School of Mathematics and Statistics
MAST30028 Numerical Methods & Scientific Computing
Week 2

Drag and drop the folder `Week2` from `L: \MAST30028` to `C:\...\MATLAB` and include it in the path, as described in Week 1. Now MATLAB knows how to find the files in `Week2`.

1 Further MATLAB

This relates to material presented in Lecture 3 (Further MATLAB)

Exercise Set 1

a. Define the function $x \mapsto x^2 \cos(\pi x) e^{-x^2}$ using an anonymous function.

   Evaluate it at $x = 4$ and plot it on the interval $[-5, 5]$ using `ezplot`.

b. Many root-finders , trying to solve

$$f(x) = 0$$

   take the form of a fixed point iteration

$$x_{n+1} = g(x_n) \quad n = 0, 1, 2, \dots$$

   until some convergence criterion is satisfied.

   Let x_1, x_2 be the previous and current iterate. Write anonymous functions that take x_1, x_2 and a tolerance *tol* and return `true` if the condition given below is satisfied and `false` otherwise.

   - the absolute difference between x_1 and x_2 is less than the tolerance.
   - the relative difference between x_1 and x_2 is less than the tolerance.

   Hint: these are very simple functions!

c. Write a function that looks for the fixed point of a function by iterating it. The inputs are : a function handle to the function you want to iterate i.e. the function g, a function handle to the criterion you want to apply (from the previous exercise), the initial guess and the tolerance. The iterator should stop if the condition is not satisfied in 10,000 iterations.

d. Test your function from above by feeding it the Newton iteration to find $\sqrt{5}$.

e. Modify your `newtonFuncWhileLoop` so that it finds the square root of the input a from the initial guess x_0, with default value for a of 5 if no value is given. Test your new function.

2 Introduction to NumPy/SciPy

This relates to material presented in Lab 2.

While MATLAB has a large installed base and is popular in numerical analysis and engineering circles, an increasing fraction of scientific computing is being done on open source platforms, including *R* for statistics and machine learning and *Octave*, for scientific computing. However, the fastest growing segment appears to be packages which extend the general purpose object-oriented language *Python* to create overall a MATLAB-like environment, sometimes called the 'SciPy stack'.

Python is a popular introductory programming language with a rich set of data types (lists, tuples, sets and dictionaries) for general-purpose programming. But it doesn't have arrays, especially *numeric arrays* which are the dominant datatype in MATLAB and commonly used in scientific programming.

IPython is a Python interpreter, allowing interactive evaluation of Python commands and programs.

NumPy is a module that extends Python by adding the *ndarray* datatype to Python, which are arrays filled with items of the same numeric type — called *numeric arrays.* With numeric arrays now defined, it supports vectorization of the common mathematical functions over arrays, linear algebra operations, random numbers and more functions such as FFT.

SciPy is a module that builds on NumPy to add functions for root-finding, numerical integration, solving differential equations, interpolation, optimization , special functions and the like.

Matplotlib is a module for graphics. The submodule *PyPlot* is enough to do simple plots of the kind we meet in the next lecture.

Spyder is an Integrated Development Environment IDE that includes a text editor with debugger, an IPython console, and more development tools. Like the MATLAB Desktop environment.

Anaconda is a distribution that allows a simple download of all of the above plus more modules plus a package manager. Other distributions are Enthought Canopy and Python(x,y)[Windows only]. You can download Anaconda on to your own machine: just Google 'anaconda downloads'.

Python comes in 2 branches: Python2 (actually Python 2.7) and Python3 (currently up to Python 3.6) which is NOT backwards-compatible with Python2! Since the distribution in the labs is a couple of years old, it uses Python 2.7. But Python 2 will no longer be supported after 2020 so the general advice is for new learners to use Python3. You can choose which version you want to install when you go to Anaconda downloads.

Exercise Set 2

Run each code section in `lab2.py` that uses NumPy, see what it does by using the Variable Explorer, Object Inspector etc. then write MATLAB code that does the same thing, if possible.

3 Simple plotting

Exercise Set 3

This relates to material presented in Lecture 4 (Plotting in MATLAB).

Plotting in MATLAB

Write MATLAB commands to create the following kinds of plots:

a. Plot $\sin\theta$ versus θ for 60 points in the interval $0 \le \theta \le 2\pi$. Connect the points with a dashed line *and* label the points with open circles.

b. Create a plot that will reveal whether the data $\{y_n\}$ decay geometrically with n e.g. $y_n = 4.3(0.5)^n$

c. Create a plot that will reveal whether the data $\{y_n\}$ decay algebraically with n e.g. $y_n = 7.2n^{-2}$

d. Plot $y = x^2; y = \sin(x^2)$, and $y = \sin(x)$ on the same graph. Label them with arrows pointing to the respective curve (for example, have an arrow from the label '$y = x^2$' to the curve $y = x^2$). I suggest you do this interactively with Plot Tools.

e. Use a parametric plot to plot the ellipse

$$\frac{x^2}{2^2} + \frac{y^2}{3^2} = 1$$

Plotting with Matplotlib

By following the examples in `lec4.py`, write Python commands to create the same plots as the Exercises above.

It will be sufficient to use the PyPlot submodule of Matplotlib.