Many problems in this homework make use of a few MATLAB/Octave .m files that are provided on the website. In order to use them, make sure that the files are in the same directory that you are running MATLAB/Octave from (to see which directory this is, type pwd in MATLAB/Octave).

1. **Compute the determinant of a $4 \times 4$ Vandermonde matrix. Bonus: show that the general formula for the determinant of a Vandermonde matrix is correct.**

   Here is the general calculation, although you only need to hand in the $4 \times 4$ case. The inductive step is to show that the formula for $n$ follows from the formula for $n - 1$. Let $d(n; x_1, x_2, \ldots, x_n)$ be the determinant of the $n \times n$ Vandermonde matrix with variables $x_1, x_2, \ldots, x_n$. We begin with the same steps as the $3 \times 3$ example done in lectures. First we subtract $x_n$ times the second column from the first column, then $x_n$ times the third column from the second column, and so on. This doesn’t change the determinant

   $$
   d(n; x_1, x_2, \ldots, x_n) = \det \begin{pmatrix}
   x_1^{n-1} & x_1^{n-2} & \cdots & x_1^2 & x_1 & 1 \\
   x_2^{n-1} & x_2^{n-2} & \cdots & x_2^2 & x_2 & 1 \\
   x_3^{n-1} & x_3^{n-2} & \cdots & x_3^2 & x_3 & 1 \\
   \vdots & \vdots & \ddots & \vdots & \vdots & \vdots \\
   x_n^{n-1} & x_n^{n-2} & \cdots & x_n^2 & x_n & 1
\end{pmatrix}
\text{det}
\begin{pmatrix}
   x_1^{n-1} - x_1^n - x_1^{n-2} & x_1^{n-2} - x_1^{n-3} & \cdots & x_1^2 - x_1x_n & x_1 - x_n & 1 \\
   x_2^{n-1} - x_2^n - x_2^{n-2} & x_2^{n-2} - x_2^{n-3} & \cdots & x_2^2 - x_2x_n & x_2 - x_n & 1 \\
   x_3^{n-1} - x_3^n - x_3^{n-2} & x_3^{n-2} - x_3^{n-3} & \cdots & x_3^2 - x_3x_n & x_3 - x_n & 1 \\
   \vdots & \vdots & \ddots & \vdots & \vdots & \vdots \\
   x_n^{n-1} - x_n^n - x_n^{n-2} & x_n^{n-2} - x_n^{n-3} & \cdots & x_n^2 - x_nx_n & x_n - x_n & 1
\end{pmatrix}
= \det
\begin{pmatrix}
   (x_1 - x_n)x_1^{n-2} & (x_1 - x_n)x_1^{n-3} & \cdots & (x_1 - x_n)x_1 & x_1 - x_n & 1 \\
   (x_2 - x_n)x_2^{n-2} & (x_2 - x_n)x_2^{n-3} & \cdots & (x_2 - x_n)x_2 & x_2 - x_n & 1 \\
   (x_3 - x_n)x_3^{n-2} & (x_3 - x_n)x_3^{n-3} & \cdots & (x_3 - x_n)x_3 & x_3 - x_n & 1 \\
   \vdots & \vdots & \ddots & \vdots & \vdots & \vdots \\
   0 & 0 & \cdots & 0 & 0 & 1
\end{pmatrix}
$$

Now expand along the bottom row and remove the common factors.

$$
d(n; x_1, x_2, \ldots, x_n) = (x_1 - x_n)(x_2 - x_n) \cdots (x_{n-1} - x_n)d(n - 1, x_1, x_2, \ldots, x_{n-1})
= (-1)^{n-1}(x_n - x_1)(x_n - x_2) \cdots (x_n - x_{n-1})d(n - 1, x_1, x_2, \ldots, x_{n-1})
$$

We may assume that the result for $n - 1$ is known. That is

$$
d(n - 1, x_1, x_2, \ldots, x_{n-1}) = (-1)^{(n-1)(n-2)/2} \prod_{n \geq i > j \geq 2} (x_i - x_j)
$$
Thus we conclude
\[ d(n; x_1, x_2, \ldots, x_n) = (-1)^{n-1}(x_n - x_1)(x_n - x_2) \cdots (x_n - x_{n-1})(-1)^{(n-1)(n-2)/2} \prod_{n \geq i \neq j \geq 2} (x_i - x_j) \]
\[ = (-1)^{(n-1)+(n-1)(n-2)/2} \prod_{n \geq i \neq j \geq 1} (x_i - x_j) \]
and the result follows from the calculation
\[ (n-1) + (n-1)(n-2)/2 = (2n - 2 + (n-1)(n-2))/2 = (2n - 2 + n^2 - 3n + 2)/2 = n(n-1)/2. \]

2. Let \( V_n \) be the Vandermonde matrix for \( n \) equally spaced points between 0 and 1. Do you think the condition number of \( V_n \) is increasing exponentially in \( n \)? To make an informed guess, use MATLAB/Octave to make a plot of \( \log(\text{cond}(V_n)) \) against \( n \). You will need to use relatively small values of \( n \) (say \( n < 20 \) or so) to get a reasonable looking plot. What do you think is happening when you use larger values of \( n \)?

The point behind this plot is that if \( \text{cond}(V_n) \) is exponentially increasing, that is \( \text{cond}(V_n) \sim C e^{\alpha n} \) then taking logs we get \( \log(\text{cond}(V_n)) \sim \log(C) + \alpha n \), that is, a linear function of \( n \) with slope \( \alpha \). So if we plot \( \log(\text{cond}(V_n)) \) and it looks linear, then this is evidence of exponential growth for \( \text{cond}(V_n) \).

The quantity \( \ln(\text{cond}(V_n)) \) can be computed as \( \log(\text{cond(vander(linspace(1,0,n))))} \). If you compute this for a collection of \( n \)'s and plot the resulting points against \( n \) you get a graph that looks like .... well it depends on what \( n \)'s you choose. My first try (for relatively small values of \( n \)) looks nicely linear.

However if \( n \) gets large the graph starts to look rough:
This is almost certainly due to truncation error when the computer is dealing with extremely large numbers. So I would still count this as evidence that the condition number is increasing exponentially, based on the first part of the graph (although maybe not completely convincing!).

3. Use MATLAB/Octave to plot the Lagrange interpolating function through the points $(1,2.3), (2.5), (2.4,9), (2.5,5), (3,0)$ and $(5,-1)$. Plot the cubic spline interpolating function on the same figure (and through the same points). You may use the Matlab file plotspline.m.

The MATLAB/Octave commands to plot this are:

```matlab
> X = [1 2 2.4 2.5 3 5]
X =
  1.0000  2.0000  2.4000  2.5000  3.0000  5.0000

> Y = [2.3 5 9 5 0 -1]
Y =
  2.30000  5.00000  9.00000  5.00000  0.00000  -1.00000

> V = vander(X);
> a = V\Y';
> XL = linspace(1,5,100);
> YL = polyval(a,XL);
> plot(X,Y,'bo')
> hold on
> plot(XL,YL,'b-')
> plotspline(X,Y);
> axis([0 6 -100 600])
> hold off
```

The resultant plots looks like this (the Lagrange interpolation is the dotted line):
The purpose of this problem was to show that even with only a few points, if those points are badly distributed the Lagrange interpolating polynomial can give a terrible fit.

4. Derive the matrix equation to solve in order to find the cubic spline passing through the three points \((0, 3), (0.5, 1)\) and \((1, 6)\). Plot the resulting spline (you may use the file plotspline.m). Take the cubic spline model in which the cubic polynomials in the two intervals are:

\[
\begin{align*}
p_1(x) &= a_1 x^3 + b_1 x^2 + c_1 x + d_1 & 0 \leq x \leq 0.5 \\
p_2(x) &= a_2 x^3 + b_2 x^2 + c_2 x + d_2 & 0.5 \leq x \leq 1.
\end{align*}
\]

Write your matrix equation in the form

\[
Ax = b
\]

where \(x^T = [a_1, b_1, c_1, d_1, a_2, b_2, c_2, d_2]\).

Take the cubic polynomials in the two intervals to be

\[
\begin{align*}
p_1(x) &= a_1 x^3 + b_1 x^2 + c_1 x + 1 & 0 \leq x \leq 1/2 \\
p_2(x) &= a_2(x - 1/2)^3 + b_2(x - 1/2)^2 + c_2(x - 1/2) + 2 & 1/2 \leq x \leq 1
\end{align*}
\]

At \(x = x_2 = 1/2\) we impose \(p_1 = p_2 = 2\) (the latter of which is already satisfied), \(p_1' = p_2'\) and \(p_1'' = p_2''\).

This gives the three equations

\[
\begin{align*}
a_1(1/2)^3 + b_1(1/2)^2 + c_1(1/2) + 1 &= 2, \\
3a_1(1/2)^2 + 2b_1(1/2) + c_1 &= c_2, \\
6a_1(1/2) + 2b_1 &= 2b_2.
\end{align*}
\]

At \(x = x_1 = 0\) impose \(p_1 = 1\) (already satisfied), \(p_1'' = 0\). This gives

\[
2b_1 = 0. 
\]

At \(x = x_3 = 1\) impose \(p_2 = 4\), \(p_2'' = 0\). This gives

\[
\begin{align*}
a_2(1/2)^3 + b_2(1/2)^2 + c_2(1/2) + 2 &= 4, \\
6a_2(1/2) + 2b_2 &= 0.
\end{align*}
\]
Multiplying equation (2) by 1/2 and equations (3), (5) and (6) by (1/2)^2. Writing this in matrix form find

\[
\begin{bmatrix}
1 & 1 & 1 & 0 & 0 & 0 \\
3 & 2 & 1 & 0 & 0 & -1 \\
6 & 2 & 0 & 0 & -2 & 0 \\
0 & 0 & 0 & 1 & 1 & 1 \\
0 & 2 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 \\
\end{bmatrix}
\begin{bmatrix}
a_1(1/2)^3 \\
b_1(1/2)^2 \\
c_1(1/2) \\
a_2(1/2)^3 \\
b_2(1/2)^2 \\
c_2(1/2) \\
\end{bmatrix}
= 
\begin{bmatrix}
1 \\
0 \\
0 \\
2 \\
0 \\
0 \\
\end{bmatrix}
\]

We use `plotspline.m` to create the plot

```matlab
> X = [0 1/2 1];
> Y = [1 2 4];
> plotspline(X,Y)
> axis([-0.1 1.1 0.9 4.1])
```

producing the plot

5. **What happens to the condition number of the matrix S used in cubic spline interpolation as the size n becomes large (you may use the file `splinemat.m`)?**

In contrast to the Vandermonde matrix, the condition number of the spline matrix approaches a constant. Here is a plot

![Plot](image_url)
6. A parabolic runout spline is the interpolating function you get by changing the condition 
\( f''(x_1) = f''(x_n) = 0 \) to the condition that \( p_1(x) \) and \( p_{n-1}(x) \) should be quadratic polynomials (that is, \( a_1 = a_{n-1} = 0 \)). Modify the file splinemat.m so that it computes the matrix relevant to this modified problem. Call the modified file splinematpr.m. (Hand in a a print-out of the modified file and an explanation of your changes.) Use your new file to graph the parabolic runout spline for the points \((1,1), (2,1), (3,2), (4,4)\) and \((5,3)\). (The easiest way to do this is to change splinemat to splinematpr inside the file plotspline.m and call the modified file plotsplinepr.m. Use this new file to plot the modified spline.) Hand in a plot of both the parabolic runout spline and the cubic spline on the same graph.

In splinemat.m change the lines

\[
\begin{align*}
T &= \begin{bmatrix} 0 & 0 & 0; 0 & 2 & 0; 0 & 0 & 0 \end{bmatrix}; \\
V &= \begin{bmatrix} 1 & 1 & 1; 0 & 0 & 0; 6 & 2 & 0 \end{bmatrix};
\end{align*}
\]

to

\[
\begin{align*}
T &= \begin{bmatrix} 0 & 0 & 0; 1 & 0 & 0; 0 & 0 & 0 \end{bmatrix}; \\
V &= \begin{bmatrix} 1 & 1 & 1; 0 & 0 & 0; 1 & 0 & 0 \end{bmatrix};
\end{align*}
\]

Here are the two types of spline on the same graph.
7. Consider the problem of interpolating four points \((x_1, y_1), (x_2, y_2), (x_3, y_3)\) and \((x_4, y_4)\) with a function \(f(x)\) that is given by a quadratic polynomial in each interval \(x_i, x_{i+1}\), (i.e., \(p_i(x) = a_i(x - x_i)^2 + b_i(x - x_i) + c_i\)) and whose first derivative \(f'(x)\) is continuous across the points \(x_i\). Write down the system of equations for this problem. Is there a unique solution to this problem?

The equations are

\[
\begin{align*}
p_1(x_1) &= y_1, & p_2(x_2) &= y_2, & p_3(x_3) &= y_3 \\
p_1(x_2) &= y_2, & p_2(x_3) &= y_3, & p_3(x_4) &= y_4 \\
p_1'(x_2) &= p_2'(x_2) \\
p_2'(x_3) &= p_3'(x_3)
\end{align*}
\]

These can be written

\[
\begin{align*}
c_1 &= y_1 \\
c_2 &= y_2 \\
c_3 &= y_3 \\
a_1(x_2 - x_1)^2 + b_1(x_2 - x_1) + c_1 &= y_2 \\
a_2(x_3 - x_2)^2 + b_2(x_3 - x_2) + c_2 &= y_3 \\
a_3(x_4 - x_3)^2 + b_3(x_4 - x_3) + c_3 &= y_4 \\
2a_1(x_2 - x_1) + b_1 &= b_2 \\
2a_2(x_3 - x_2) + b_2 &= b_3
\end{align*}
\]

This is system of 8 equations in 9 unknowns and therefore does not have a unique solution.

8. Check if the mathematical modeling for cubic splines gave an answer matching the physical world. In particular, using the Matlab file plotspline.m, plot a spline passing through the points \((0, 0), (5, 1), (10, -1), (15, 2)\). Then find a tube, or some other stiff but flexible rod-type-thingy, and overlay it on top of your graph, clamped in place at the above points. You are also welcome to play with the choice of interpolation points. How well did the modeling seem to work? (If you want to make the professor happy, come by office hours and show him your work!)