Uncertainty in Computer Models – Part 2

Tony O'Hagan

Outline

- Gaussian processes
- Practical issues in emulator construction
- Calibration
- Dynamic models

Gaussian processes

GP basics

- A Gaussian process is a probability distribution for a function f(x)
  - It is a mathematically convenient distribution for our purposes
- It says that
  - f(x) is normally distributed for every x
  - $f(x_1)$, $f(x_2)$, …, $f(x_n)$ are jointly normally distributed for any set of x_1, x_2, …, x_n
- The GP is specified by giving
  - A mean function m(x) = E(f(x))
  - A covariance function c(x, x') = cov(f(x),f(x'))

GP as a prior distribution

- We use a GP to represent prior beliefs about the computer model f(x)
  - Formally, this is belief about the model prior to making any runs
  - m(x) is our prior expectation of f(x)
  - c(x,x) = var(f(x)) describes our uncertainty about f(x)
  - c(x,x') describes correlation between points x and x'
    - And so represents how smooth the function is
- In practice, we rarely have detailed prior knowledge of this kind
  - Instead, we express prior information just through the general shape of these functions

Hierarchical modelling – m(x)

- Prior information about the general way in which f(x) responds to the inputs x is expressed in a linear modelling form
  - $m(x) = \beta' h(x)$
  - where h(x) is a vector of known functions and β is a vector of unknown regression parameters
- For instance, we often set
  - $h(x) = (1, x)'$
  - This expresses a prior expectation that the model will respond roughly linearly to each input
  - Then the β vector comprises the intercept and slope of the response to each input

Hierarchical modelling – c(x,x')

- The covariance function defines uncertainty and correlation
- The usual formulation is
  - $c(x,x') = \sigma^2 \exp\{-(x-x')'V(x-x')\}$
  - where σ^2 is a variance parameter and V is a matrix of parameters controlling correlation
- In turn, we usually assume V is diagonal, so that there is a correlation parameter for each input
- There are other covariance functions that are sometimes used
  - This form implies that f(x) is differentiable

Hyperparameter distributions

- In this hierarchical framework, we next need to supply distributions for the “hyperparameters”
  - β, σ^2 and (the diagonal elements of) V
- Informative prior distributions may be elicited to express genuine knowledge
  - For instance, by talking to the modellers
- In practice, we generally assume vague prior distributions for all the hyperparameters

The emulator

- Having set up the GP prior distribution, we use the model runs to train it
- The resulting posterior distribution is the emulator
  - Conditional on the hyperparameters, the posterior is a GP
    - But with more complex mean and covariance structure reflecting the training data
  - The training runs are also used to update the distributions for the hyperparameters
    - This is typically rather complex for the correlation parameters in V

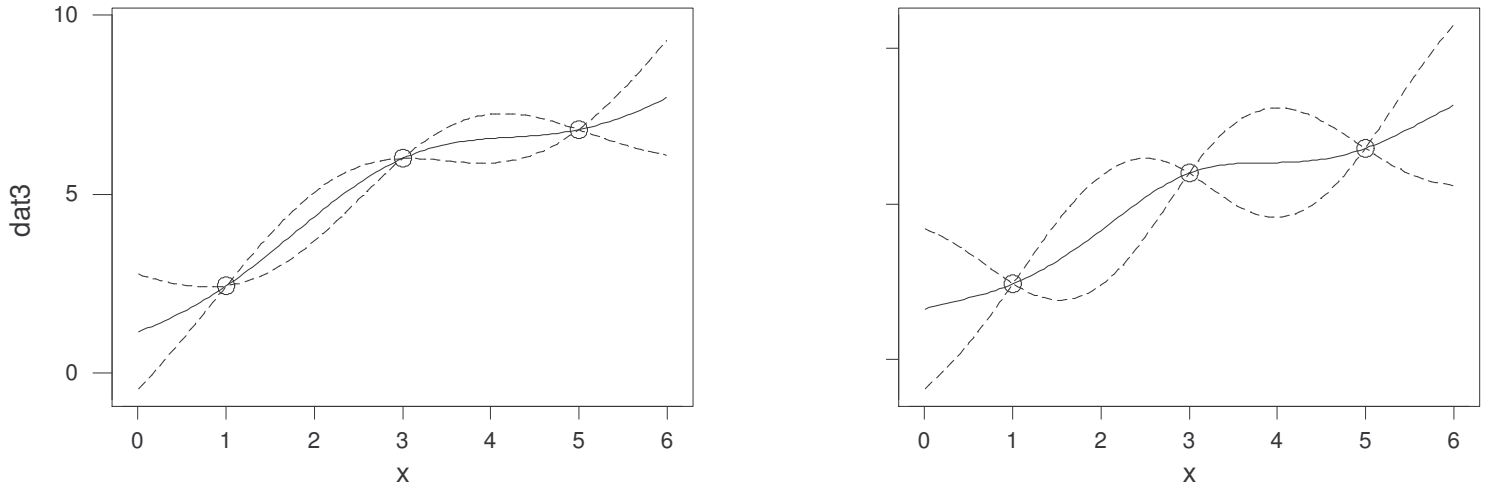
Practical issues in emulator construction

Smoothness

- It is the basic assumption of a (homogeneously) smooth, continuous function that gives the GP its computational advantages
- The actual degree of smoothness concerns how rapidly the function “wiggles”
  - Which in turn is controlled by the correlation length parameters
- A rough function responds strongly to quite small changes in inputs
- We need many more data points to emulate accurately a rough function over a given range

Effect of correlation length

- These parameters determine how fast the uncertainty increases between data points
  - High correlation length = high smoothness



Higher dimensions

- With 2 inputs, we are fitting a surface through the data
- With many inputs, the principles are the same
  - Correlation length parameters (one for each dimension, usually) are crucial
  - Estimation can be difficult, though
- In many dimensions there is much more “space” between data points
  - But we also get more smoothness

Automatic screening

- Models never respond strongly to all their inputs
- Pragmatically, we get a high level of smoothness except in a few dimensions
- By estimating correlation parameters, the GP automatically adjusts
  - Effectively it projects points down through smooth dimensions
  - 200 points in 25 dimensions look sparse
  - But in 5 dimensions they are pretty good
- Models with hundreds or thousands of inputs still pose big computational problems

Design

- We need to choose input configurations at which to run the model to get training data
  - They don’t need to be random
- The objective is to learn about the function
- We need well spaced points that cover the region of interest
- E.g. generate 100 random Latin Hypercube samples and choose the one having largest minimum distance between points

Computational challenges

- Many inputs
  - Need to estimate many correlation parameters
  - Searching the space of correlation parameters becomes computationally burdensome
- Many training runs
  - The method requires inversion of a variance matrix whose dimension is the number of runs
  - This matrix can easily become ill-conditioned with the usual covariance structure
- These problems typically arise together

Inhomogeneity and discontinuity

- The GP assumes homogeneous smoothness and variance properties across the input space
  - Many models have regions of input space where the model behaves differently from others
  - Some models do not vary continuously as the inputs are varied
- In such cases the GP emulator may not fit well

Calibration

Traditional calibration

- Calibration is the process of using observational data to learn about uncertain parameters/inputs in the model
- Traditional approach:
  - Adjust the parameters until the model outputs come as close as possible to the observed data
  - Set the parameters to those best fitting values, acting as if they are now known
- Deficiencies:
  - Ignores uncertainty in fitted parameters
  - Often, best fitting parameter values are implausible

Models and reality

- Notation:
  - Inputs x = (c, t) comprise control inputs c and uncertain parameters t
  - We have n observations of the real process yielding data

$$z_j = \zeta(c_j) + \varepsilon_j \,, \quad \text{for } j = 1, 2, \ldots, n$$

  - Relationship between model and reality

$$\zeta(c) = f(c, \theta) + \delta(c)$$

- The model discrepancy term $\delta(c)$ is crucial
  - Model output does not equal reality, even when we use the best values θ of the uncertain parameters t

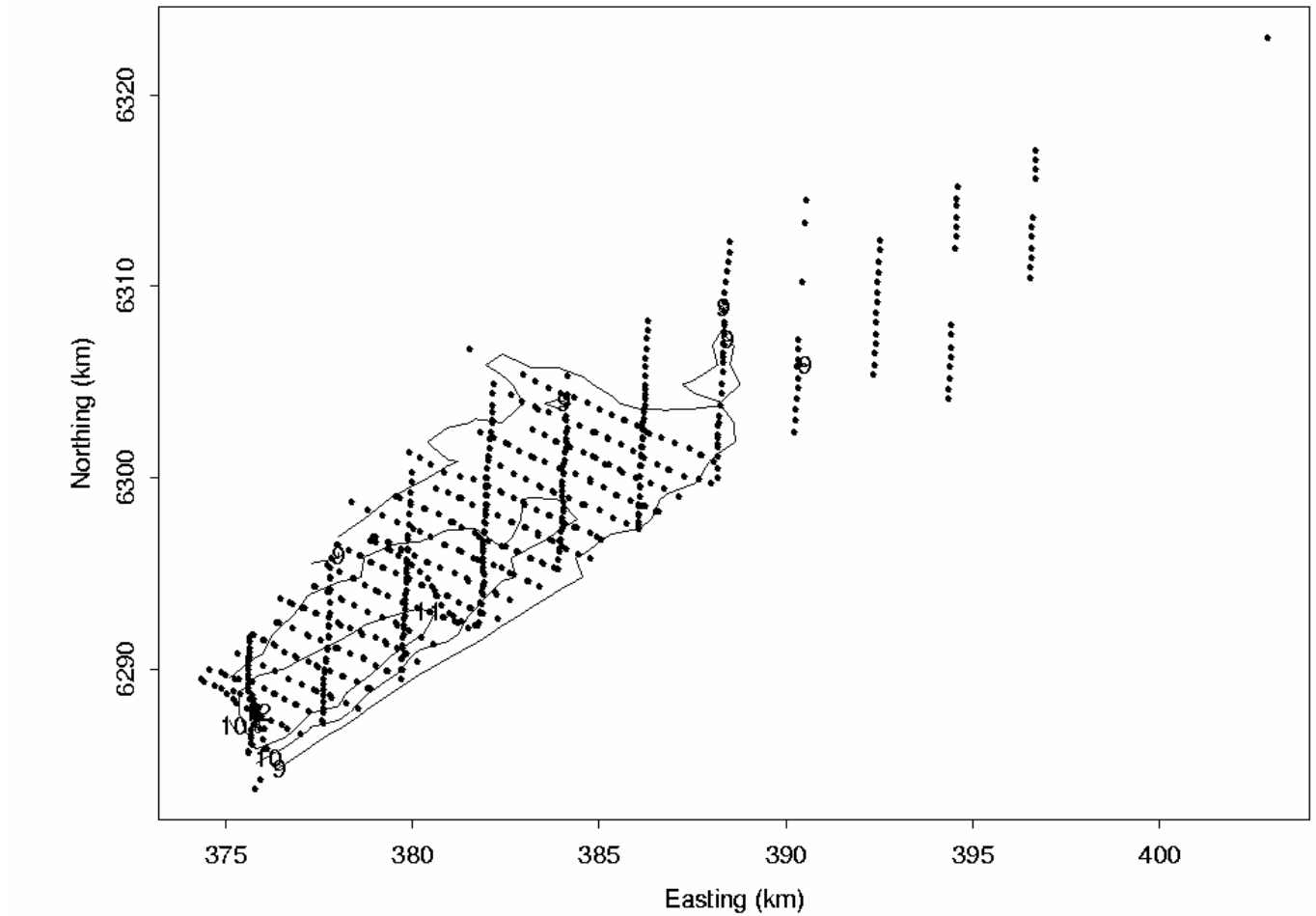
Bayesian method

- Model δ(c) as another GP
- Use observational data to learn about both δ(c) and θ
  - If the computer model is computationally intensive, we may also need to emulate f(x)
- Predictions of the real process allow for remaining uncertainty in θ, ζ(c)
  - and f(x) if we have to emulate it
  - and c if appropriate

Example: Nuclear accident

- Radiation was released after an accident at the Tomsk-7 chemical plant in 1993
- Data comprise measurements of the deposition of ruthenium 106 at 695 locations obtained by aerial survey after the release
- The computer code is a simple Gaussian plume model for atmospheric dispersion
- Two calibration parameters
  - Total release of 106Ru (source term)
  - Deposition velocity

Data


Northing (km)
6290
6300
6310
6320
375
380
385
390
395
400
Easting (km)

Calibration

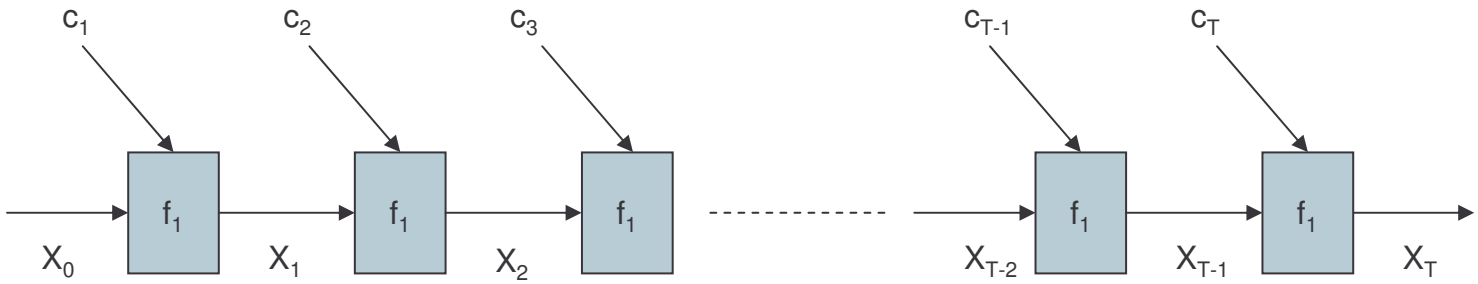
- A small sample (N=10 to 25) of the 695 data points was used to calibrate the model
- Then the remaining observations were predicted and RMS prediction error computed

Sample size N	*10*	*15*	*20*	*25*
Best fit calibration	0.82	0.79	0.76	0.66
Bayesian calibration	0.49	0.41	0.37	0.38

- On a log scale, error of 0.7 corresponds to a factor of 2

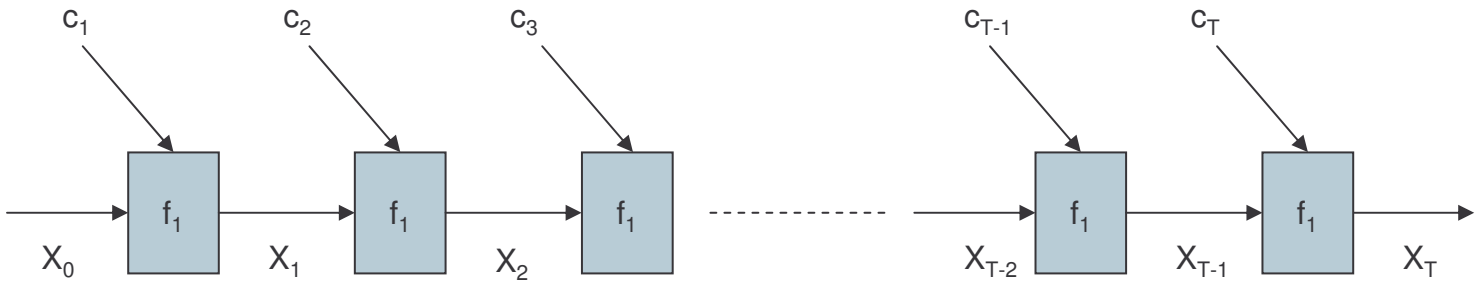
Emulating dynamic models

Dynamic models



- Initial state x_0 is updated recursively by the one-step model $f_1(x,c)$
- Forcing inputs c_t
- Interested in sequence $x_1, x_2, \ldots x_T$
- At least 4 approaches to emulating this

1. Treat time as input

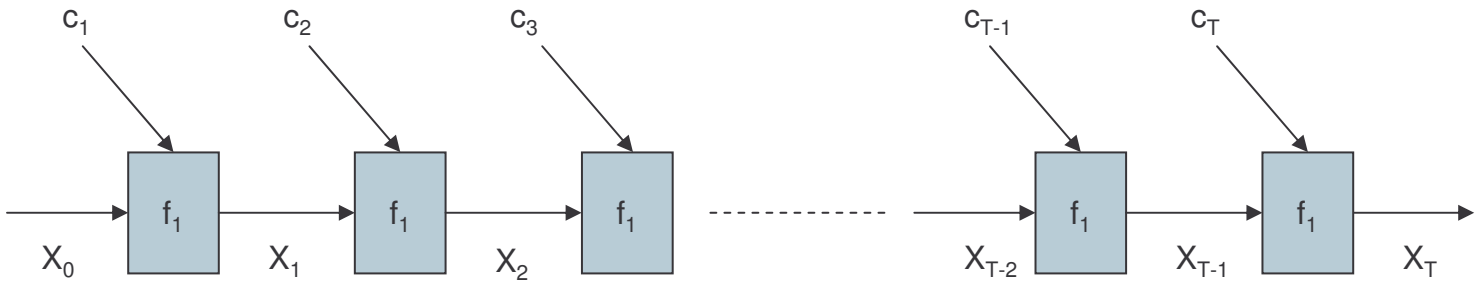


- Emulate x_t as the function

$$f(x_0,t) = f_t(x_0,\mathbf{c}^t) = f_1(x_{t-1},c_t)$$
$$= f_1(\ldots f_1(f_1(x_0,c_1),c_2)\ldots,c_t)$$

- Easy to do
- Hard to get the temporal correlation structure right

2. Multivariate emulation

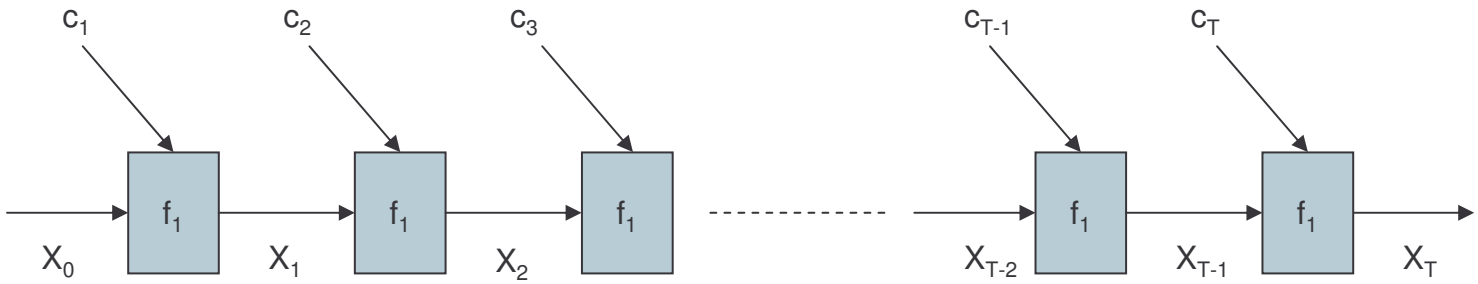


- Emulate the vector $\mathbf{x} = (x_1, x_2, \ldots x_T)$ as the multi-output function

$$\mathbf{f}_T(x_0) = (f_1(x_0,\mathbf{c}^1), f_2(x_0,\mathbf{c}^2), \ldots, f_T(x_0,\mathbf{c}^T))$$

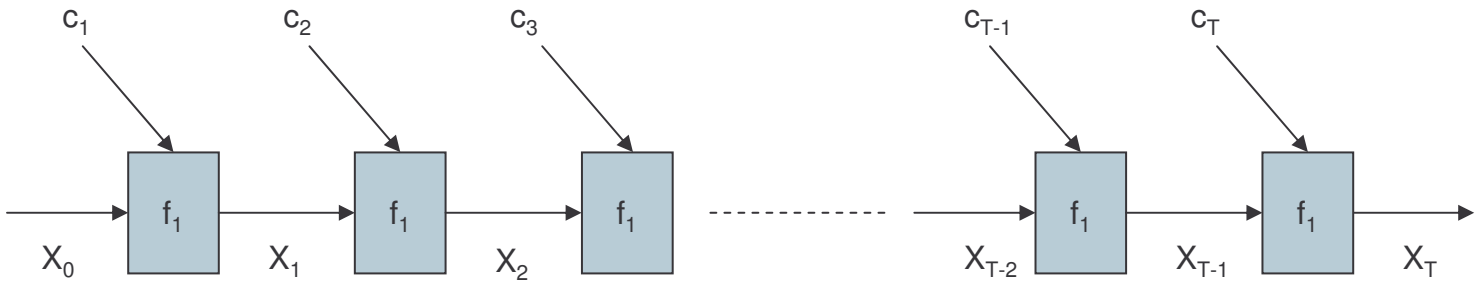
- Simple extension of univariate theory
- Restrictive covariance structure

3. Functional output emulation



- Treat x series as a functional output
- Identify features (e.g. principal components) to summarise function
- Same theory as for multivariate emulation, but lower dimension
- Loss of information, no longer reproduces training data

4. Recursive emulation



- Emulate the single step function

  $f_1(x,c)$

- Iterate the emulator numerically
  - May take longer than original model!
- Or approximate filtering algorithm
  - May be inaccurate for large T

Some comparisons

- Multivariate emulation (approach 2) is better than treating time as an input (approach 1)
  - Validates better
  - Able to work with partial series
- Recursive emulation still under development
  - Looking promising
- Only recursive emulation can realistically treat uncertainty in forcing inputs
- Only recursive emulation can extend T beyond training data

References

- Papers
  - Kennedy, M. C. and O'Hagan, A. (2001). Bayesian calibration of computer models (with discussion). *Journal of the Royal Statistical Society* B 63, 425-464
  - Conti, S. and O'Hagan, A. (2007). Bayesian emulation of complex multi-output and dynamic computer models. *Research Report No. 569/07*, Department of Probability and Statistics, University of Sheffield
  - Higdon, D., Kennedy, M. C., Cavendish, J. C., Cafeo, J. A. and Ryne, R. D. (2004). Combining field data and computer simulations for calibration and prediction. *SIAM Journal on Scientific Computing* 26, 448-466
- Websites
  - mucm.group.shef.ac.uk
  - tonyohagan.co.uk/academic
    - For GEM-Cal, dynamic emulation paper