Uncertainty in Computer Models – Part 1

Tony O'Hagan

Outline

- Background
- Sources of uncertainty
- Uncertainty analysis
- Sensitivity analysis
- UK carbon flux example
- Computation
- Resources

Background

Computer models

- In almost all fields of science, technology, industry and policy making, people use mechanistic models to describe complex real-world processes
  - For understanding, prediction, control
  - Usually implemented in computer codes
- There is a growing realisation of the importance of uncertainty in model predictions
  - Can we trust them?

For instance …

- Models for climate change produce different predictions for the extent of global warming or other consequences
  - Which ones should we believe?
  - What error bounds should we put around these?
  - Are model differences consistent with the error bounds?
- Until we can answer such questions convincingly, governments can continue to dismiss the science

Examples

- Climate prediction
- Molecular dynamics
- Nuclear waste disposal
- Oil fields
- Engineering design
- Hydrology

Sources of uncertainty

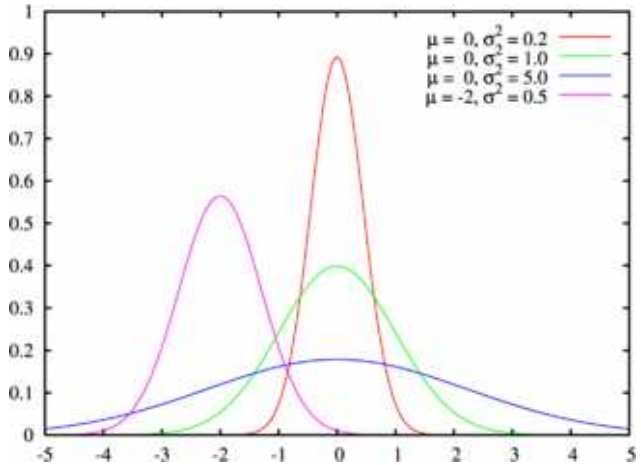
Sources of uncertainty

- A computer model takes inputs x and produces outputs y = f(x)

- How might y differ from the true real-world value z that the model is supposed to predict?
  - Error in inputs x
    - Initial values, forcing inputs, model parameters
  - Error in model structure or solution
    - Wrong, inaccurate or incomplete science
    - Bugs, solution errors

Quantifying uncertainty

- The ideal is to provide a probability distribution p(z) for the true real-world value
  - The centre of the distribution is a best estimate
  - Its spread shows how much uncertainty about z is induced by uncertainties on the last slide



- How do we get this?
  - Input uncertainty: characterise p(x), propagate through to p(y)
  - Structural uncertainty: characterise p(z-y)

Uncertainty analysis

Output uncertainty

- Input uncertainty induces uncertainty in the output y
- It also has a probability distribution
- In theory, this is completely determined by
  - the probability distribution on x
  - and the model f
- In practice, finding this distribution and its properties is not straightforward

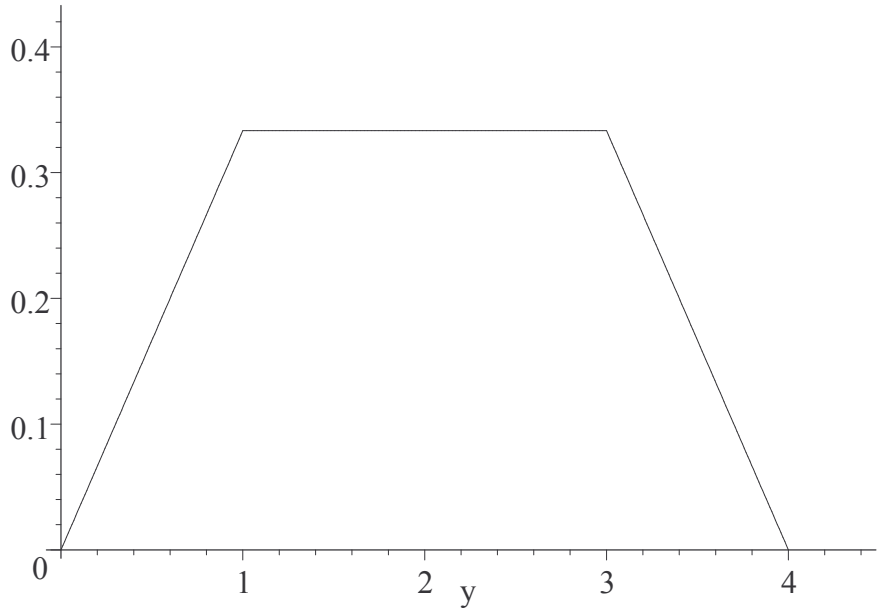
A trivial model

- Suppose we have just two inputs and a simple linear model

$$y = x_1 + 3x_2$$

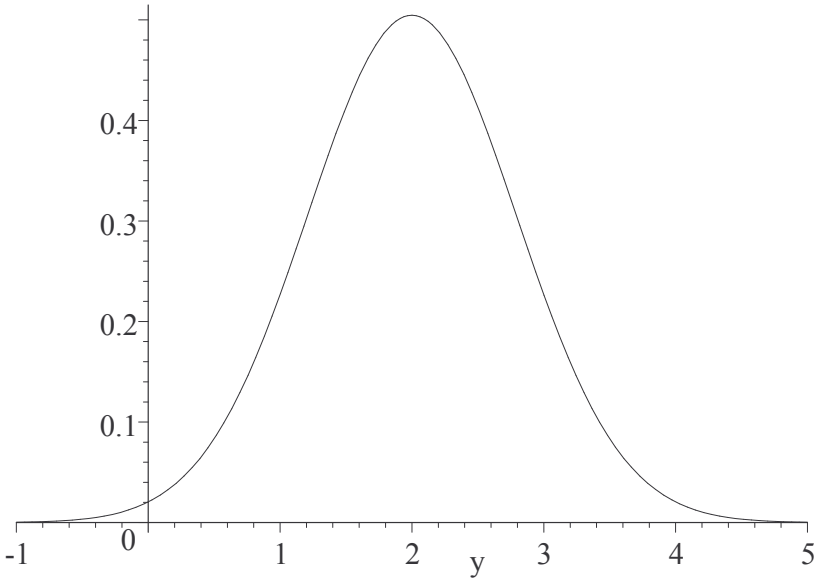
- Suppose that x_1 and x_2 have independent uniform distributions over [0, 1]
  - i.e. they define a point that is equally likely to be anywhere in the unit square
- Then we can determine the distribution of y exactly

Trivial model – output distribution



- The distribution of y has this trapezium form

Trivial model – normal inputs



- If x_1 and x_2 have normal distributions N(0.5, 0.25^2) we get a normal output

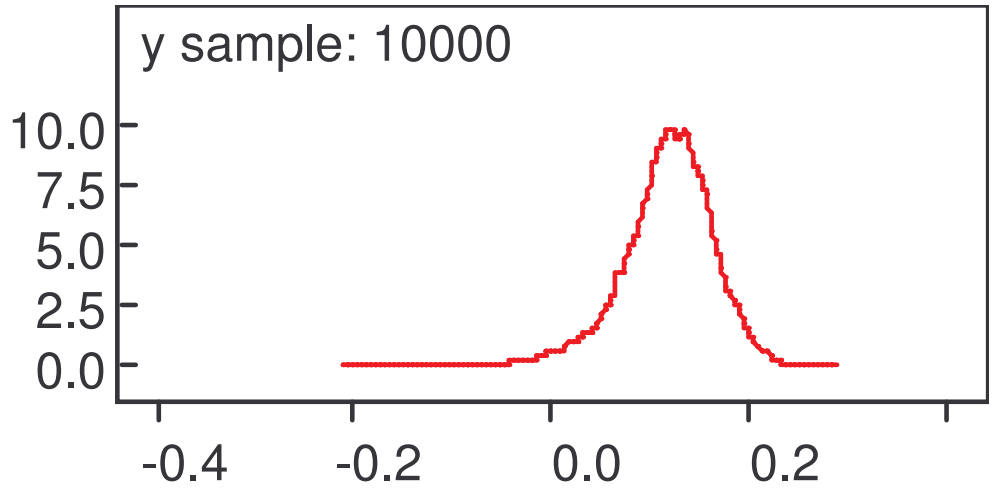
A slightly less trivial model

- Now consider the simple nonlinear model

$$y = \sin(x_1)/\{1+\exp(x_1+x_2)\}$$

- We still have only 2 inputs and quite a simple equation
- But even for nice input distributions we cannot get the output distribution exactly
- The simplest way to compute it would be by Monte Carlo

Monte Carlo output distribution

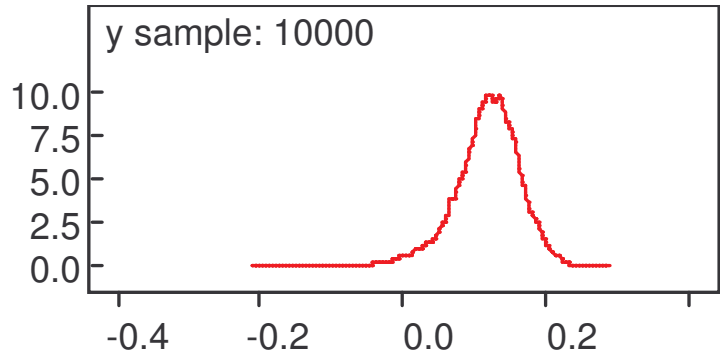


- This is for the normal inputs
- 10,000 random normal pairs were generated and y calculated for each pair

Uncertainty analysis (UA)

- The process of characterising the distribution of the output y is called **uncertainty analysis**
- Plotting the distribution is a good graphical way to characterise it
- Quantitative summaries are often more important
  - Mean, median
  - Standard deviation, quartiles
  - Probability intervals

UA of slightly nonlinear model



- Mean = 0.117, median = 0.122
- Std. dev. = 0.048
- 50% range (quartiles) = [0.092, 0.148]
- 95% range = [0.004, 0.200]

UA versus plug-in

- Even if we just want to estimate y, UA does better than the “plug-in” approach of running the model for estimated values of x
  - For the simple nonlinear model, the central estimates of x_1 and x_2 are 0.5, but

    $\sin(0.5)/(1+\exp(1)) = 0.129$

    is a slightly too high estimate of y compared with the mean of 0.117 or median of 0.122
- The difference can be much more marked for highly nonlinear models

Summary

Why UA?

- Proper quantification of output uncertainty
  - Need proper probabilistic expression of input uncertainty
- Improved central estimate of output
  - Better than the usual plug-in approach

Sensitivity analysis

Which inputs affect output most?

- This is a common question
- Sensitivity analysis (SA) attempts to address it
- There are various forms of SA
- The methods most frequently used are not the most helpful!

Forms of SA

- Local – derivatives
  - Gives no idea of behaviour over a range of inputs
  - No measure of interaction
- One way – vary one input at a time
  - Answer depends on range of variation
  - No way to estimate interactions
- Multi way – vary all inputs, e.g. factorial design
  - Answer still depends on range of variation
  - Complex and requires very many model runs
- Probabilistic – vary according to joint distribution

Variance decomposition

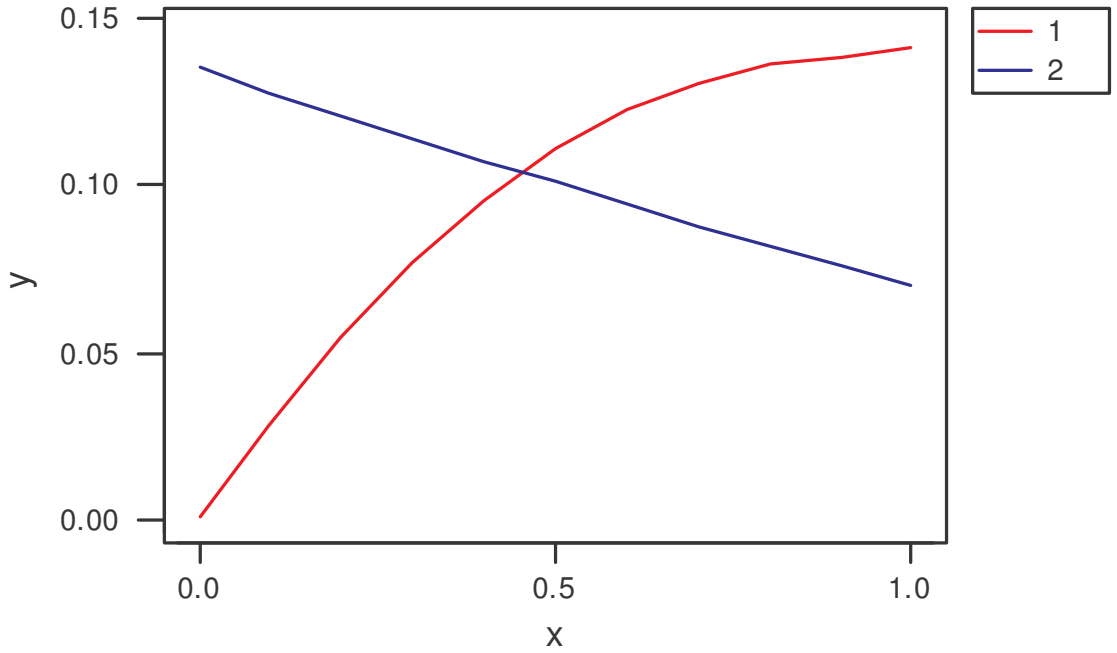
- One way to characterise the sensitivity of the output to individual inputs is to compute how much of the UA variance is due to each input
- For the simple non-linear model, we have

Input	Contribution
X1	80.30 %
X2	16.77 %
X1.X2 interaction	2.93 %

Main effects

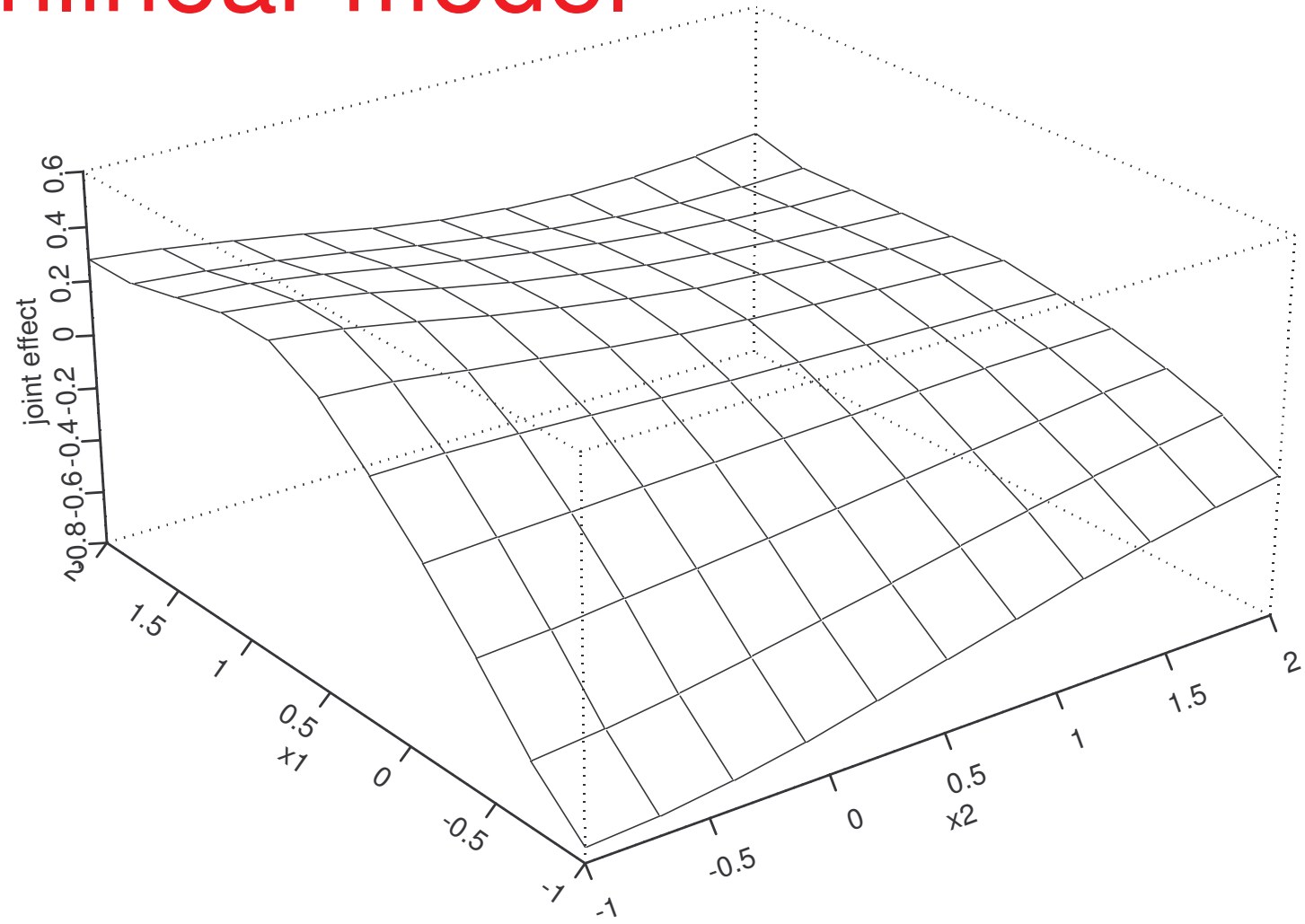
- We can also plot the effect of varying one input *averaged* over the others
- Nonlinear model
  - Averaging $y = \sin(x_1)/\{1+\exp(x_1+x_2)\}$ with respect to the uncertainty in x_2, we can plot it as a function of x_1
  - Similarly, we can plot it as a function of x_2 averaged over uncertainty in x_1
- We can also plot interaction effects

Main effects in the simple nonlinear model



- Red is main effect of x_1 (averaged over x_2)
- Blue is main effect of x_2 (averaged over x_1)

Joint effect in the simple nonlinear model

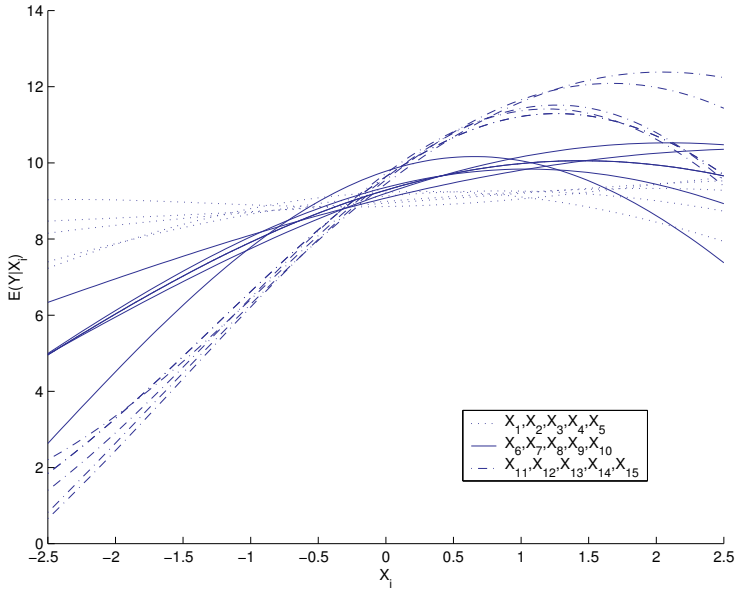


A more complex example

- 5 inputs have appreciable influence, and account for 57% of the total UA variance
- Interactions account for 28%

Amplifying on variances

- Main effect plots amplify on the information given in the variance decomposition
- The variance component associated with input x_i is equal to the amount by which its main effect varies over the range of uncertainty in x_i



The 5 inputs with most influence are dashed

Summary

Why SA?

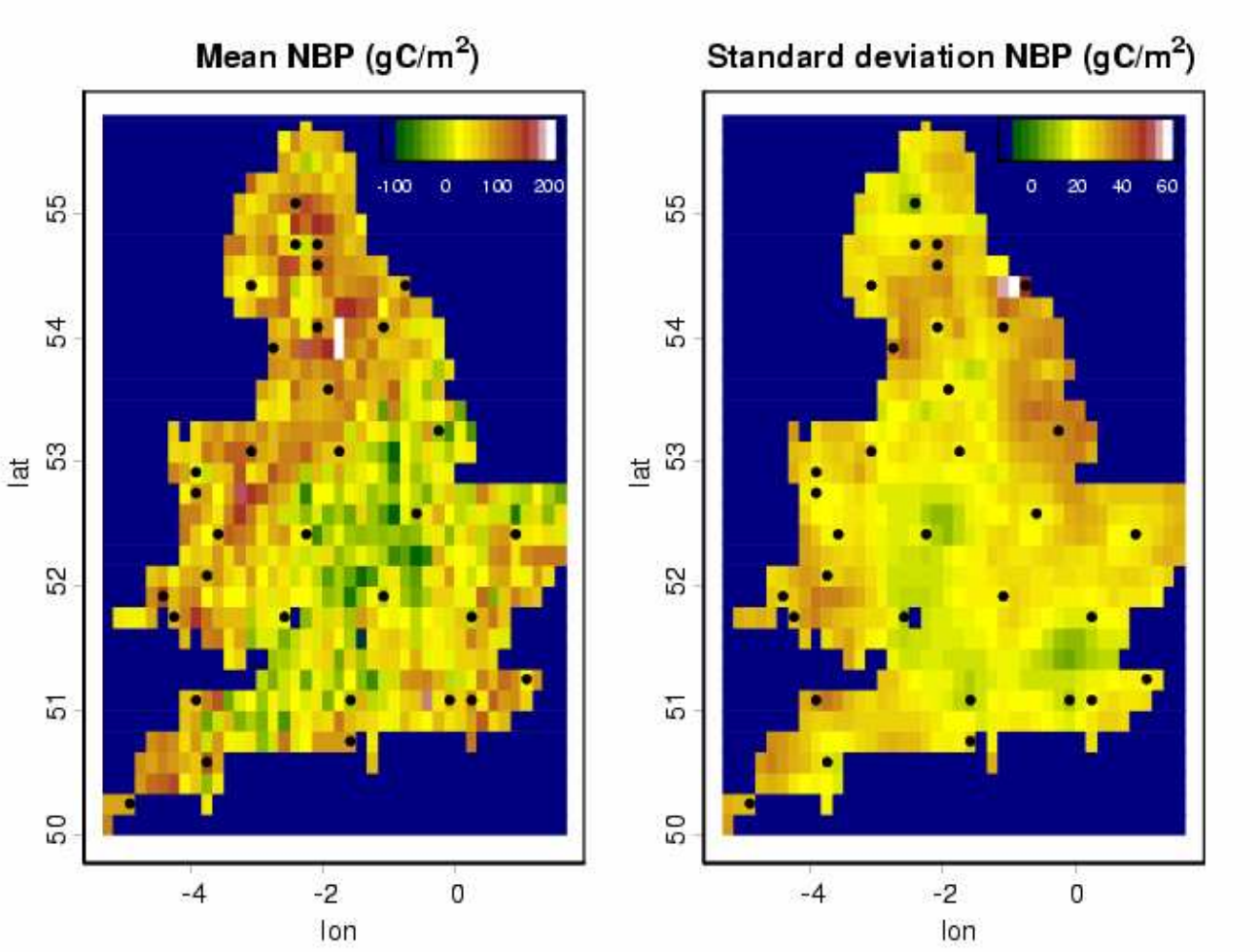
- For the model user: identifies which inputs it would be most useful to reduce uncertainty about
- For the model builder: main effect and interaction plots demonstrate how the model is behaving
  - Sometimes surprisingly!

UK carbon flux example

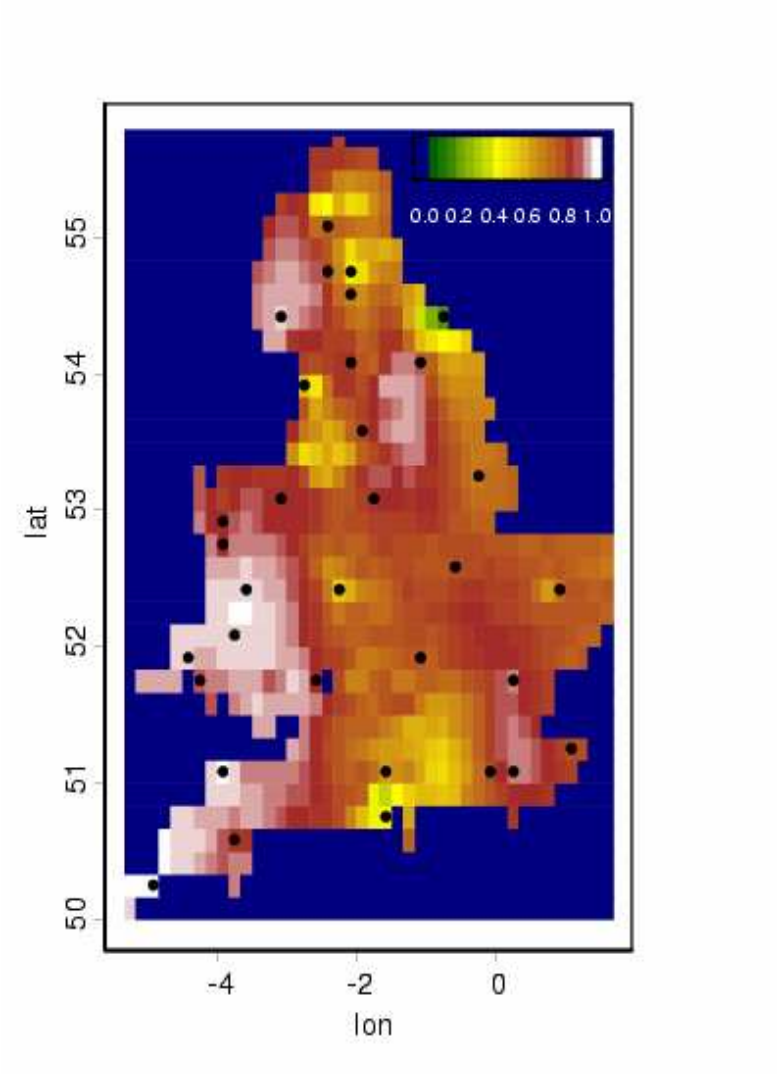
Example: UK carbon flux in 2000

- Vegetation model predicts carbon exchange from each of 700 pixels over England & Wales
  - Principal output is Net Biosphere Production
- Accounting for uncertainty in inputs
  - Soil properties
  - Properties of different types of vegetation
- Aggregated to England & Wales total
  - Allowing for correlations
  - Estimate 7.55 Mt C
  - Std deviation 0.56 Mt C

Maps


Mean NBP (gC/m^2)
-100 0 100 200
lat
55
54
53
52
51
50
-4
-2
0
lon
Standard deviation NBP (gC/m^2)
0 20 40 60
lat
55
54
53
52
51
50
-4
-2
0
lon

Sensitivity analysis



- Map shows proportion of overall uncertainty in each pixel that is due to uncertainty in the vegetation parameters
  - As opposed to soil parameters
- Contribution of vegetation uncertainty is largest in grasslands/moorlands

England & Wales aggregate

PFT	Plug-in estimate (Mt C)	Mean (Mt C)	Variance (Mt C^2)
Grass	5.28	4.64	0.269
Crop	0.85	0.45	0.034
Deciduous	2.13	1.68	0.013
Evergreen	0.80	0.78	0.001
Covariances			0.001
Total	9.06	7.55	0.317

Computation

The problem of big models

- The standard method for computing UA is Monte Carlo (MC)
  - Sample x, run model to get f(x), repeat many times
  - The result is a sample from the output uncertainty distribution
  - Typically needs thousands of model runs
- MC methods for SA require thousands of runs for each variance component
- This is impractical if the model takes more than a few seconds to run
  - We need a more efficient technique

Gaussian process representation

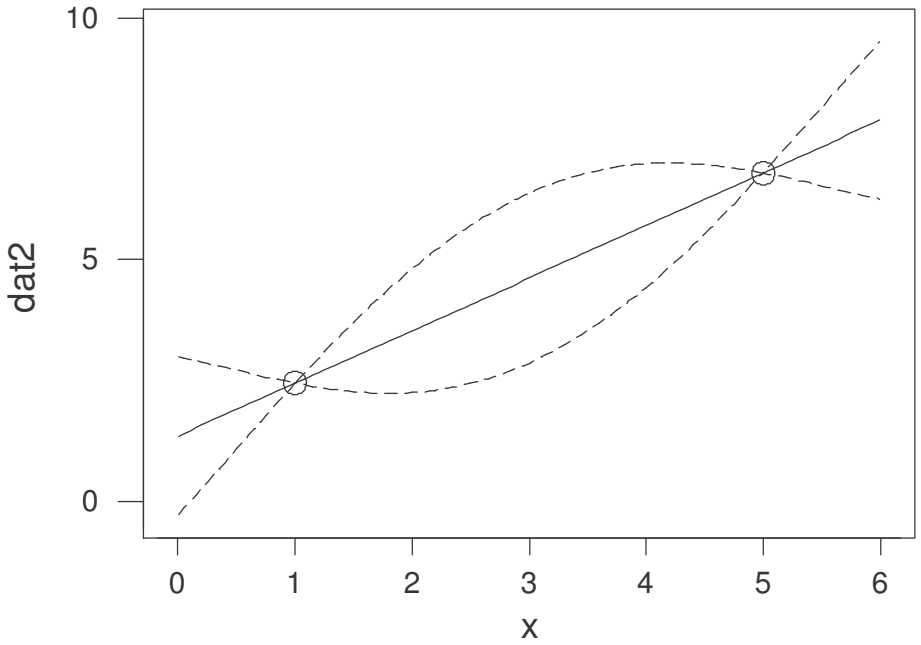
- More efficient approach
  - First work in early 1980s (DACE)
- Consider the code as an unknown function
  - f(.) becomes a random process
  - We represent it as a Gaussian process (GP)
- Training runs
  - Run model for sample of x values
  - Condition GP on observed data
  - Typically requires many fewer runs than MC
    - And x values don’t need to be chosen randomly

Emulation

- Adopting a Bayesian approach
  - Formulate prior distributions for GP hyperparameters
  - Use model runs to estimate hyperparameters
- The posterior distribution is known as an **emulator** of the computer code
  - Posterior mean estimates what the code would produce for any untried x (prediction)
  - With uncertainty about that prediction given by posterior variance
  - Correctly reproduces training data

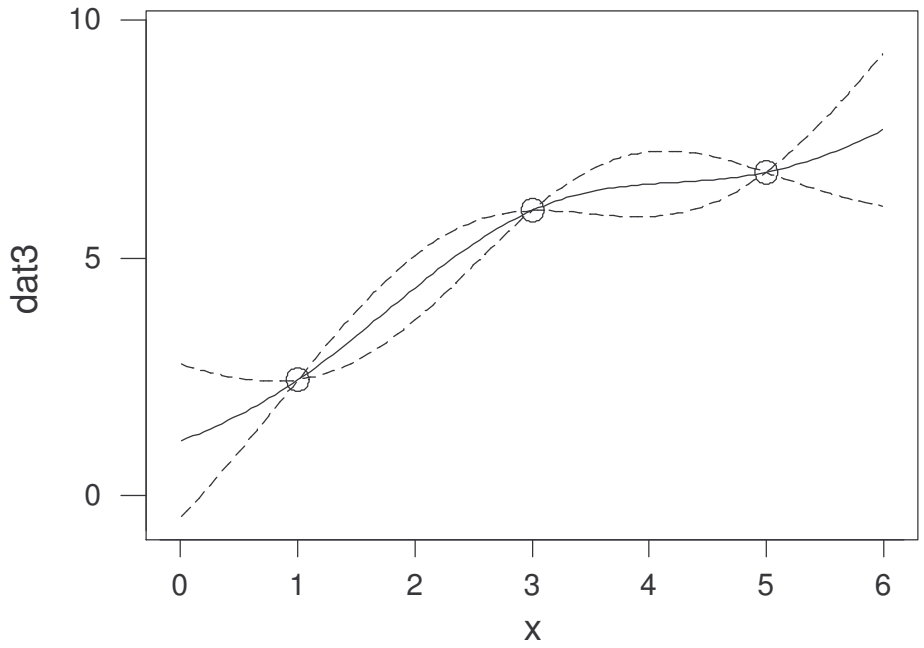
2 code runs

- Consider one input and one output
- Emulator estimate interpolates data
- Emulator uncertainty grows between data points



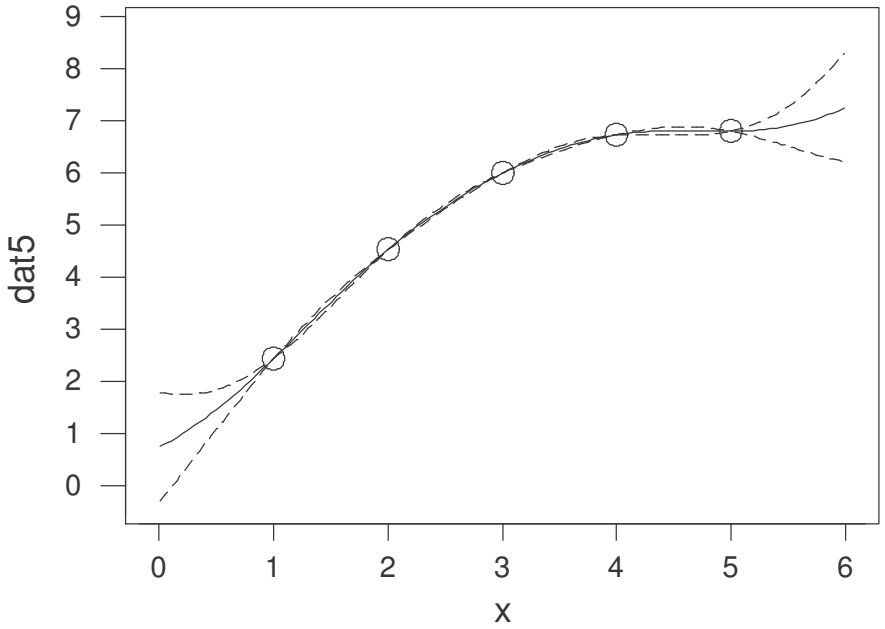
3 code runs

- Adding another point changes estimate and reduces uncertainty



5 code runs

- And so on



Then what?

- Given enough training data points we can emulate any model accurately
  - So that posterior variance is small “everywhere”
- Use the emulator to make (posterior) inference about other things of interest
  - E.g. uncertainty analysis, sensitivity analysis, optimisation
  - Inferences subject to emulation uncertainty
    - Analogous to Monte Carlo error estimates
    - To achieve comparable accuracy with MC would typically require orders of magnitude more model runs

Resources

GEM-SA

- GEM-SA is the first stage of the GEM project
  - GEM = “Gaussian Emulation Machine”
- It uses a set of model runs to build a GP emulator of the computer code
  - Some diagnostics for emulator fit
  - Can also provide suitable designs for model runs
- GEM-SA does UA and SA
  - GEM-Cal includes calibration – beta version available
  - Future stages of GEM will add more functionality

- Managing Uncertainty in Complex Models
  - Large 4-year research grant
    - 7 postdoctoral research assistants, 4 PhD studentships
  - Started in June 2006
  - Based in Sheffield and 4 other UK universities
- Objective:
  - Develop these methods into a robust technology …
    - toolkit
  - that is widely applicable across the spectrum of modelling applications
    - case studies

References

- Papers
  - O'Hagan, A. (2006). Bayesian analysis of computer code outputs: a tutorial. *Reliability Engineering and System Safety* 91, 1290-1300.
  - Haylock, R. G. and O'Hagan, A. (1996). On inference for outputs of computationally expensive algorithms with uncertainty on the inputs. In *Bayesian Statistics 5*, J. M. Bernardo et al (eds.). Oxford University Press, 629-637.
  - Oakley, J. E. and O'Hagan, A. (2004). Probabilistic sensitivity analysis of complex models: a Bayesian approach. *Journal of the Royal Statistical Society* B 66, 751-769.
  - Kennedy, M. C., O'Hagan, A., Anderson, C. W., Lomas, M., Woodward, F. I., Heinemeyer, A. and Gosling, J. P. (2006). Quantifying uncertainty in the biospheric carbon flux for England and Wales. *Journal of the Royal Statistical Society, Series A* (in press).
- Websites
  - tonyohagan.co.uk/academic
  - mucm.group.shef.ac.uk