Projected Sequential Gaussian Processes and Design for Random Ouput Emulation

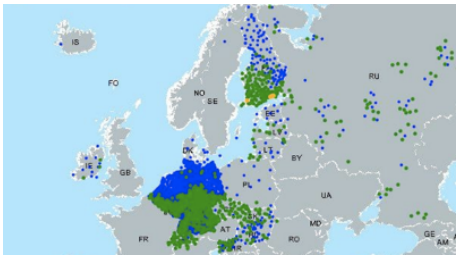
Dan Cornford[1], Remi Barillec, Alexis Boukouvalas

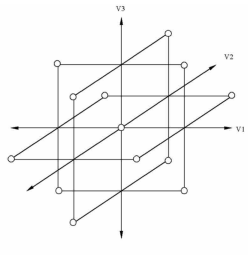
Neural Computing Research Group

MUCM team meeting, Soton, July 2009

[1] http://wiki.aston.ac.uk/DanCornford/

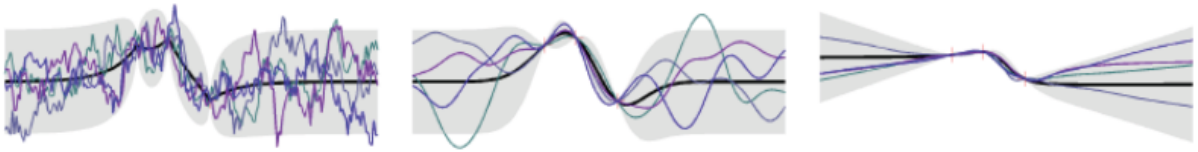
Overview





- Will cover two main areas:
  - How to emulate when we need large numbers of simulator runs?
  - How to select our experimental design when we are emulating random output simulators?
- I will provide a context / motivation for these works first.
- Note: Remi has been working on the INTAMAP project!

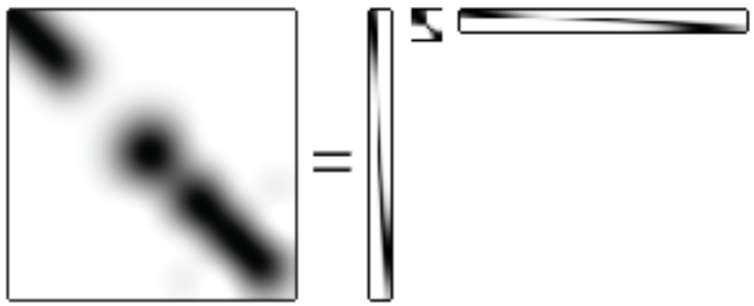
Gaussian Processes for Interpolation



- We all use Gaussian processes for emulation.
- A Gaussian process is specified by a mean function $\mu(x)$, and a covariance function, $c(x, x')$.
- Using Gaussian processes with large simulator design sets (of n design points) raises issues:
  - computational issues - likelihood / Bayesian inference scales as $O(n^3)$ - sampling painful, complex simulators impossible?;
  - inferential issues - full Bayesian treatments produce non-Gaussian process posteriors;
- Remi will show a projected, sequential Gaussian processes implementation in C++ to address computational issues.
- Does not address the inferential issues, since it is a maximum-likelihood / MAP method.

Aston University
Birmingham

Solutions for large n



- Several options are available for dealing with large training sets:
  - Subset the data into smaller regions of interest – border effects;
  - Select nearest n_{max} outputs for local prediction – discontinuities, parameter inference?;
  - Spectral methods – usually require inputs on a regular grid;
  - Sparse matrices – generally require the scale of variation to be short with respect to the overall domain being studied (might work well if the mean function is doing all the work);
  - Low-rank / projected approximations – difficult to determine the low-rank matrix apriori.
- Here we adopt the low rank (projected) approximation, but determine this sequentially.

Aston University
Birmingham

Projected Sequential Gaussian Process

- The basic algorithm is quite simple. The Gaussian process posterior is parametrised using α_i and $C(i,j)$ as:

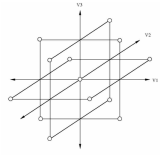
$$\mu_{posterior}(x) = \mu_{prior}(x) + \sum_{i}^{m} \alpha_i c(x, x_i), \text{ and,}$$

$$c_{posterior}(x, x') = c_{prior}(x, x') + \sum_{i,j=1}^{m} c(x, x_i) C(i,j) c(x_j, x')$$

- If m, the size of the active set, is the same as the data set size n, then this is an exact parametrisation.
- Main benefit is a sequential approach to updating the posterior:
  - as each design point is added it is possible (but not necessary) to increase the size of the active set, depending on how much information would be lost;
  - using the expectation propagation method we can recycle the design points.
- Remi will explain and show more ...

Aston University
Birmingham

Experimental design for random output simulators



- Alexis has been working on emulation of random output (stochastic) simulators:
  - Initial work has focussed on modelling the output as a heteroscedastic Gaussian process;
  - This has a separate GP for the mean output and (at an input point) variance of the output.
  - Optimal designs for such settings are not known, since the random (heteroscedastic) output means it might well be efficient to run the simulator a number of times at each design point.
  - Here we consider designs which are informative about model parameters and also designs that are informative for model predictions.
- Alexis will focus on the question of designs for parameter estimation in heteroscedastic GPs.

Aston University
Birmingham

Final remarks

The projected, sequential Gaussian process is ongoing work

The implementation does not address:

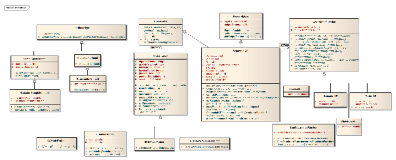
- mean functions (we will be working on that),
- full Bayesian approaches,
- optimisations for compact covariances,
- Remi will show where we are now.

Optimal design for random output emulation is ongoing work

- working with Noha and LSE to link to ASCM,
- early results promising, but still need full results,
- Alexis will show where we are now.

Aston University
Birmingham

Software implementation



- Implementing the psgp library was important to us – we want to extend and reuse it.
- Implementation details can make a big difference to performance.
- Built in C++ on top of the IT++ libraries for matrix / vector operations and Blas for linear algebra - will produce Matlab and R interfaces.
- Class design based on:
  - `CovarianceFunction` *interface* (abstract class, strictly) for the Gaussian process;
  - `LikelihoodType` for the output characteristics (Gaussian in MUCM);
  - The `sequentialGP` implements two *interfaces*, `ForwardModel` and `Optimisable`.
- This will be released on GoogleCode soon.

Aston University
Birmingham