Plotting, Checking, and Calibrating Emulators

Andrew Parnell and Philip Cardiff
andrew.parnell@mu.ie

Maynooth University
National University of Ireland Maynooth

https://github.com/andrewcparnell/intro_emulators

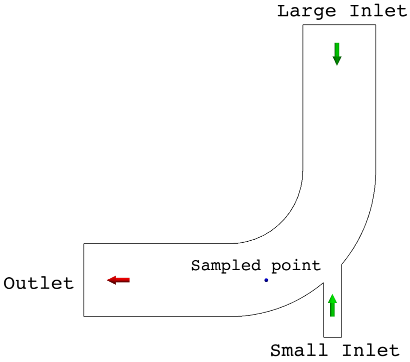
Introduction

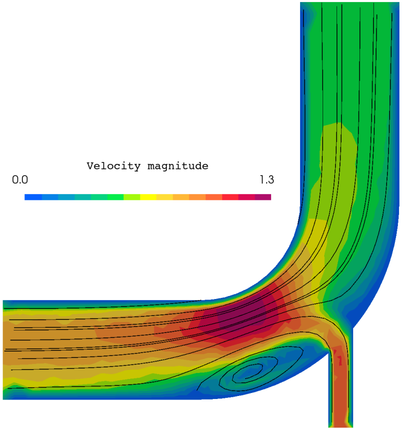
- We now have a shiny and very fast emulator for our simulator
- But how do we know whether it works well?
- In this section we will go through some simple plots that work in both low and high dimensional problems
- We will then discuss some performance measures that enable us to judge whether the emulator is fitting well or not

It is helpful for this section to revise the use of `ggplot2`; see the Rfternoon course for a revision class

Reminder: Navier-Stokes example

Recall our 2D Navier-Stokes example with 3 inputs and 1 output:

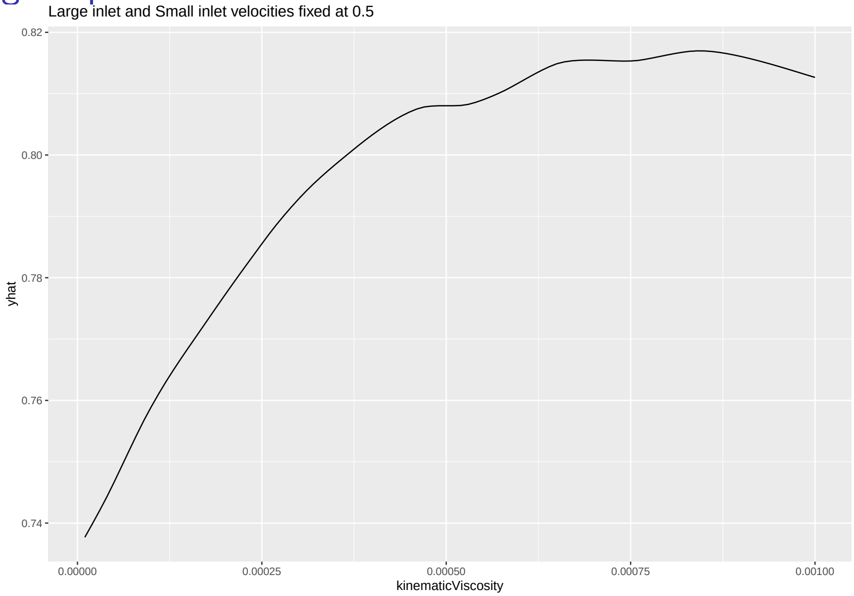




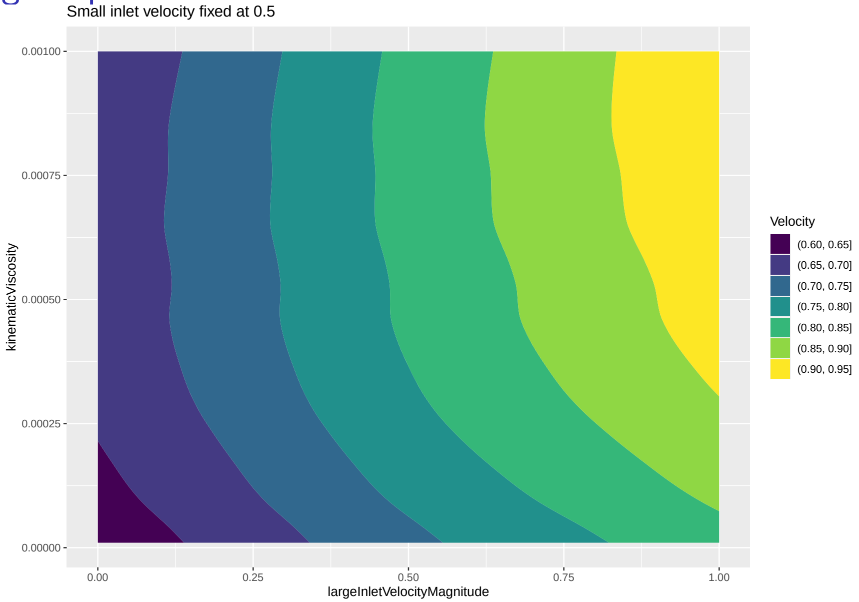
Plotting the output

- One way of creating a neat plot is to fix all the variables but one or two
- We can then produce a plot of the remaining variable(s) changing across its range of values according to the emulator
- Suppose for example that we were just interested in the variable `kinematicViscosity`, we could fix the others at chosen values (e.g. 0.5) and then predict across the chosen variable
- We could extend this to look at two dimensions whilst holding the other fixed
- With further plotting magic we could add slider bars or an animation to look at the effect of changing the input values

A 1D marginal plot



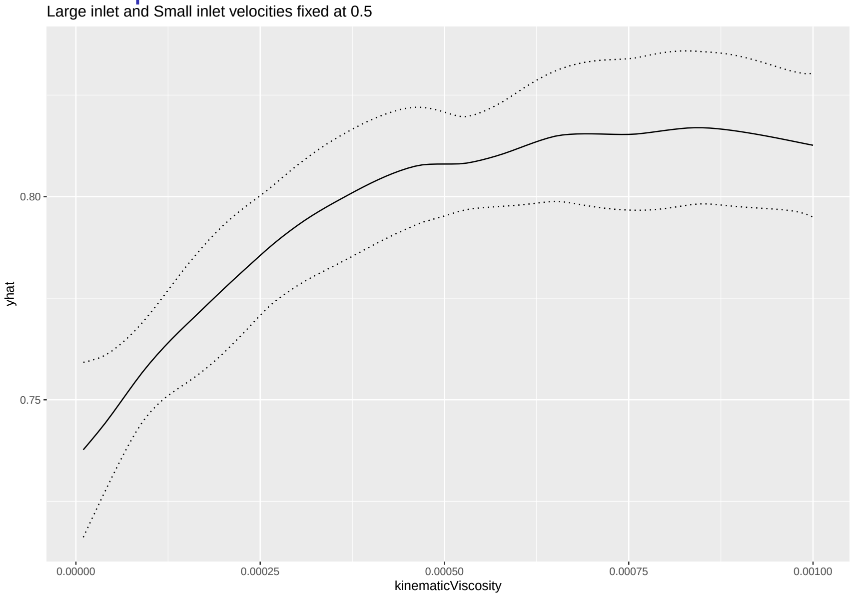
A 2D marginal plot



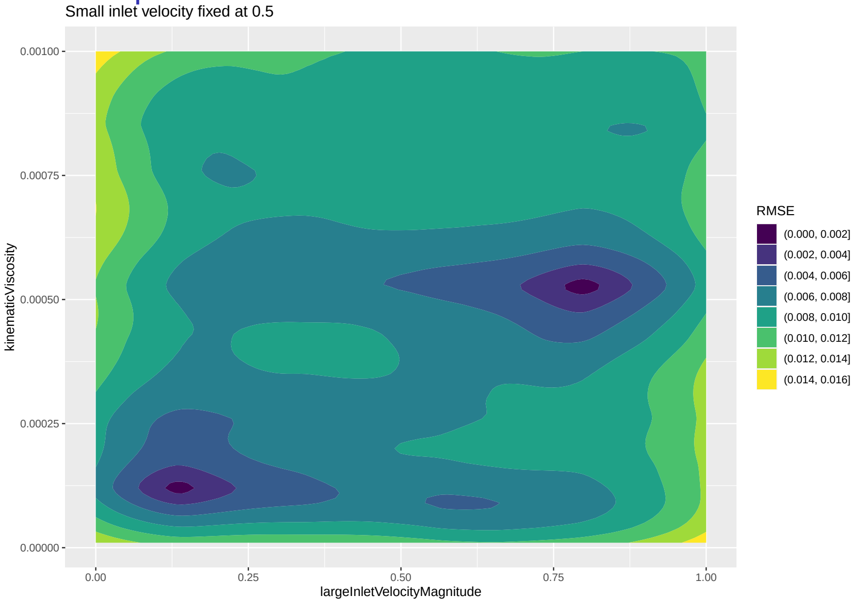
Further plots

- These marginal plots give us some idea as to the behaviour of a subset of the variables when others are fixed
- But the Gaussian Processes also produces an estimate of the error and we can use this too
- We can plot these alongside the marginal plots to give an idea of places where the emulator is performing poorly

1D marginal error plot



2D marginal error plot



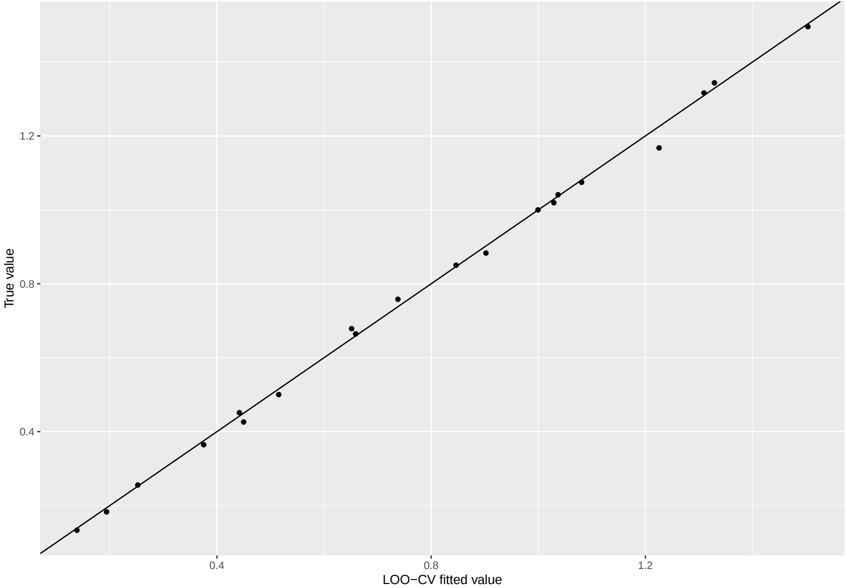
Cross validation

- An even better way of judging whether the emulator is working is to use **Cross Validation (CV)**
- This works by removing some of the data that the model was trained on, re-fitting the emulator, and predicting for the left out observations
- If the emulator can predict the left out observations well then it is doing a good job
- We usually plot the predictions from the left out data against the true observations. If they follow an identity relationship then the emulator is working well

Leave one out CV code

```
loo_fit <- rep(NA, n_runs)
for (i in 1:n_runs) {
  print(i)
  # Fit an emulator with one point missing
  curr_emulator <- GP_fit(
    initial_grid[-i, ],
    df_grid$out[-i]
  )
  # Get the prediction for the missing point
  loo_fit[i] <- predict(
    curr_emulator,
    initial_grid[i, , drop = FALSE]
  )$Y_hat
}
```

Leave one out CV results



Calibrating the emulator will real-world observations

- Sometimes in addition to the emulator values we also have real-world on the process
- We thus want to check whether the real world data match the emulator and whether there are systematic biases
- The usual equation people fit is:

$$y(X) = m(X) + \delta(X) + \epsilon$$

- Where X are our inputs and y our output
- m is the emulator (already fitted)
- $\delta(x)$ is another GP representing systematic bias
- ϵ is a pure noise error term representing observation error
- You will need to take another module (on Bayesian Modelling) to fit these kind of models!

Summary

- We can create marginal 1D and 2D plots to check how well our emulator fits
- We can also run cross-validation to check out of sample performance
- Next (and finally): deploying the emulator and making it useful