

ECE 111 - Homework #1

Week #1: Matlab Introduction

Bison Academy: Homework Sets & Solutions

1) What are the two solutions to

$$y = x + \frac{1}{x}$$

$$y = 5$$

hint: See homework #2, problem #1 solutions for Spring 2025

note: If you ever get stuck on a homework set, a good place to go is under HW & Solutions. This has homework problems and solutions from previous semesters. The problems change each semester. The approach to solve and the Matlab commands used are similar, however

Next, use Newton's method to find the solution to $f(x) = 0$. Start with a guess of $x=2$:

```
x3 = 2;  
for i=1:10  
    x1 = x3;  
    y1 = Prob1(x1);  
  
    x2 = x1 + 0.001;  
    y2 = Prob1(x2);  
  
    x3 = x1 - (x2-x1)/(y2-y1) * y1;  
    disp([i, x1, y1]);  
    pause(0.1);  
end
```

Result: $x = 4.7913$

i	x	error
1	2.000000000000000	-2.500000000000000
2	5.332778147902003	0.520297668171830
3	4.793521858293856	0.002136740954666
4	4.791287916073244	0.000000065607260
5	4.791287847478571	0.000000000000623
6	4.791287847477920	0
7	4.791287847477920	0
8	4.791287847477920	0
9	4.791287847477920	0
10	4.791287847477920	0

Roots to a Polynomial

2) Use the *roots()* command to find the roots to the following.

note: Matlab represents polynomials as a 1xn matrix with each element being the coefficients in decreasing powers of x.

a) $y = x^3 + 13x^2 + 50x + 56$

```
>> P = [1,13,50,56]
```

```
P =      1      13      50      56
```

```
>> roots(P)
```

```
-7.0000  
-4.0000  
-2.0000
```

b) $y = x^4 + 34x^3 + 399x^2 + 1854x + 2808$

```
>> P = [1,34,399,1854,2808]
```

```
P =      1      34      399      1854      2808
```

```
>> roots(P)
```

```
-13.0000  
-12.0000  
-6.0000  
-3.0000
```

c) $y = x^5 + 28x^4 + 284x^3 + 1274x^2 + 2403x + 1386$

```
>> P = [1,28,284,1274,2403,1386]
```

```
P =      1      28      284      1274      2403      1386
```

```
>> roots(P)
```

```
-11.0000  
-7.0000  
-6.0000  
-3.0000  
-1.0000
```

Matlab as a Graphing Calculator:

Assume a thermistor (temperature sensitive resistor) and voltage divider have the following relationship:

$$R = 3000 \cdot \exp\left(\frac{3500}{T+273} - \frac{3500}{298}\right) \Omega$$

$$V = \left(\frac{R}{R+10,000}\right) \cdot 10V$$

3) Determine the resistance and voltage if

T = 5 degrees C (refrigerator)

```
>> T = 5
```

```
T =      5
```

```
>> R = 3000 * exp(3500/(T+273) - 3500/298)
```

```
R = 6.9837e+003
```

T = -20C (freezer)

```
>> T = -20
```

```
T =    -20
```

```
>> R = 3000 * exp(3500/(T+273) - 3500/298)
```

```
R = 2.4231e+004
```

note: Matlab can be used like any old calculator.

4) Plot the resistance vs. temperature for $-20\text{C} < T < +20\text{C}$. From the graph, determine

- The temperature when $R = 15,000$ Ohms
- The temperature when $R = 5,000$ Ohms

$$R = 3000 \cdot \exp\left(\frac{3500}{T+273} - \frac{3500}{298}\right) \Omega$$

Matlab Code

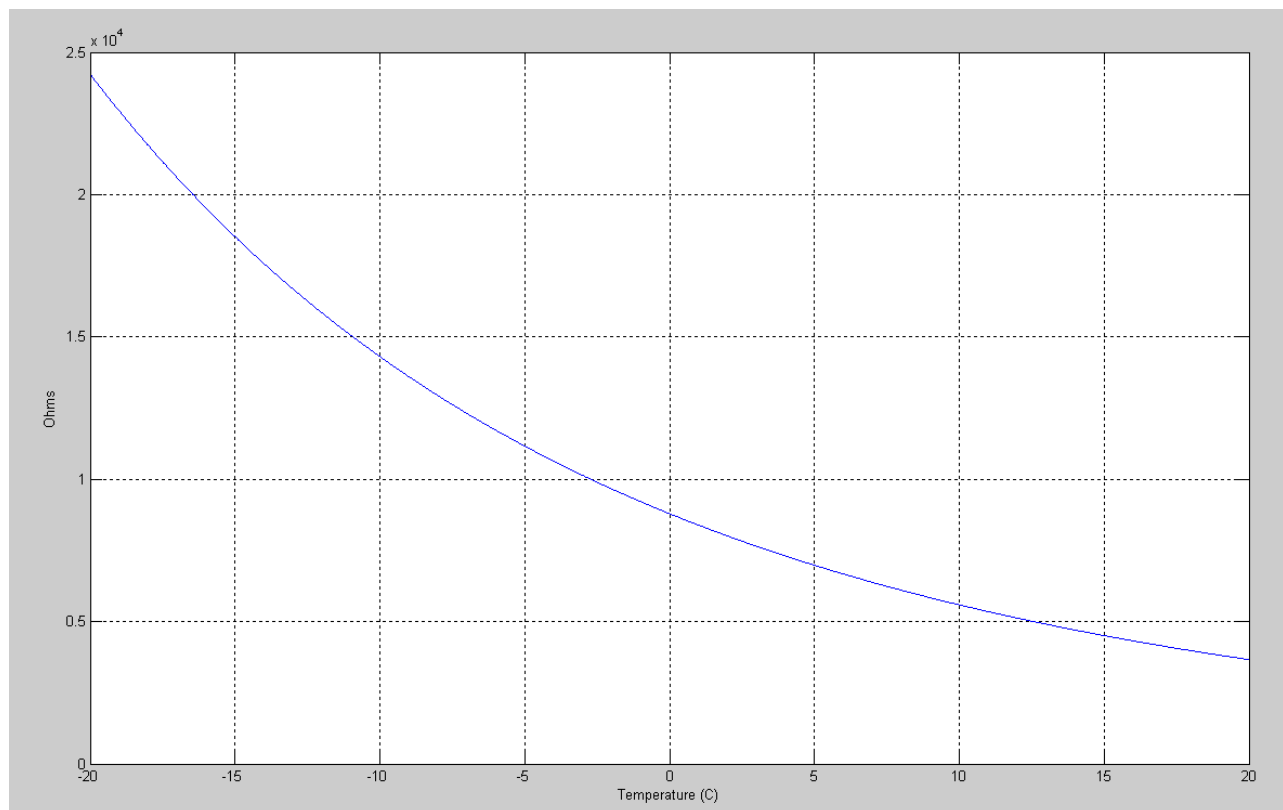
```
>> T = [-20:0.01:20]';  
>> R = 3000 * exp(3500 ./ (T+273) - 3500/298);  
>> plot(T,R);  
>> xlabel('Temperature (C)')  
>> ylabel('Ohms')  
>> grid  
>>
```

From the graph

- $R = 15,000$ Ohms $T = -12\text{C}$
- $R = 5,000$ Ohms $T = +12,5\text{C}$

note:

- Matlab is also a graphing calculator
- T goes from -20 to +20C in steps of 0.01. This gives 4000 points on the graph (maybe more than you need). Usually, 100+ points is enough for any graph (I usually try for 1000 points).
- The dot-notation tells Matlab to treat these as 4000 separate calculations rather than attempt to do matrix operations. Matlab is actually a matrix language. More on this later...



5) Plot the voltage vs. temperature for $-20^{\circ}\text{C} < T < +20^{\circ}\text{C}$. From the graph, determine

- The temperature when $V = 6.00$ Volts
- The temperature when $V = 4.00$ Volts

$$R = 3000 \cdot \exp\left(\frac{3500}{T+273} - \frac{3500}{298}\right) \Omega$$

$$V = \left(\frac{R}{R+10,000}\right) \cdot 10V$$

In Matlab

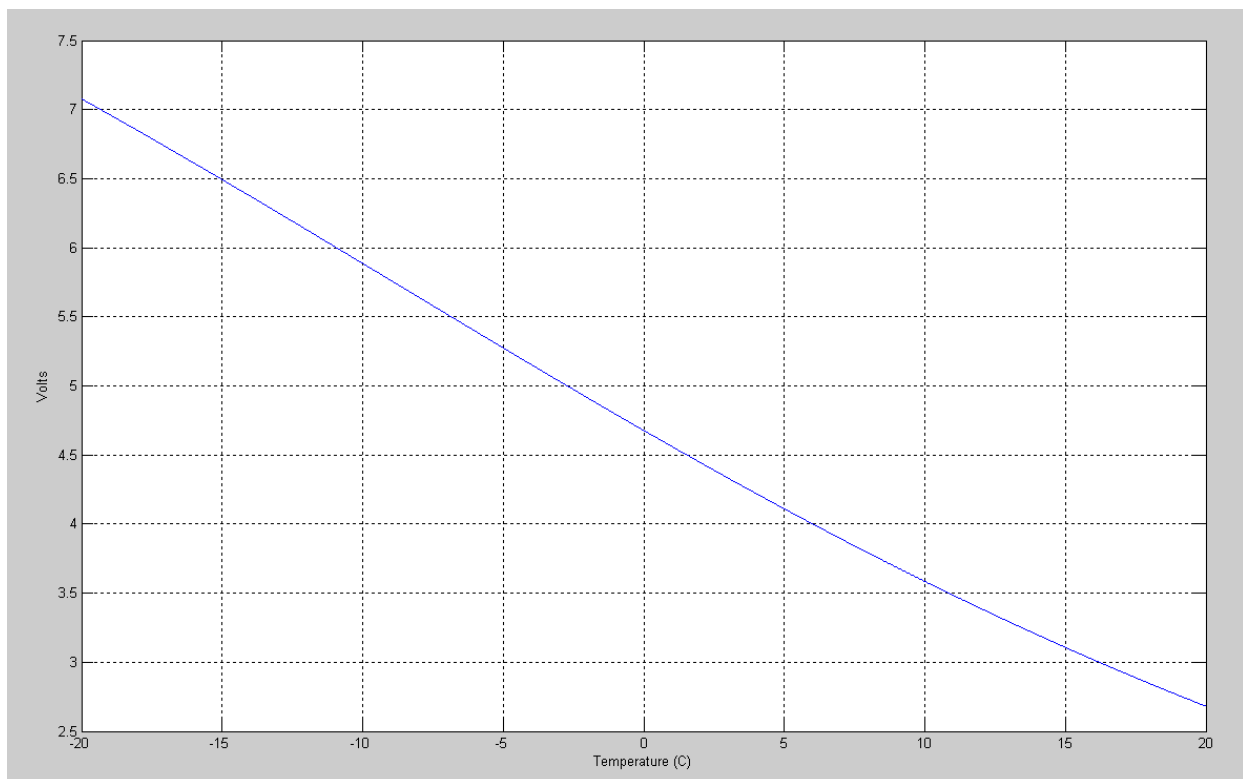
```
>> T = [-20:0.01:20]';  
>> R = 3000 * exp(3500 ./ (T+273) - 3500/298);  
>> V = ( R ./ (R + 10000) ) * 10;  
>> plot(T,V);  
>> xlabel('Temperature (C)')  
>> ylabel('Volts')  
>> grid  
>>
```

From the graph, the temperature is:

- $V = 6.00V$ $T = -12^{\circ}\text{C}$
- $V = 4.00V$ $T = +6^{\circ}\text{C}$

note:

- Once you create a variable in Matlab, you can use it in future calculations.



For-Loops

6) A and B are playing a game of dice. For each game

- A rolls four eight-sided dice and takes the sum ($A = 4d8$)
- B rolls three 12-sided dice and takes the sum ($B = 3d12$).

Whoever has the higher total wins the game (A wins on ties). Determine the odds that A wins the game using a Monte-Carlo simulation with 100,000 games.

Matlab Code (Matlab script)

```
tic
W = 0;
L = 0;
for k=1:1e5
    A = sum(randi(8, 1, 4));
    B = sum(randi(12, 1, 3));
    if(A >= B)
        W = W + 1;
    else
        L = L + 1;
    end
end
disp('A wins')
disp(W)
disp('B wins')
disp(L)

toc
```

Result

```
A wins
    44560

B wins
    55440

Elapsed time is 2.203334 seconds.
```

Comment:

- **A wins 44.5% of the time (varies)**
- Matlab scripts are convenient ways to write a cluster of Matlab code and execute them together. This is a convenient way to try out your code and fix it until the script runs correctly.
- For-loops are a convenient way to execute a section of code a fixed number of times (100,000 in this case)

7) Instead of playing a single game of dice, A and B play a match consisting of 5 games. Determine the odds that A wins the match using a Monte-Carlo simulation with 100,000 games.

Matlab Code:

```
tic
Match = 0;
for n = 1:1e5
    W = 0;
    L = 0;
    for k=1:5
        A = sum(randi(8, 1, 4));
        B = sum(randi(12, 1, 3));
        if(A >= B)
            W = W + 1;
        else
            L = L + 1;
        end
    end
    if(W > L)
        Match = Match + 1;
    end
end
disp('A wins')
disp(Match)
disp('B wins')
disp(1e5 - Match)

toc
```

Result:

```
A wins
    40351

B wins
    59649

Elapsed time is 11.827740 seconds.
```

With this format, A wins 40.3% of the time

Comment:

- You can have nested for-loops
- The inner loop (blue) repeats five times (a 5-game match)
- The output loop (red) repeats this process 100,000 times (Monte-Carlo simulation)

While-Loops

8) Instead of playing a match of 5 games, assume the match continues until one player is up 3 games. Determine the odds that A wins the match using a Monte-Carlo simulation with 100,000 games.

Matlab Code

```
tic
Match = 0;
for n = 1:1e5
    W = 0;
    L = 0;
    while(abs(W-L) < 3)
        A = sum(randi(8, 1, 4));
        B = sum(randi(12, 1, 3));
        if(A >= B)
            W = W + 1;
        else
            L = L + 1;
        end
    end
    if(W > L)
        Match = Match + 1;
    end
end
disp('A wins')
disp(Match)
disp('B wins')
disp(1e5 - Match)

toc
```

Result:

```
A wins
    34821
```

```
B wins
    65179
```

```
Elapsed time is 22.769504 seconds.
```

Comment:

- With this format, A wins 34.8% of the time
- While loops keep going until a condition is met
- While loops can put you into an infinite loop. If that happens, click on the Command window and type control-C (break). This stops the program.
- Scripts are a nice way to debug your code. If you forget to reset W and L to 0 at the start of the while-loop, you will get A or B winning 100% of the time. That doesn't seem right. When that happens, you can try different fixes in the script window to get your code to run correctly.

9) Instead of playing a match of 5 games, assume the match continues until one player is up 5 games. Determine the odds that A wins the match using a Monte-Carlo simulation with 100,000 games.

Matlab Code

```
tic
Match = 0;
for n = 1:1e5
    W = 0;
    L = 0;
    while(abs(W-L) < 5)
        A = sum(randi(8, 1, 4));
        B = sum(randi(12, 1, 3));
        if(A >= B)
            W = W + 1;
        else
            L = L + 1;
        end
    end
    if(W > L)
        Match = Match + 1;
    end
end
disp('A wins')
disp(Match)
disp('B wins')
disp(1e5 - Match)

toc
```

Result:

```
A wins
    26395

B wins
    73605

Elapsed time is 56.202795 seconds.
>>
```

Comment:

- With this format, A wins 26.3% of the time
- A nice feature of using scripts is you can sometimes solve a new problem just by changing a few lines of code in the script (the bold-line in this case)
- Monte-Carlo simulations are a simple way to compute approximate odds. They can take quite a while to execute, however (56 seconds in this case)