# Introduction to Emulators

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


https://andrewcparnell.github.io/intro_emulators/

# Introduction

- In this course I aim to teach you how to start with a complicated and slow mathematical model and create a very fast shortcut to it
- This statistical shortcut is known as an **emulator** and is built using machine learning techniques
- If your emulator is really good you never need your original mathematical model
- Examples include: climate models, power plants, bombs, biological systems, etc
- The course is broken into 5 parts: designing, building, checking, deploying, extending

# Terminology

- The original mathematical model you want to emulate is called a **simulator**
- The simulator has inputs ($X$) and outputs ($Y$)
- The shortcut to the simulator is called an **emulator**
- If the simulator $f$ gives you $Y = f(X)$, then the emulator gives you $\hat{Y} = \hat{f}(X)$
- The statistical method we use to fit the model is called a **Gaussian Process**
- Emulators are sometimes called **code surrogates**
- The general topic is often known as **uncertainty quantification (UQ)** or **design and analysis of computer experiments (DACE)**

# A simulator

Here is a simple simulator:

```
f <- function(x1, x2) {
  return(10 * sin(pi * x1 * x2))
}
```

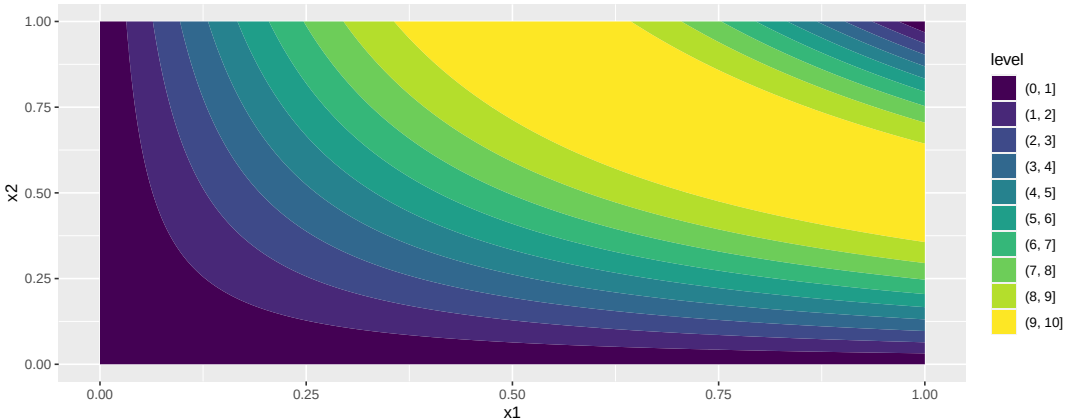
We will define it on $x_1, x_2 \in (0, 1)$ and use it with:

```
f(0.7, 0.7)
```

```
## [1] 9.995066
```

(All the code for the examples we use is in the `code` folder on GitHub)

# Understanding the simulator



# Assumptions

- You can run the simulator a few times but now too many as they are usually slow to run (which is why you want an emulator)
- The outputs of the simulator are identical for the same inputs; it is a **deterministic** simulator
- The outputs of the simulator are smooth, i.e. similar for similar values of the inputs

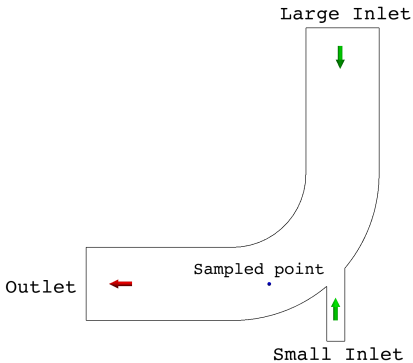
# Overview of the process

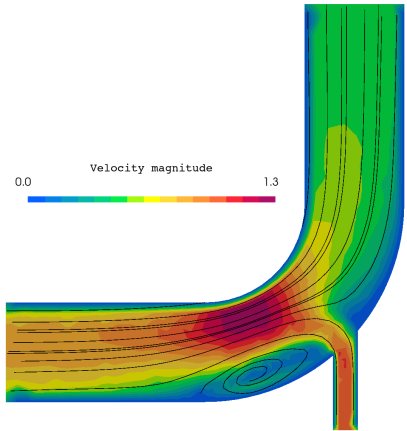
1. We first run the simulator at a few well chosen input values to explore the full input space. The input values are usually chosen using methods from the field of design of experiments
2. We use Gaussian Processes (a type of machine learning technique) to approximate the output values at these inputs values. The fitted Gaussian Process becomes our emulator
3. We check the emulator using graphs and cross validation
4. (Optionally) we calibrate the emulator with real data
5. We deploy the emulator in a manner suitable for its use

# Some good books/resources (with hyperlinks)

- You need to have completed the Rfternoon series of introductory R lectures
- There is a free book: The Design and Analysis of Computer Experiments by Santner, Williams, and Notz
- There are lots of R packages:
  - `GPfit`
  - `emulator`
  - `SAVE`
  - `DiceKriging`
- And a Python package
  - `GP_emulator`

# Example 2: 2D steady-state Navier-Stokes flow



# Example 2 continued

This is a steady-state incompressible Newtonian isothermal laminar Navier-Stokes model

$$\nabla \cdot v = 0; \ \frac{\partial v}{\partial t} + \nabla \cdot (vv) = \nu \nabla^2 v - \frac{1}{\rho} \nabla p + \mathbf{g}$$

... where $\nu$ (velocity vector field) and $p$ (pressure field) are the primary variables/unknowns

# Summary

- We have a slow simulator that we want to avoid running as much as possible
- We run it a few times and use a Gaussian Process to build an emulator of it
- We use the emulator in place of the simulator
- Next: choosing which values to run the simulator at