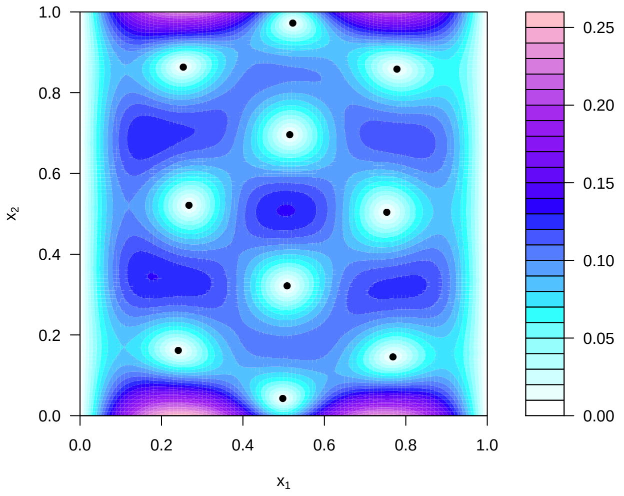
The emulator stan. dev. $\sqrt{\text{Var}_{L \cup K}[f(x)]}$



Designing runs in the presence of known boundaries $\mathcal{K}$ and $\mathcal{L}$.

Parallel boundaries, with $\mathcal{K} : x_1 = 0$ and $\mathcal{L} : x_1 = 1$

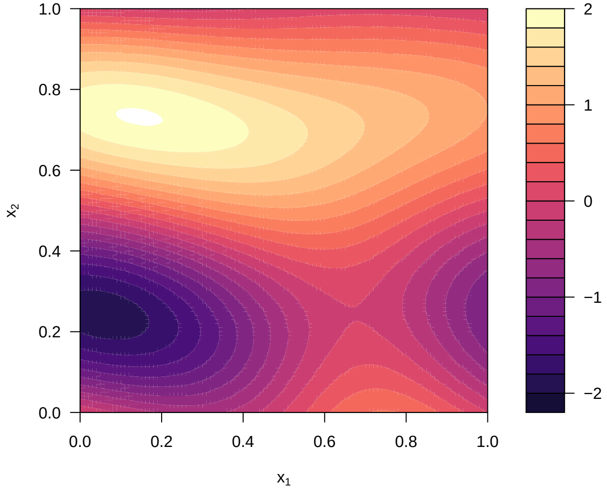
The emulator stan. dev. $\sqrt{\mathrm{Var}_{D \cup L \cup K}[f(x)]}$, with D a 10 pt V-optimal design



Designing runs in the presence of known boundaries $\mathcal{K}$ and $\mathcal{L}$.

Parallel boundaries, with $\mathcal{K} : x_1 = 0$ and $\mathcal{L} : x_1 = 1$

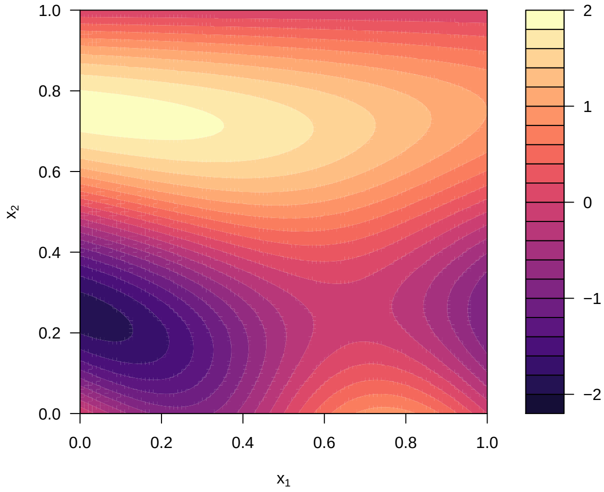
The emulator expectation $\mathrm{E}_{D \cup L \cup K}[f(x)]$, with D a 10 pt V-optimal design



Designing runs in the presence of known boundaries $\mathcal{K}$ and $\mathcal{L}$.

Parallel boundaries, with $\mathcal{K} : x_1 = 0$ and $\mathcal{L} : x_1 = 1$

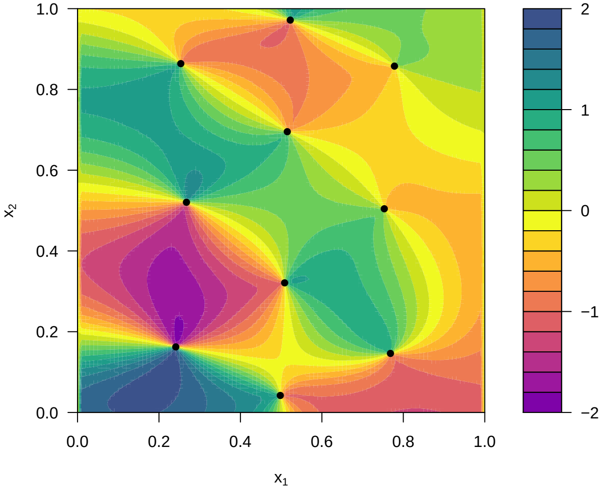
The true 2-dimensional function $f(x)$



Designing runs in the presence of known boundaries $\mathcal{K}$ and $\mathcal{L}$.

Parallel boundaries, with $\mathcal{K} : x_1 = 0$ and $\mathcal{L} : x_1 = 1$

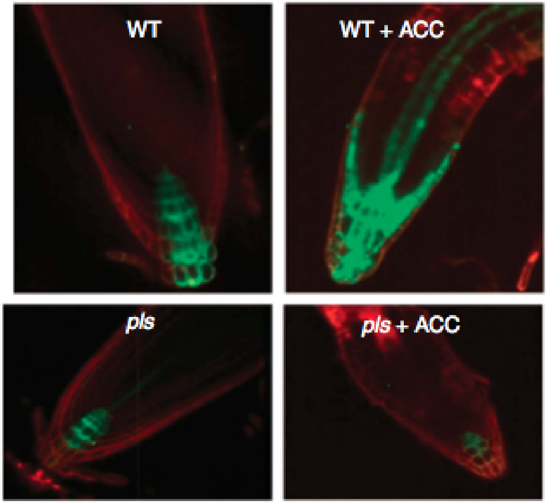
Emulator diagnostics $S_{D \cup L \cup K}(x)$



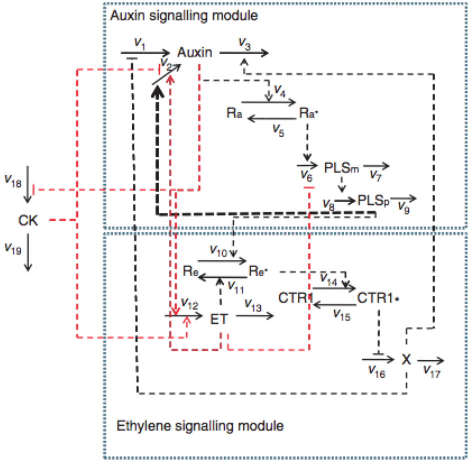
Application to a Systems Biology Model of Arabidopsis

- Small flowering plant related to cabbage and mustard.
- One of the model organisms used for studying plant biology and the first plant to have its entire genome sequenced.
- Changes in it are easily observed, making it very useful.

Measurements of root hormone level.



Root Hormonal Crosstalk Reaction Network Model of Arabidopsis Thaliana



- Liu et. al. developed a kinetic model of hormonal crosstalk in Arabidopsis,
- Model describes the function of POLARIS (PLS) peptide in auxin biosysthesis.

$$\frac{d[Auxin]}{dt} = \frac{k_{1a}}{1+\frac{[X]}{k_1}} + k_2 + k_{2a}\frac{[ET]}{1+\frac{[CK]}{k_{2b}}}\frac{[PLSp]}{k_{2c}+[PLSp]} + \frac{V_{IAA}[IAA]}{Km_{IAA}+[IAA]} - \left(k_3 + \frac{k_{3a}[PIN1pm]}{k3auxin+[Auxin]}\right)[Auxin]$$

$$\frac{d[X]}{dt} = k_{16} - k_{16a}[CTR1^*] - k_{17}[X]$$

$$\frac{d[PLSp]}{dt} = k_8[PLSm] - k_9[PLSp]$$

$$\frac{d[Ra]}{dt} = -k_4[Auxin][Ra] + k_5[Ra^*]$$

$$\frac{d[Ra^*]}{dt} = k_4[Auxin][Ra] - k_5[Ra^*]$$

$$\frac{d[CK]}{dt} = \frac{k_{18a}}{1+\frac{[Auxin]}{k_{18}}} - k_{19}[CK] + \frac{V_{CK}[cytokinin]}{Km_{CK}+[cytokinin]}$$

$$\frac{d[ET]}{dt} = k_{12} + k_{12a}[Auxin][CK] - k_{13}[ET] + \frac{V_{ACC}[ACC]}{Km_{ACC}+[ACC]}$$

$$\frac{d[PLSm]}{dt} = \frac{k_6[Ra^*]}{1+\frac{[ET]}{k_{6a}}} - k_7[PLSm]$$

$$\frac{d[Re]}{dt} = k_{11}[Re^*][ET] - (k_{10} + k_{10a}[PLSp])[R$$

$$\frac{d[Re^*]}{dt} = -k_{11}[Re^*][ET] + (k_{10} + k_{10a}[PLSp])$$

$$\frac{d[CTR1]}{dt} = -k_{14}[Re^*][CTR1] + k_{15}[CTR1^*]$$

$$\frac{d[CTR1^*]}{dt} = k_{14}[Re^*][CTR1] - k_{15}[CTR1^*]$$

$$\frac{d[PIN1m]}{dt} = \frac{k_{20a}}{k_{20b}+[CK]}[X]\frac{[Auxin]}{k_{20c}+[Auxin]} - k_{1_v21}[PIN1m]$$

$$\frac{d[PIN1pi]}{dt} = k_{22a}[PIN1m] - k_{1_v23}[PIN1pi] - k_{1_v24}[PIN1pi] + \frac{k_{25a}[PIN1pm]}{1+\frac{[Auxin]}{k_{25b}}}$$

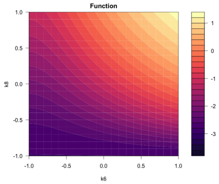
$$\frac{d[PIN1pm]}{dt} = k_{1_v24}[PIN1pi] - \frac{k_{25a}[PIN1pm]}{1+\frac{[Auxin]}{k_{25b}}}$$

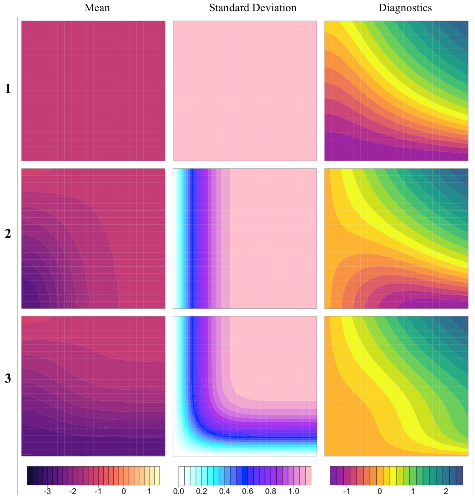
$$\frac{d[IAA]}{dt} = 0$$

$$\frac{d[cytokinin]}{dt} = 0$$

$$\frac{d[ACC]}{dt} = 0$$

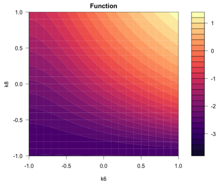
Arabidopsis.

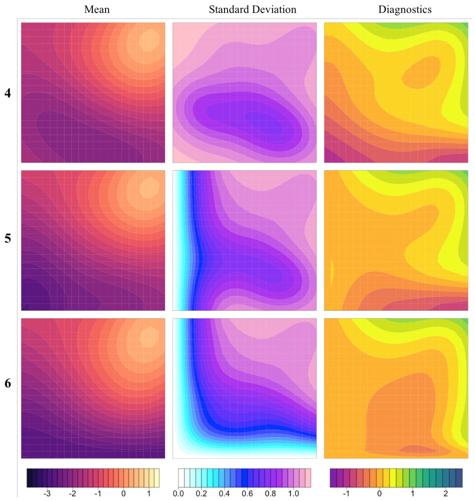




Results of emulating, without training points, a 2-dimensional k_6 (x-axes) by k_8 (y-axes) slice of the 6-dimensional input space, with each of the inputs $\{k_4, k_{6a}, k_7, k_9\}$ set to the mid-values of their square root ranges. The first row shows the results when using prior emulator beliefs only, the second row shows the results when updating by the boundary $\mathcal{K} : k_6 = 0$ only, and the third row shows the results when updating using both boundaries $\mathcal{K} : k_6 = 0$ and $\mathcal{L} : k_8 = 0$. Each column from left to right shows emulator mean, standard

Arabidopsis.





Results of emulating, with training points, a 2-dimensional k_6 (x-axes) by k_8 (y-axes) slice of the 6-dimensional input space, with each of the inputs $\{k_4, k_{6a}, k_7, k_9\}$ set to the mid-values of their square root ranges. The first row shows the results when updating by the training points only, the second row shows the results when updating by the training points and the known boundary $\mathcal{K} : k_6 = 0$, and the third row shows the results when updating by the training points and the two known boundaries $\mathcal{K} : k_6 = 0$ and $\mathcal{L} : k_8 = 0$. Each column from

Arabidopsis.

θ	Known Boundaries	Maximin LH	Warped Maximin LH
0.7	Without	**0.9247**	0.9489
	With	0.6763	**0.5886**
1.2	Without	**0.4427**	0.6601
	With	0.2986	**0.2530**

Table: A table of RMSEs of the 2000 diagnostic points using emulators constructed with and without both the known boundaries $\mathcal{K}$ and $\mathcal{L}$ for a maximin Latin hypercube design and the warped version of this design, for two choices of correlation length θ. The numbers in bold correspond to the preferred strategy, for the given knowledge of the boundaries.

Arabidopsis.

θ	Known Boundaries	Iterative V-Opt.	Warped Iter. V-Opt.	Iter. V-Opt. with KBs
0.7	Without	**0.8166**	0.9013	0.9700
	With	0.5815	**0.5091**	**0.5101**
1.2	Without	**0.4476**	0.6687	0.9028
	With	0.2830	**0.2340**	**0.2414**

Table: A table of RMSEs of the 2000 diagnostic points using emulators constructed with and without the known boundaries $\mathcal{K}$ and $\mathcal{L}$ for three designs, namely a standard iterative V-optimal design without the known boundaries, the warped version of this design, and an iterative V-optimal design which takes account of the known boundaries.

Extensions: Multiple Boundaries of Differing Dimension

- In the systems biology example there were actually multiple known boundaries of differing dimension.
- This approach can be used to incorporate multiple parallel and perpendicular boundaries of differing dimension as long as they satisfy certain rules.
- For example,

  Theorem: The expectation and covariance of $f(x)$, sequentially adjusted by multiple perpendicular boundaries $\mathcal{K}_1, ..., \mathcal{K}_h$, are given by:

$$\mathrm{E}_{K_1\cup\cdots\cup K_h}[f(x)] \;=\; \mathrm{E}[f(x)] + \sum_{i=1}^{h}(-1)^{i+1} \sum_{A\subset 1:h,|A|=i} \prod_{j\in A} \mathrm{r}_{k_{j-1}+1:k_j}(a)\Delta f(x^{K_A})$$

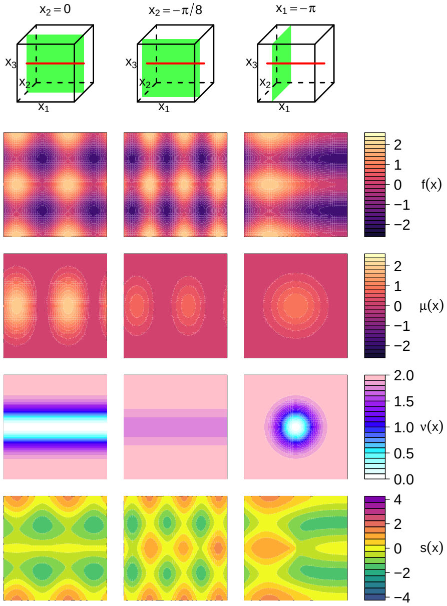
$$\mathrm{Cov}_{K_1\cup\cdots\cup K_h}[f(x), f(x')] \;=\; \prod_{j=1}^{h} R_{k_{j-1}+1:k_j}(a,a')\,\mathrm{r}_{k_h+1:p}(x-x')$$

Extensions: Multiple Boundaries of Differing Dimension

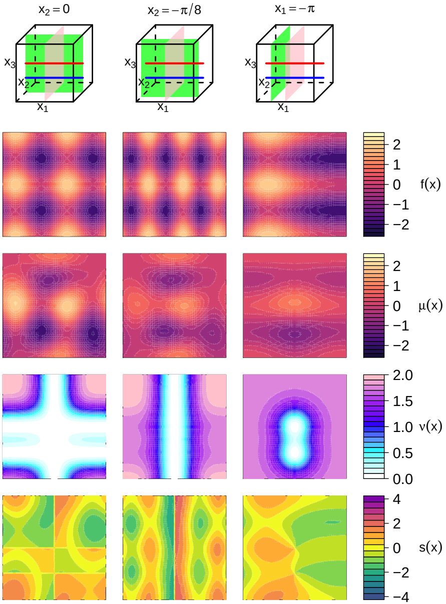
- **Theorem:** The expectation of $f(x)$ adjusted by multiple parallel boundaries $\mathcal{K}_1, .., \mathcal{K}_h$ is given by:

$$
\begin{aligned}
&\mathrm{E}_{K_1 \cup \cdots \cup K_h}[f(x)] \\
&= \mathrm{E}[f(x)] + \mathrm{r}_{1:k_1}(a^{K_1}) \Delta f(x^{K_1}) \\
&+ \sum_{\gamma=2}^{h} \frac{R^{(\gamma-1)}_{k_1,\dots,k_{\gamma-1}}(a^{K_1}, ..., a^{K_{\gamma-1}}, K_1K_\gamma, ..., K_{\gamma-1}K_\gamma)}{R^{(\gamma-1)}_{k_1,\dots,k_{\gamma-1}}(K_1K_\gamma, ..., K_{\gamma-1}K_\gamma, K_1K_\gamma, ..., K_{\gamma-1}K_\gamma)} \mathrm{r}_{k_{\gamma-1}+1:k_\gamma}(a^{K_\gamma}) \\
&\quad * \Bigg(\Delta f(x^{K_\gamma}) + \sum_{j=2}^{\gamma} \sum_{b \subset 1:\gamma, b_1 < \ldots < b_j = \gamma} (-1)^{j+1} \\
&\quad \prod_{l=1}^{j-1} \frac{R^{(b_l-1)}_{k_1,\dots,k_{b_l-1}}(K_1K_{b_j}, ..., K_{b_l-1}K_{b_j}, K_1K_{b_l}, ..., K_{b_l-1}K_{b_l})}{R^{(b_l-1)}_{k_1,\dots,k_{b_l-1}}(K_1K_{b_l}, ..., K_{b_l-1}K_{b_l}, K_1K_{b_l}, ..., K_{b_l-1}K_{b_l})} \\
&\quad * \mathrm{r}_{k_{b_l-1}:k_{b_l}}(K_{b_l}K_{b_{l+1}}) \, \Delta f(x^{K_{b_j} \ldots K_{b_1}}) \Bigg)
\end{aligned}
$$

Extensions: Multiple Boundaries of Differing Dimension



Extensions: Multiple Boundaries of Differing Dimension



Conclusions

- We should seek to identify any known boundaries/Dirichlet boundary conditions for the model $f(x)$ over the input space $\mathcal{X}$.
- If they exist, and are of appropriate form, we can incorporate them into the emulator for negligible computational cost.
- If we know about them in advance we can design sets of runs accordingly.
- $d - k$ dimensional boundaries can be highly informative compared to zero-dimensional runs.

Vernon, I. R., Jackson, S. E. & Cumming, J. A. (2019). "*Known Boundary Emulation of Complex Computer Models*". SIAM/ASA Journal on Uncertainty Quantification, 7 (3), arXiv:1801.03184v2 [stat.ME].

Jackson, Samuel E. & Vernon, Ian (2023). "*Efficient Emulation of Computer Models Utilising Multiple Known Boundaries of Differing Dimension*". Bayesian Analysis 18(1): 165-191, arXiv:1910.08846v2 [stat.ME].

Conclusions

- We should seek to identify any known boundaries/Dirichlet boundary conditions for the model $f(x)$ over the input space $\mathcal{X}$.
- If they exist, and are of appropriate form, we can incorporate them into the emulator for negligible computational cost.
- If we know about them in advance we can design sets of runs accordingly.
- $d-k$ dimensional boundaries can be highly informative compared to zero-dimensional runs.

Vernon, I. R., Jackson, S. E. & Cumming, J. A. (2019). "*Known Boundary Emulation of Complex Computer Models*". SIAM/ASA Journal on Uncertainty Quantification, 7 (3), arXiv:1801.03184v2 [stat.ME].

Jackson, Samuel E. & Vernon, Ian (2023). "*Efficient Emulation of Computer Models Utilising Multiple Known Boundaries of Differing Dimension*". Bayesian Analysis 18(1): 165-191, arXiv:1910.08846v2 [stat.ME].

Conclusions

- We should seek to identify any known boundaries/Dirichlet boundary conditions for the model $f(x)$ over the input space $\mathcal{X}$.
- If they exist, and are of appropriate form, we can incorporate them into the emulator for negligible computational cost.
- If we know about them in advance we can design sets of runs accordingly.
- $d - k$ dimensional boundaries can be highly informative compared to zero-dimensional runs.

Vernon, I. R., Jackson, S. E. & Cumming, J. A. (2019). "*Known Boundary Emulation of Complex Computer Models*". SIAM/ASA Journal on Uncertainty Quantification, 7 (3), arXiv:1801.03184v2 [stat.ME].

Jackson, Samuel E. & Vernon, Ian (2023). "*Efficient Emulation of Computer Models Utilising Multiple Known Boundaries of Differing Dimension*". Bayesian Analysis 18(1): 165-191, arXiv:1910.08846v2 [stat.ME].

Conclusions

- We should seek to identify any known boundaries/Dirichlet boundary conditions for the model $f(x)$ over the input space $\mathcal{X}$.
- If they exist, and are of appropriate form, we can incorporate them into the emulator for negligible computational cost.
- If we know about them in advance we can design sets of runs accordingly.
- $d - k$ dimensional boundaries can be highly informative compared to zero-dimensional runs.

Vernon, I. R., Jackson, S. E. & Cumming, J. A. (2019). "*Known Boundary Emulation of Complex Computer Models*". SIAM/ASA Journal on Uncertainty Quantification, 7 (3), arXiv:1801.03184v2 [stat.ME].

Jackson, Samuel E. & Vernon, Ian (2023). "*Efficient Emulation of Computer Models Utilising Multiple Known Boundaries of Differing Dimension*". Bayesian Analysis 18(1): 165-191, arXiv:1910.08846v2 [stat.ME].

Conclusions

- We should seek to identify any known boundaries/Dirichlet boundary conditions for the model $f(x)$ over the input space $\mathcal{X}$.
- If they exist, and are of appropriate form, we can incorporate them into the emulator for negligible computational cost.
- If we know about them in advance we can design sets of runs accordingly.
- $d - k$ dimensional boundaries can be highly informative compared to zero-dimensional runs.

Vernon, I. R., Jackson, S. E. & Cumming, J. A. (2019). "*Known Boundary Emulation of Complex Computer Models*". SIAM/ASA Journal on Uncertainty Quantification, 7 (3), arXiv:1801.03184v2 [stat.ME].

Jackson, Samuel E. & Vernon, Ian (2023). "*Efficient Emulation of Computer Models Utilising Multiple Known Boundaries of Differing Dimension*". Bayesian Analysis 18(1): 165-191, arXiv:1910.08846v2 [stat.ME].

Conclusions

- We should seek to identify any known boundaries/Dirichlet boundary conditions for the model $f(x)$ over the input space $\mathcal{X}$.
- If they exist, and are of appropriate form, we can incorporate them into the emulator for negligible computational cost.
- If we know about them in advance we can design sets of runs accordingly.
- $d - k$ dimensional boundaries can be highly informative compared to zero-dimensional runs.

Vernon, I. R., Jackson, S. E. & Cumming, J. A. (2019). "*Known Boundary Emulation of Complex Computer Models*". SIAM/ASA Journal on Uncertainty Quantification, 7 (3), arXiv:1801.03184v2 [stat.ME].

Jackson, Samuel E. & Vernon, Ian (2023). "*Efficient Emulation of Computer Models Utilising Multiple Known Boundaries of Differing Dimension*". Bayesian Analysis 18(1): 165-191, arXiv:1910.08846v2 [stat.ME].

References

Vernon, I, Goldstein, M, Rowe, J, Liu, J and 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, arXiv:1607.06358v1 [q-bio.MN].

Danny Scarponi, Andrew Iskauskas, Rebecca A. Clark, Ian Vernon, Trevelyan J. McKinley, Michael Goldstein, Christinah Mukandavire, Arminder Deol, Chathika Weerasuriya, Roel Bakker, Richard G. White, Nicky McCreesh, *Demonstrating multi-country calibration of a tuberculosis model using new history matching and emulation package - hmer*, Epidemics 43 (2023) 100678.

Hu, Baishan, Jaing, Weiguang, Miyagi, Takayuki, Sun, Zhonghao, Ekstrom, Andreas, Forssen, Christian, Hagen, Gaute, Holt, Jason D., Papenbrock, Thomas, Stroberg, S. Ragnar & Vernon, Ian (2022). *Ab initio predictions link the neutron skin of 208Pb to nuclear forces*. Nature Physics 18(10): 1196–1200.

Vernon, I., Owen, J., Aylett-Bullock, J., Cuestra-Lazaro, C., Frawley, J., Quera-Bofarull, A., Sedgewick, A., Shi, D., Truong, H., Turner, M., Walker, J., Caulfield, T., Fong, K. & Krauss, F. (2022). *Bayesian Emulation and History Matching of JUNE*. Philosophical Transactions A 380(2233): 20220039.

Rodrigues, L.F.S., Vernon, I., Bower, R.G.: *Constraints to galaxy formation models using the galaxy stellar mass function, stronger feedback during starbursts?*, MNRAS (2017) 466 (2): 2418-2435. arXiv:1609.06922v3

Vernon, I.; Goldstein, M.; Bower, R. G.; Galaxy Formation: "*Bayesian History Matching for the Observable Universe*". *Statistical Science* 29 (2014), no. 1, 81–90.

Vernon, I., Goldstein, M., and Bower, R. G. (2010), "*Galaxy Formation: a Bayesian Uncertainty Analysis*", *Bayesian Analysis*, 5(4): 619–670, with rejoinder. Invited discussion paper. Awarded Mitchell Prize.

Bower, R., Vernon, I., Goldstein, M., et al. (2010), "*The Parameter Space of Galaxy Formation*", *Mon.Not.Roy.Astron.Soc.*, 407: 2017–2045.

References

Andrianakis, I., Vernon, I., McCreesh, N., McKinley, T.J., Oakley, J.E., Nsubuga, R., Goldstein, M., White, R.G.: *Bayesian history matching of complex infectious disease models using emulation: A tutorial and a case study on HIV in Uganda*. PLoS Comput Biol. 11(1), 1003968 (2015)

Andrianakis, I., McCreesh, N., Vernon, I, McKinley, T. J. Oakley, J. E. Nsubuga, R. Goldstein, M. & White, R. G. (2016). "*History matching of a high dimensional individual based HIV transmission model*". JUQ 5(1):694-719.

McCreesh, N., Andrianakis, I, Nsubuga, R., Strong, M., Vernon, I., McKinley, T.J. Oakley, J.E., Goldstein, M., Hayes, R. & White, R.G. "*Universal Test, Treat, and Keep: Improving ART Retention is Key in Cost-effective HIV Control in Uganda*". BMC Infectious Diseases (2017) 17:322

Trevelyan J. McKinley, Ian Vernon, Ioannis Andrianakis, Nicky McCreesh, Jeremy E. Oakley, Rebecca N. Nsubuga, Michael Goldstein, Richard G. White (2016). "*Approximate Bayesian Computation and simulation-based inference for large-scale stochastic epidemic models*". Statistical Science 33(1): 4–18.

Goldstein, M., Seheult, A., Vernon, I.: "*Assessing Model Adequacy*". In: Wainwright, J., Mulligan, M. (eds.) Environmental Modelling: Finding Simplicity in Complexity, 2nd edn. John Wiley & Sons, Ltd, Chichester, (2013)

Ferreira, Carla, Vernon, Ian, Caiado, Camila, Formentin, Helena, Avansi, Guilherme, Goldstein, Michael & Schiozer, Denis (2019). *Efficient Selection of Reservoir Model Outputs within an Emulation-Based Bayesian History Matching Uncertainty Analysis*. SPE Journal OTC-29801-MS, 1-34.