1. Use Matlab (or some other program) to calculate the polynomial which interpolates the function
\[ f(x) = \frac{1}{1 + 25x^2} \]
at equally spaced points between -1 and 1, in steps of 0.25. You can use a Matlab built-in to find the polynomial.
Plot the original function (in the default color blue) and the interpolating polynomial (in red), and put circles around the interpolation points. Make sure you use enough points so that you get smooth-looking curves. (10)

2. (a) Given a point \( x_0 \) and stepsize \( h \), use Lagrange polynomials to find the quadratic polynomial which interpolates a function \( f(x) \) at the points \( x_0 - 2h, x_0, \) and \( x_0 + h \).
   (b) Integrate the polynomial on \([x_0 - 2h, x_0 + h]\) to produce an integration formula. Use the formula to estimate \( \int_0^3 e^{-x} \, dx \). (10)

3. This is a modified form of problem 2.3, page 31, in the Iserles book.
   Recall that the \( k \)-step Adams-Bashforth and Adams-Moulton methods are based on
   \[ y(t_{n+k}) = y(t_{n+k-1}) + \int_{t_{n+k-1}}^{t_{n+k}} f(\tau, y(\tau)) \, d\tau. \]
You interpolate \( f \) at \( t_n, t_{n+1}, \ldots, t_{n+k-1} \) (AB) or \( t_n, t_{n+1}, \ldots, t_{n+k} \) (AM) by a polynomial, and integrate that.
You can also start integrating at \( t_{n+k-2} \) instead of \( t_{n+k-1} \). For \( k = 2 \), Iserles calls these the \textit{Nystrom} and \textit{Milne} methods. That is, start with
   \[ y(t_{n+2}) = y(t_n) + \int_{t_n}^{t_{n+2}} f(\tau, y(\tau)) \, d\tau, \]
and interpolate at \( t_n, t_{n+1} \) (explicit) or \( t_n, t_{n+1}, t_{n+2} \) (implicit).
   Find the leading error term, and the order of the method, for both cases. (10)

4. The IVP
   \[ y' = -y + 2e^{-t}\cos(2t) \]
   \[ y(0) = 0 \]
has the true solution \( y(t) = e^{-t}\sin(2t) \).
Solve it numerically, using Euler’s method with 20, 40, 80, 160 steps from 0 to 1, and compute the global error at the endpoint for each stepsize. Verify that the error goes down approximately linearly. (10)
5. Solve

\[ y''(t) = \begin{pmatrix} 0 \\ -10 \end{pmatrix} - 0.1 \, y'(t), \]
\[ y(0) = \begin{pmatrix} 0 \\ 0 \end{pmatrix}, \]
\[ y'(0) = \begin{pmatrix} 10 \\ 10 \end{pmatrix} \]

by using Euler’s method with stepsize \( h = 0.1 \), until \( y_2 \) becomes 0 or negative.

**Hints:** \( y(t) \) is a vector with 2 components, representing the position of an object at time \( t \). The DE comes from Newton’s law (force = mass · acceleration). The object is launched at time \( t = 0 \) from the origin, with an initial velocity \((10, 10)^T \) (speed \( 10\sqrt{2} \) at a \( 45^\circ \) angle).

The forces acting on the object are gravity (first term on the right), and friction. I am assuming that friction is proportional to the speed \( \| y' \| \), and opposite to the direction of motion \( y'/\| y' \| \) (second term).

What you need to do first is to convert this to a first order system with 4 components, and then use Euler’s method in vector form.

Plot \( y_1 \) and \( y_2 \) as functions of \( t \) (horizontal and vertical distance as functions of time), and also plot \( y_2 \) versus \( y_1 \). That will show the path of the object. It should look a bit like a parabola, but compressed on the right because of friction.

**2 points extra credit:** Make friction proportional to some power of the speed, such as 1.5 or 2. As far as I know, aerospace engineers use different powers in different ranges (slow speed, high speed, supersonic).