Numerical Methods 35006
Computer Lab 5: Interpolation and extrapolation

1. A useful piece of sub-code in interpolation is one that computes a polynomial for a particular value of x, given a set of polynomial coefficients. That is, it computes

$$f(x) = a_0 + a_1 x + a_2 x^2 + ... + a_{n-1} x^{n-1} \ .$$

Write a python script `polyx` that is called as follows:

```
f = polyx(a,x)
```

and which returns the value of $f(x)$ given a nump array of coefficients `a`, which could be of any length. Test your script by plotting the function

$$f(x) = 1 - 2x + x^2$$

over the range $x \in [-3, 3]$.

2. This next question guides you through setting up the Vandermonde matrix (and introduces a few matrix procedures that will be useful later on).

a) First set up a series of point pairs $(-1, 2), (0, -1), (1, 1), (2, 0)$ by defining the `xp` and `yp` arrays:

```
xp = np.array([[-1],[0],[1],[2]])
yp = np.array([[2],[-1],[1],[0]])
```

b) Matrices in python are 2D numpy arrays, which can be concatenated using the `hstack` command. For example, to build the matrix

$$A = \begin{bmatrix} -1 & -1 \\ 0 & 0 \\ 1 & 1 \\ 2 & 2 \end{bmatrix} \qquad (1)$$

you could use the command

```
A = np.hstack([xp,xp])
```

Use a loop, together with `hstack` and the values of `xp` above, to construct the Vandermonde matrix V discussed in lectures.

c) Matrix inversion works using the `linalg` module within `numpy`. The matrix inverse V^{-1} con be constructed using the command

```
VI = np.linalg.inv(V)
```

and two matrices A and B can be multiplied using

```
C = matmul(A,B)
```

Find the set of polynomial coefficients `a` by multiplying V^{-1} by the column vector `yp`.

d) Finally, check the interpolation by using `polyx` to plot the interpolating polynomial on the range $x \in [-1, 2]$.

3. Using your code from Q2, create a function `vandint(x,xp,yp)` which returns the values of an interpolating polynomial using Vandemonde interpolation, and test that it works on the values from Q2. Create a new python module `myinterp.py` and save both your `polyx` and `vandint` procedures there.

4. Create a function `lagx(j,x,xp)` that, given a set of points `xp` as in Q2, returns the Lagrange polynomial for the j^{th} node:

$$\ell_j(x) = \frac{\prod_{i \neq j}(x - x_i)}{\prod_{i \neq j}(x_j - x_i)} \ .$$

Plot these functions over the range $x \in [-1, 2]$.

5. Implement the sum

$$L(x) = \sum_j^n \ell_j(x) y_j$$

to form a Lagrange interpolating polynomial for `xp` and `yp`. Plot you solution.

6. Create a function `lagint(x,xp,yp)` which returns the values of an interpolating polynomial using Lagrange interpolation, and test that it works on the values from Q2. Save this to your `myinterp` file.