Setting Up Your Simulator

Jonathan Rougier

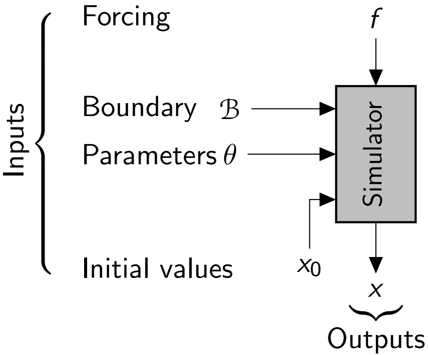
School of Mathematics
University of Bristol, UK

Cabot/CREDIBLE Uncertainty workshops, Oct 2015

http://www.maths.bris.ac.uk/~mazjcr/#SUYS

The kitchen sink simulator

The 'kitchen sink' simulator, which has all of the features that may be found in a simulator.[1] Each of these symbols may represent a large collection of values.



We focus on parameters as the least-well-defined components of the inputs.

[1] Your simulator may be missing some of the input components.

Brief summary of the clickers from the first workshop

The slides from the first workshop are available at `https://www.bris.ac.uk/engineering/research/local/model-uncertainty/`

- Simulator run-time: 40% seconds-to-minutes; 55% hours-to-days.
- Number of parameters: 43% less than 10, but quite a few spatially-distributed fields of parameters.[2]
- Access to the source code: nearly 90%.
- Availability of observations: 80%.
- Mode of parameter tuning (where performed): manual adjustment 65%, automatic adjustment 35%.

---

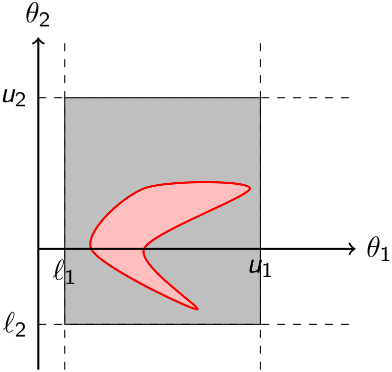
[2] Spatially-distributed parameters need to be treated carefully! I come back to this at the end.

The parameter box

We aim to construct a box for the p parameters, designated

$$\mathcal{C} := (\ell_1, u_1) \times \cdots \times (\ell_p, u_p) \subset \mathbb{R}^p.$$

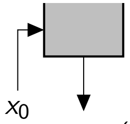
Values outside the box are ruled out.



Some values inside the box are 'possibly good'; some may have been ruled out. Smaller boxes are better.

Observables and observations

Observables and observations are used to set the limits of the parameter box.



$$x = (x_1, \ldots, x_m)$$

$$\left.\begin{array}{ll} \text{Observables} & y_j = g_j(x_0, x_1, \ldots, x_m) \\ \text{Observations} & y_j^{\text{obs}} \text{ (measured values)} \end{array}\right\} \quad j = 1, \ldots, n$$

When focusing on the role of the parameters, I write

$$y_j(\theta) := g_j\big(x_0, x_1(\theta), \ldots, x_m(\theta)\big).$$

What makes a good observable?

1. y_j corresponds to a (well-) measured system property.
2. y_j is similar to decision-relevant outputs.
3. You think y_j is sensitive to some of the important parameters,
4. and yet it's different from other observables you might be considering.
5. You can quantify y_j's 'tolerance' as the value Δ_j.

Tolerance: Δ_j is the largest difference you can tolerate between $y_j(\theta)$ and y_j^{obs} before ruling out θ as a 'possibly good' parameter value.

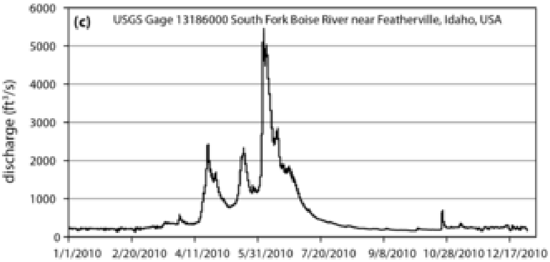
Reality check! If you and your collaborators cannot provide one or more tolerances, then you have no justification for using your simulator to make inferences about the underlying system.

Quantifying Δ_j is the difference between being a modeller and being a scientist.

What makes a good observable? (cont)

TOP TIP — Make a careful choice of a few key observables. Think about using non-linear functions of the outputs.

For example, don't just use a sequence of points in a field, e.g. times in a time-series, or locations in a 2D domain.



Q: What can my simulator get right? What does the Mayor care about? What are the key parameters? Can I split them across observables? What tolerances can I quantify?

Sanity check

TOP TIP — You will learn a lot from trying to break your simulator, based on your initial assement of the parameter ranges.

θ_1	θ_2	θ_3	θ_4	θ_5	y_1
0.5	0.5	0.5	0.5	0.5	1.75
0	0.5	0.5	0.5	0.5	−0.88
1	0.5	0.5	0.5	0.5	2.12
0.5	0	0.5	0.5	0.5	2.25
0.5	1	0.5	0.5	0.5	1.25
0.5	0.5	0	0.5	0.5	1.62
0.5	0.5	1	0.5	0.5	1.87
0.5	0.5	0.5	0	0.5	1.87
0.5	0.5	0.5	1	0.5	1.62
0.5	0.5	0.5	0.5	0	1.62
0.5	0.5	0.5	0.5	1	1.87
0	1	0	1	0	−2.00
1	0	1	0	1	4.00

You will need to figure out what to do about the `NA`'s . . .

Implausibility

- We represent the tolerance Δ_j as $3\sigma_j$, where σ_j is a standard deviation. If it is helpful, this standard deviation can be built up using

$$\sigma_j := \sqrt{\sigma^2_{j,\mathrm{obs}} + \sigma^2_{j,\mathrm{repr}} + \sigma^2_{j,\mathrm{str}}},$$

  i.e. 'observation', 'representation', and 'structural'. Of these, the third is often the largest, because it captures the limitations of the simulator as a quantitative description of the system.

- A parameter value θ is implausible under observable j if $y_j(\theta)$ is too far away from the corresponding observation y_j^{obs}. Its measure is

$$I_j(\theta) := \frac{\left|y_j^{\mathrm{obs}} - y_j(\theta)\right|}{\sigma_j}. \qquad \text{(implausibility)}$$

Implausibility (cont)

- 'Too far away' can be defined using a statistical criterion:

$$\mathcal{C}_j := \left\{\theta \in \mathbb{R}^p : I_j(\theta) \leq 3\right\}$$

  is a 95% confidence set for the 'best' parameter value θ^*.[3]

- The same statistical criterion explains how to combine implausibilities for a set of observables J:

$$I_J(\theta) := \max_{j \in J} I_j(\theta) \qquad \mathcal{C}_J := \left\{\theta \in \mathbb{R}^p : I_J(\theta) \leq 3\right\}.$$

- And also how to project implausibilities down into a subset of the parameters, including a single parameter:

$$I_J(\theta_k) := \min_{\theta_{-k}} I_J(\theta_k, \theta_{-k}) \qquad \mathcal{C}_{Jk} := \left\{\theta_k \in \mathbb{R} : I_J(\theta_k) \leq 3\right\}$$

  where θ_{-k} is all of the parameters except θ_k.

---

[3] Term and conditions apply.

Implausibility (cont)

TOP TIP Accept this generous gift from Statistics!

- At this stage, don't go making up your own scoring function, or making up your own combination and projection operations.
- Define the parameter box based on observables J as

$$\mathcal{C} = \mathcal{C}_{j1} \times \cdots \times \mathcal{C}_{Jp}.$$

- So now our setting-up has simplified to:
  1. choosing a set of observables J,
  2. running the simulator to compute implausibilities $I_J(\cdot)$ for specified parameter values, and
  3. figuring out what these tell us about $\mathcal{C}_{Jk}$ for $k = 1, \ldots, p$,

  and repeating as necessary.

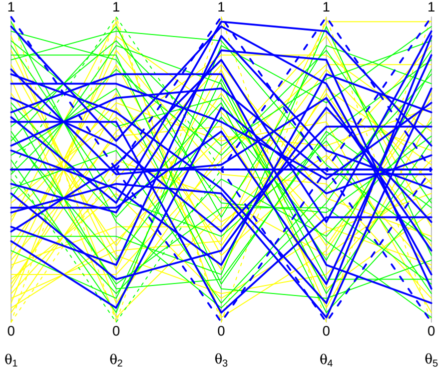
One set of runs of your simulator

$$
\begin{array}{lccccccc}
 & \multicolumn{3}{c}{\text{Parameter values}} & & \multicolumn{3}{c}{\text{Observable values}} \\
\text{Run 1} & \theta_1^{(1)} & \dots & \theta_p^{(1)} & \longrightarrow & y_1(\theta^{(1)}) & \dots & y_m(\theta^{(1)}) \\
\text{Run 2} & \theta_1^{(2)} & \dots & \theta_p^{(2)} & \longrightarrow & y_1(\theta^{(2)}) & \dots & y_m(\theta^{(2)}) \\
\text{Run 3} & \theta_1^{(3)} & \dots & \theta_p^{(3)} & \longrightarrow & y_1(\theta^{(3)}) & \dots & y_m(\theta^{(3)}) \\
\text{Run 4} & \theta_1^{(4)} & \dots & \theta_p^{(4)} & \longrightarrow & y_1(\theta^{(4)}) & \dots & y_m(\theta^{(4)}) \\
 & \vdots & \ddots & \vdots & & \vdots & \ddots & \vdots \\
\text{Run n} & \underbrace{\theta_1^{(n)} \quad \dots \quad \theta_p^{(n)}}_{\text{Design matrix}} & & & \longrightarrow & \underbrace{y_1(\theta^{(n)}) \quad \dots \quad y_m(\theta^{(n)})}_{\text{Observable matrix}} & &
\end{array}
$$

$$
\begin{array}{lccc}
\text{Observations:} & y_1^{\text{obs}} & \dots & y_m^{\text{obs}} \\
\text{Std. deviations:} & \sigma_1 & \dots & \sigma_m
\end{array}
$$

- This is all of the information you need for the next stage.

A Parallel Coordinates Plot (PCP)



- Drawn for a specific set of observables J and a specific number of runs. It's important that these runs form a space-filling design.
- My colour scale: blue $= I_J(\theta) \leq 3$, green $=$ 3–6, yellow $>$ 6.
- 'Non-blue' region on axis k indicates the complement of $\mathcal{C}_{Jk}$.

Running the simulator

Calculations (described in the paper) will tell you roughly how many runs you need to do in each 'wave' of the setting-up process.

TOP TIP CPU cycles and your time are both valuable. Use an open-ended space-filling design and consult your diary.

- A open-ended space-filling design like a Sobol sequence. More runs are always welcome, so your simulator should be running up until the time you have set aside to analyse the results.
- Proceed in 'waves'. You will be much more successful making a series of adjustments to a few parameters at a time, than trying to do the whole setting-up in one go.

Crudely, you should aim to 'use up' observables in an efficient order. There is a semi-automated method to achieve this.

Brief summary

1. Always start your experiment with a sanity check.
2. Identify a small but powerful set of observables.
3. Run your simulator with an open-ended space-filling design.
4. Proceed in waves to pick the low-hanging fruit first.
5. Use a parallel coordinate plot with implausibility colour-scale to refine your ranges.
6. Specifying the tolerances is hard: don't be ashamed to revisit your choices.

More resources

- Find these slides and the current version of the paper at

  `http://www.maths.bris.ac.uk/~mazjcr/#SUYS`

- There is R code for the calculations at the same location.
- This is only half of a longer paper. The second half covers:
  1. Last gasp design: putting runs into the 'possibly good' region.
  2. Fields of parameters, like a spatial field of Manning's n values. These need to be handled differently in space-filling designs.
  3. Stochastic simulators, where two runs at the same parameter values do not necessarily give the same output.
  4. Expensive simulators, where sometimes an *emulator* can help.

  There is also a Glossary.
- By all means contact me if you are struggling.

After lunch . . .

We will work though the R worksheet individually or in small groups, with regular pauses for catching up and discussion.