Exercise 7.9: Compute the $1-norm$ and $\infty-norm$ of the following matrix,

$$A = \begin{pmatrix} 5 & 6 \\ 7 & 8 \end{pmatrix}$$

Also compute the condition number $k(A) = \|A\| \cdot \|A^{-1}\|$ in the $1-norm$ and $\infty-norm$.

Solution:
From class we proved that the $1-norm$ of the matrix A is simply the maximum $1-norm$ of the column vectors, so we get that $\|A\|_1 = 14$. We also showed in class that the $\infty-norm$ of a matrix A is the maximum $1-norm$ of the row vectors, so we get $\|A\|_\infty = 15$. Computing the inverse of A,

$$A^{-1} = \frac{1}{40-42}\begin{pmatrix} 8 & -6 \\ -7 & 5 \end{pmatrix} = \begin{pmatrix} -4 & 3 \\ 3.5 & -2.5 \end{pmatrix}.$$

Computing the norms for A^{-1} we get that $\|A^{-1}\|_1 = 7.5$ and $\|A^{-1}\|_\infty = 7$. Therefore we get the following condition numbers

$$k(A)_1 = 7.5 * 14 = 105,$$

$$k(A)_\infty = 7 * 15 = 105.$$

Exercise 7.10: Show that for all n-vectors v,

1. $\|v\|_\infty \leq \|v\|_2 \leq \sqrt{n}\|v\|_\infty$

   Solution:
   Note that for every non empty v there exists some v_j such that,

   $$|v_j| \geq |v_n|,$$

   for all v_n; there must exists a maximum vector element. Now the following expression follows expression,

   $$v_j^2 \leq \sum_{i=1}^{n} v_i^2 \leq nv_j^2$$

   Where v_j^2 must be the smallest form the sum of the squares can take on, and nv_j^2 begin the largest. Taking the positive square root of the expression,

   $$|v_j| \leq (\sum_{i=1}^{n} v_i^2)^{\frac{1}{2}} \leq n^{\frac{1}{2}}|v_j|,$$

Which is equivalent to the following,

$$\|v\|_\infty \leq \|v\|_2 \leq \sqrt{n}\|v\|_\infty.$$

2. $\|v\|_2 \leq \sqrt{n}\|v\|_1$
   Start with an induction/triangle inequality argument to show that for any number of n, real numbers x_i

$$x_1^2 + x_2^2 + \cdots + x_n^2 \leq (|x_1| + |x_2| + \cdots + |x_n|)^2.$$

   Then it follows by the definition of the 1 and 2 norms.

3. $\|v\|_1 \leq n\|v\|_\infty$
   Substituting the definitions we get that,

$$\|v\|_1 \leq n\|v\|_\infty,$$
$$(|x_1| + |x_2| + \cdots + |x_n|) \leq n \max\{|x_n|\}.$$

   Clearly this is true.

Exercise 7.14: Find the polynomial of degree 10 which best fits the function $b(t) = cos(4t)$ at 50 equally spaced points t between 0 and 1. Set up the matrix A and right-hand side vector b, and determine the polynomial coefficients in two different ways,

1. By using the Matlab command $x = Ab$.

   Console:

```
>> x = linspace(0,1,50);
>> f = @(x) cos(4*x)

>> b = f(x);
>> A = [];
>> for i = 0:10
        y = x.^i;
        A = [A y'];
    end
```

```
>> c = A\b'

c =

    1.000000038120647
   -0.000012132618937
   -7.999528596502270
   -0.007137816703811
   10.722601995909741
   -0.257549358411933
   -4.946643811578490
   -1.373303866928413
    3.243340066641667
   -1.145391639738877
    0.109981465124875
```

From here we have vector c which lists the coefficients of our polynomial in ascending order of degree, i.e we would use $polyval(c')$.

2. By solving the normal equations $A^TAx = A^Tb$.

Console:

```
>> cnormal = (A'*A)\(A'*b')

cnormal =

    1.000000032521446
   -0.000010998683460
   -7.999562825884168
   -0.006728614560766
   10.720073608297179
   -0.248463295429089
   -4.966699583559977
   -1.345726488854232
    3.220313782953815
   -1.134708087892108
    0.107868811092834
```

We have demonstrated in class that solving for the coefficient using the normal equations leads to more error since the condition number of A^TA is the square of R where

$A = QR$. Recall that solving the system through QR decomposition with the backslash operator in Matlab means matlab is solving the following,

$$Rc = Q^T b.$$

Similarly when we solve the normal equations with the backslash operator in Matlab we solve the follow system,

$$R_n c = Q_n^T b.$$

Where $A^T A = Q_n R_n$ then comparing the condition numbers of R_n and R we see that solving the normal equations leads to a more ill conditioned system and thus less precision.

Console:

```
>> [Q, R] = qr(A);
>> [Qn, Rn] = qr(A'*A);
>> cond(R)

ans =

    2.035662963861317e+07

>> cond(Rn)

ans =

    4.138669220598705e+14
```

Supplemental 1: Let

$$A = \begin{pmatrix} 1 & -1 & 0 & \alpha-\beta & \beta \\ 0 & 1 & -1 & 0 & 0 \\ 0 & 0 & 1 & -1 & 0 \\ 0 & 0 & 0 & 1 & -1 \\ 0 & 0 & 0 & 0 & 1 \end{pmatrix}; \qquad \mathbf{b} = \begin{pmatrix} \alpha \\ 0 \\ 0 \\ 0 \\ 1 \end{pmatrix}$$

a) Show that that for any choice of numbers α and β, the solution of $A\mathbf{x} = \mathbf{b} = (1, 1, 1, 1, 1)^T$.

Solution:
Performing a simple upper triangular solve we can see that the solution will always be

$b = (1, 1, 1, 1, 1)^T$,

$$
\begin{aligned}
x_5 &= 1 \\
x_4 &= 1(x_5) = 1 \\
x_3 &= 1(x_4) = 1 \\
x_2 &= 1(x_3) = 1 \\
x_1 &= \alpha - \beta(x_5) - (\alpha - \beta)(x_4) + 1(x_2) = 1
\end{aligned}
$$

b) This is an upper triangular matrix! For $\alpha = 0.1$ and $\beta = 10^1, 10^2, \ldots, 10^{12}$ solve $A\mathbf{x} = \mathbf{b}$ using your `usolve` code. Present a table of $\|\mathbf{x} - \hat{\mathbf{x}}\|_\infty$

Solution:
Console:

```
>> b = [.1  0  0  0  1];

>> X = [] % Matrix to save x values; column index is beta factor

    for i = 1:12
        A = MatBuild(.1, 1*10^i);
        x = usolve_hw9(A,b');
        X = [X x];
    end


>> D = [] % Matrix to save the error in each u_solve

    for i = 1:12
        x = X(:,i) - [1 1 1 1 1]';
        D = [D x];
    end


>> N = [] % Vector to save the Norm of the error

    for i = 1:12
        N(i) = norm(D(:,i),inf);
    end

>> N'

ans =

   1.0e-04 *
```

```
0.000000000004441   Beta = 10^1
0.000000000057732
0.000000000227596
0.000000003638201
0.000000058207883
0.000000232831532
0.000003725291187
0.000059604645664
0.000238418579324
0.003814697265847
0.061035156250222
0.244140625000888   Beta = 10^12
```

c) Present a table of the ∞ norm condition numbers of the matrices A from the previous problem.

Solution:
Console:

```
>> N = []

for i = 1:12
    A = MatBuild(.1, 1*10^i);
    x = cond(A, inf);
    N = [N x];
end

>> format long
N'
ans =

   1.0e+24 *

   0.000000000000000     Beta = 10^1
   0.000000000000000
   0.000000000000000
   0.000000000000000
   0.000000000000020
   0.000000000002000
   0.000000000200000
   0.000000020000001
```

```
0.000002000000012
0.000200000000115
0.020000000001150
2.000000000011500      Beta = 10^12
```

d) Discuss the relationship between parts (b) and (c).

Solution:
As the matrix A becomes more ill-conditioned, since the condition numbers increase as *beta* increases we see a rise in the amount of error propagated when we solve $Ax = b$ through QR factorization.

Supplemental 2: Suppose you have data points $(1, y_1), \ldots, (n, y_n)$ and that the points $(k, \log(y_k))$ all lie on a line with positive slope. Show that there are constants $C > 0$ and $\alpha > 1$ such that

$$y_k = C\alpha^k$$

Solution:
Assuming the *log* is natural then if the points $(k, \ln(y_k))$ all lie on a line with positive slope then there exists a line such that,

$$ln(y_k) = Mk + b.$$

Where $M > 0$ and $b \in \mathbb{R}$. Through some algebra we get that,

$$\begin{aligned} ln(y_k) &= Mk + b, \\ e^{ln(y_k)} &= e^{Mk+b}, \\ y_k &= e^b e^{Mk}, \\ y_k &= e^b (e^M)^k. \end{aligned}$$

Note that $e^b > 0$ and $e^M > 1$ since $M > 0$. Thus we let $C = e^b$ and $\alpha = e^M$ and we get that,

$$y_k = C\alpha^k.$$

Supplemental 3: We will shortly be seeing the Vandermonde matrices, which show up when doing polynomial interpolation. So, they appear naturally in the real world, and the point of this exercise is to characterize just how awfully their condition number grows as the size of the matrix grows.

Given a vector $\mathbf{x} = (x_0, x_1, \ldots, x_n)^T$, the $(n+1) \times (n+1)$ Vandermode matrix associated with $\mathbf{x}$ is defined on page 181 in your text. You can create one in matlab with the command `vander(x)`.

1. For $n = 1, 2, \ldots, 20$, let $\mathbf{x} = (0, 1/n, 2/n, \ldots, 1)$, and let κ_n be the 2-norm condition number of the Vandermonde matrix associated with $\mathbf{x}$. Make a plot of $\log(\kappa_n)$ versus n.

   Solution:
   The following funciton takes n as a parameter and return a vector of the 2-norm condition number of the Vandermonde matrix associated with $\mathbf{x}$.

   Console:

```
function [N] = vandercond(n)

N = [];
    for j = 1:n
        x = [0];
            for i = 1:j
            x = [x i/j];
            end
        V = vander(x');
        N = [N, cond(V,2)];
    end
end
```

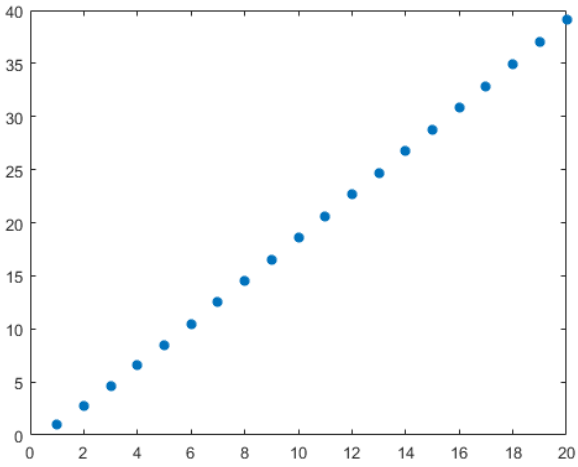
   Then we find the log of the vector and plot the points over the corresponding value of n, **Console:**

```
>> N = vandercon(20);
>> logN = log(N');
>> n = [1:20];
>> plot(n,logN','*','linewidth',2)
```

2. If everything has gone well, your plot will look like a straight line! Use a least squares method to find m and b such that

$$\log(\kappa_n) \approx mn + b$$

Figure 1: Plot of $log(\kappa_n)$ over n



Then plot your line on the same graph as in part (b).

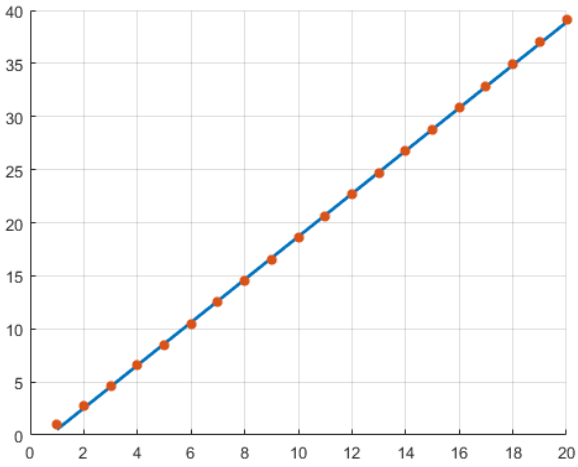
Console:

```
n = [1:20];
N = vandercon(20);
LogN = log(N');

>> V = [n',ones(size(n',1),1)];

>> c = V\logN

c =

    2.019377877332083    %Solved for M
   -1.501448611841827    %Solved for b

>> hold on
>> plot(x, polyval(c,x),'linewidth',2)
>> plot(n,logN','*','linewidth',2)
>> grid on
>> hold off
```

3. Find constants C and α such that

$$\kappa_n \approx C\alpha^n$$

Figure 2: Plot of $log(\kappa_n)$ over n (red) and $\log(\kappa_n) \approx 2.02n - 1.50$ (blue)



Solution:
From Supplemental 2 we know that $C = e^b$ and $\alpha = e^M$ so plugging in our values for M and b that we solved for in the previous problem we get that,

$$C = e^{1.50} \approx 0.2228,$$

$$\alpha = e^{2.02} \approx 7.5336.$$