Goal:
To see how good and bad various interpolation methods can be, using Matlab’s interpolation routines on data generated from Runge’s function:

\[ f(x) = \frac{1}{1+x^2}. \]

In Matlab, do the following:

1. Problem setup:

Generate \( N + 1 = 11 \) equally-spaced nodes \( x_i \) in the interval \([-5, 5]\)

\[
N = 10; \\
x = linspace(-5,5,N+1); \\% to see values, omit the ;
\]

and then evaluate \( f(x) \) at these nodes

\[
f = inline('1./(1+x.*x)','x'); \\
y = f(x);
\]

The \( N + 1 \) points \( (x_i, y_i) \) are the data points to be interpolated by various methods. Plot them to see where they are:

\[
plot(x,y,'o') \\
title('N+1 = 11 equally-spaced data points')
\]

Also generate lots of points \( t_i \) at which to evaluate \( f \), and the interpolants, for plotting:

\[
t = [-5:.1:5];
\]

Evaluate \( f \) at these \( t_i \)’s and plot \( f(t) \) in a new figure window:

\[
figure; \\
plot(t,f(t),'-') \\
title('Runge function')
\]
2. Nth degree interpolating polynomial:

Use Matlab’s `polyfit` to construct (the coefficients of) the Nth degree interpolating polynomial

- Step a:
  \[ PN = \text{polyfit}(x,y,N); \]

- Step b: Now this can be evaluated anywhere in the interval \([-5,5]\), e.g., at the \(t_i\)’s
  \[ v = \text{polyval}(PN,t); \]

- Step c: Find the (inf-norm) error \( \|f(t) - PN(t)\|_\infty = \max|f(t) - PN(t)| \)
  \[ \text{err} = \text{norm}(f(t)-v,\text{inf}) \]

- Step d: Plot both \(f(t)\) and \(PN(t)\) on the same plot as the data points
  ```matlab
  figure;
  plot(x,y,’o’,t,f(t),’-’,t,v,’--’)
  title(sprintf(‘f(t) and P_{10}(t), err=%g’,err))
  ```

3. Interpolation at Chebychev nodes: Instead of using equally spaced interpolation nodes, we use a different set of nodes. The \(x_i\)s are roots of the Chebychev polynomial. Construct the points \((x_i, y_i)\) as follows. We will call \(x_i = \text{xcheb}(i)\) and \(y_i = \text{ycheb}(i)\).

Generate \(N + 1 = 11\) Chebychev nodes

```
K = N+1;
a=-5;
b=5;
xcheb=zeros(1,K);
for i=1:K
  xcheb(i)=(a+b)/2 + (b-a)/2 * cos( (i-.5)*pi/K );
end
ycheb = f(xcheb);
```

Follow the steps a-d in part 2. to produce the Nth degree interpolating polynomial \(PN_{\text{cheb}}\) based on the Chebychev nodes, its values \(v_{\text{cheb}}\) at the \(t_i\)’s, and the error \(\|f(t) - PN_{\text{cheb}}(t)\|_\infty\), and plot both \(f(t)\) and \(PN_{\text{cheb}}(t)\) on the same plot as the Chebychev data. Compare the error and the plot with those from 2. Comment on why one works better than the other.
4. Piecewise linear interpolation:

Use Matlab’s `interp1` to construct the piecewise linear interpolant evaluated at the $t_i$’s

$$v_{\text{lin}} = \text{interp1}(x,y,t,’\text{linear’});$$

Repeat the steps c and d of part 2. to compute the error and plot. Compare error and plot with those from the previous examples.

5. Piecewise cubic interpolation:

Use Matlab’s `interp1` to construct the piecewise cubic interpolant evaluated at the $t_i$’s

$$v_{\text{cub}} = \text{interp1}(x,y,t,’\text{cubic’});$$

Repeat the steps c and d of part 2. Compare errors and plots.

6. Cubic spline interpolation:

Use Matlab’s `interp1` to construct the cubic spline interpolant evaluated at the $t_i$’s

$$v_{\text{spl}} = \text{interp1}(x,y,t,’\text{spline’});$$

Repeat the steps c and d of part 2. Compare errors and plots.

7. To see that the error gets worse for equally-spaced nodes but not for Chebychev nodes (for this $f(x)$ at least), repeat parts 1., 2. and 3. with $N = 20$.

Turn in a hard copy of the results of the above commands, including plots, and your answers to all of the questions for each problem. Include a printout of the script which runs the above commands.