# Applied Computational Economics
# Lab 0
# Introduction to Numerical Solutions and Coding

The University of Nottingham

**Q1** See the codes.

**Q2** See the codes.

**Q3** This is an interesting concept. Pseudo-random numbers are what a computer can generate for us. They're not really random in the sense that, conditional upon a seed, the numbers that are drawn subsequent will always be the same and appear in the same order. In the question, your $10 \times 1$ vector in step 2 should be the same as the first $10 \times 1$ vector you draw in step 5, (after you've re-set the seed). This is something we'll need to bear in mind later on when simulating artificial datasets.

**Q4** See figure 1. The code main calls the function file myfun.m for this part. You'll need them both to be in the same directory for the call to work properly.

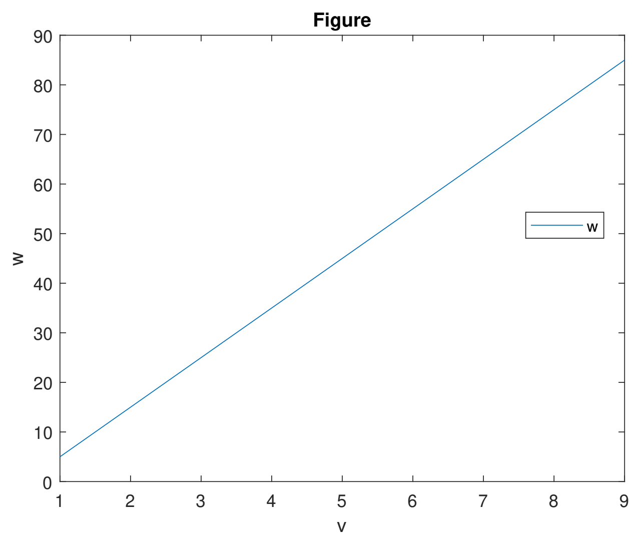


Figure 1: Figure for Q3

**Q5** The analytical solution should just be $y(x) = \frac{x}{2}$, giving $f(x, y(x)) = \frac{x^2}{4}$. Go through the code line-by-line a few times to understand the differences here. The vectorised code is definitely faster, but not to the degree that I was expecting before coding it up. The nested loops take 3.7 seconds versus the vectorised code that takes 2.6 seconds. Figure 2 shows the numerical solutions for $y^*(x)$ and $f(x, y^*(x))$.

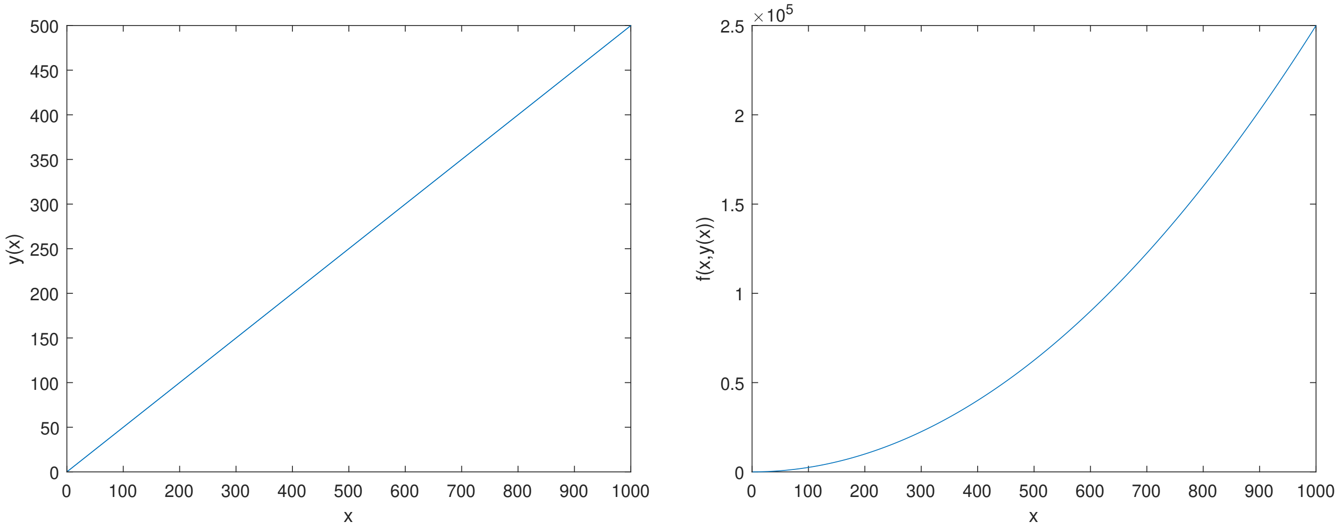


Figure 2: Numerical solutions for Q5

Let's think a little bit more about the vectorised part of the code. Again you should think about the rows of the $F$ matrix as resembling the index value of the array $\vec{x}$ while the columns resemble the index value of the $y$ variable. Let's visualise what's happening symbolically on paper. To simplify the exposition let's reduce the dimensionality to $|\vec{x}| = |\vec{y}| = 3$ and $\vec{x} = (x_1, x_2, x_3)$ with $\vec{y} = (y_1, y_2, y_3)$ (to make this super clear, assume that $\vec{y} \neq \vec{x}$). We want to create a matrix $X$ with each column as the vector $\vec{x}$ as

$$X = \begin{bmatrix} x_1 & x_1 & x_1 \\ x_2 & x_2 & x_2 \\ x_3 & x_3 & x_3 \end{bmatrix}$$

in addition to a matrix $Y$ where each rows is the vector $\vec{y}$ as

$$Y = \begin{bmatrix} y_1 & y_2 & y_3 \\ y_1 & y_2 & y_3 \\ y_1 & y_2 & y_3 \end{bmatrix}.$$

So see that $x$ is held constant along the rows of $X$ while $y$ is held constant down the columns of $Y$. Then we can create the matrix $F$ such that

$$\begin{aligned} F &= X.*Y - Y.^2 \\ &= \begin{bmatrix} x_1y_1 - y_1^2 & x_1y_2 - y_2^2 & x_1y_3 - y_3^2 \\ x_2y_1 - y_1^2 & x_2y_2 - y_2^2 & x_2y_3 - y_3^2 \\ x_3y_1 - y_1^2 & x_3y_2 - y_2^2 & x_3y_3 - y_3^2 \end{bmatrix}. \end{aligned}$$

where again recall that the . (e.g. $X. * Y$) stands for Matlab's element-by-element operator. The first row of $F$ gives all three possible values of the objective function, where each option is for a different value of the $y$ variable with $x_1$ fixed. Thus if we select the column that yields the biggest value, we will have maximised the objective conditional on $x_1$. The maximising column *number* gives the *index* in the $\vec{y}$ array. E.g. if column 3 maximises the objective, then $y_3$ will be $y^*(x_1)$. Then when you want to have the grids for $x$ and $y$ to be the same, as we do in this problem, you can perform the above where $Y = X'$.