PHYS 361 Lab 2: Matrices and Arrays

Basic Syntax Warm-Up

The arc length of a parabola ABC of a ellpse with a semi-major axes a and b is given approximately by:

$$L_{ABC} = \frac{1}{2}\sqrt{(b^2 + 16a^2)} + \frac{b^2}{8a} ln\left(\frac{4a + \sqrt{b^2 + 16a^2}}{b}\right)$$

Determine L_{ABC} when $a = 11$in and $b = 9$in.

```
a = 11;
b = 9;

tmp_sqrt = sqrt(b.^2 + 16*a.^2);

l_abc = (1/2)*tmp_sqrt + ((b.^2) / (8*a)) * log((4*a + tmp_sqrt) / b)
```

```
l_abc = 24.5637
```

Array Warm-Up: Setting up a data grid

Let's say you want to write code that calculates the temperature in a room heated by a radiant unit embedded in the floor. You need to create a data structure capable of storing temperatures over a set of grid points. On this grid, you can use a heat transfer equation to determine how the room is heated by the floor. This problem may be too complex to solve today, but we can apply what we've learned this week to establish a grid that can help address the issue.

Let's assume the room dimensions are 10 m x 10 m x 2.5 m. Now, let's set up a variable to store the temperature of this room with a grid resolution of 10 cm or 0.1 m.

Thought process:

1.How many dimensions do you need? Ans: 3

2.How many grid points you need in each dimension? Ans: 100 points x 100 points x 25 points

3.Which of the following built-in functions can be used to initialize your matrix, assuming a constant temperature of 20 C in the room?

ones(): Creates an array of all ones

eye(): Creates an identity matrix

zeros(): Creates an array of all zeros

Look up the documentation of your choosen function and write the code to set up a variable to store the temperature of the room below.

```
temp_grid = zeros(10 / 0.1, 10 / 0.1, 2.5 / 0.1) + 20;
```

4. Remember how colons can be used to replace whole rows, columns, or pages of data with a single or set of values.

```
%Examples
matrix = [1 2 3; 4 5 6]
```

```
matrix = 2×3
     1     2     3
     4     5     6
```

```
matrix(1,:)=10
```

```
matrix = 2×3
    10    10    10
     4     5     6
```

```
matrix(:,2)=20
```

```
matrix = 2×3
    10    20    10
     4    20     6
```

```
matrix(:,end)=99
```

```
matrix = 2×3
    10    20    99
     4    20    99
```

Use colons (:) to set the temperature of the floor or the bottom temperature of your data cube equal to 23 C?

```
temp_grid(:, :, 1) = 23;
```

Array Warm-Up: Element by Element Operations

THIS IS IMPORTANT! Sometimes you want to multiple or divide or exponentiate an every element in an array. You can do that in MATLAB, but you need to use **special notation (.* , .^ , ./) .**

$$a = \begin{pmatrix} a_1 & a_2 & a_3 \end{pmatrix}$$
$$b = \begin{pmatrix} b_1 & b_2 & b_3 \end{pmatrix}$$
$$a.*b = \begin{pmatrix} a_1 b_1 & a_2 b_2 & a_3 b_3 \end{pmatrix}$$
$$a./b = \begin{pmatrix} a_1/b_1 & a_2/b_2 & a_3/b_3 \end{pmatrix}$$
$$a.^b = \begin{pmatrix} a_1^{b_1} & a_2^{b_2} & a_3^{b_3} \end{pmatrix}$$

Examples:

```
%Element-by-element multiplication, division, and exponentiation
A = [1 2 3]
```

```
A = 1×3
     1     2     3
```

```
B = [4 5 6]
```

```
B = 1×3
     4     5     6
```

```
A.*B
```

```
ans = 1×3
     4    10    18
```

```
A./B
```

```
ans = 1×3
    0.2500    0.4000    0.5000
```

```
A.^2
```

```
ans = 1×3
     1     4     9
```

```
B.^2
```

```
ans = 1×3
    16    25    36
```

```
A.^B
```

```
ans = 1×3
     1    32   729
```

Note even doing the following would get you introuble using the matrixes defined above

$$A^2$$

because that is the same as A*A and the dimensions do not agree. If you wan to square each element in A you must use the element-by-element notation:

```
% A^2     %Doesn't work because # of columns is not the same as rows
```

```
A.^2    %Works because it is squaring every element
```

```
ans = 1×3
     1     4     9
```

Define the vector $x = [1, 2, 3, 4, 5]$ and $y = [2, 4, 6, 8, 10]$.

Use these vectors to calculate the following using element-by-element operations.

(a) $z = \frac{(x+y)^2}{x-y}$ (b) $w = xln(x^2+y^2) + \sqrt{\frac{y^3}{(y-x)^3}}$

```
x = [1, 2, 3, 4, 5];
y = [2, 4, 6, 8, 10];

z = ((x + y).^2) / (x - y)
```

```
z = -36.8182
```

```
w = x .* log(x.^2 + y.^2) + sqrt( (y.^3) / ((y - x).^3) )
```

```
w = 1×5
    4.4379    8.8199   14.2484   20.3565   26.9700
```

Problem 1: Solving a Linear Set of Equations

MATLAB is designed to do the type of matrix operations you would do in a linear algebra course. Run and try to understand the example code below.

Addition and subtraction is done element-by-element, so matrices and arrays have to be the same size.

```
vectA=[8 5 4];
vectB=[10 2 7];
vectA+vectB
```

```
ans = 1×3
    18     7    11
```

```
vectA-vectB
```

```
ans = 1×3
    -2     3    -3
```

If you multiply or divide an array or matrix by a scalar, every element is multiplied or divided.

```
vectA*2
```

```
ans = 1×3
    16    10     8
```

```
vectA/2
```

```
ans = 1×3
    4.0000    2.5000    2.0000
```

If you multiple two matrixes using *, it will do it the linear algebra way (matrix multiplication):

$$A = \begin{pmatrix} 1 & 2 \\ 3 & 4 \end{pmatrix} \quad B = \begin{pmatrix} 5 & 6 & 7 \\ 8 & 9 & 10 \end{pmatrix}$$

Multiplication of two matrixes:

$$A * B = \begin{pmatrix} 1*5 + 2*8 & 1*6 + 2*9 & 1*7 + 2*10 \\ 3*5 + 4*8 & 3*6 + 4*9 & 3*7 + 4*10 \end{pmatrix}$$

$$A * B = \begin{pmatrix} 21 & 24 & 27 \\ 47 & 54 & 61 \end{pmatrix}$$

This means if you are taking A*B then A must have the same number of columns as the number of rows in B (like in the example above).

```
A = [1 2; 3 4];            %Define A and B
B = [5 6 7; 8 9 10];

size(A)                    %Print the dimension
```

```
ans = 1×2
     2     2
```

```
size(B)
```

```
ans = 1×2
     2     3
```

```
C=A*B                          %Multiply
```

```
C = 2×3
    21    24    27
    47    54    61
```

```
size(C)                        %Show the dimensions of results
```

```
ans = 1×2
     2     3
```

Matrix division is a little more complex. MATLAB can do two different forms of division: left division (\) and right division (/).

If you have a matrix problem that you want to solve that looks like this:

$$AX = B.$$

You can find the solution by multiplying both sides by the inverse of A (A^{-1}):

$$A^{-1}A = I$$

where I is the identity matrix. Therefore,

$$A^{-1}AX = IX = X$$

So the solution to the original equations is:

$$X = A^{-1}B$$

In MATLAB, this is the same as doing what it calls left division:

$$X = A\backslash B$$

On the other hand, you might need to solve this equation for X:

$$XC = D$$

Since *X* and *C* are matrices, you can't just switch the order of X and C (it's not communitive). However, you can multiply the right hand side of each equation by the inverse of C.

$$XCC^{-1} = DC^{-1}$$

$$X = DC^{-1}$$

In MATLAB this is the same as doing right hand division:

$$X = D/C$$

Example: Solving a system of linear equations.

$$4x - 2y + 6z = 8$$
$$2x + 8y + 2z = 4$$
$$6x + 10y + 3z = 0$$

You have three equations and three unknowns; therefore, through substitution and algebra, you could solve them by hand. OR you could write it as a matrix problem.

$$\begin{pmatrix} 4 & -2 & 6 \\ 2 & 8 & 2 \\ 6 & 10 & 3 \end{pmatrix} \begin{pmatrix} x \\ y \\ z \end{pmatrix} = \begin{pmatrix} 8 \\ 4 \\ 0 \end{pmatrix}$$

We can rewrite this problem by defining matrixes *A, X,* and *B*, where *AX=B*. Then we can use MATLAB to solve for X using left hand division:

```
A=[4 -2 6; 2 8 2; 6 10 3];
B = [8; 4; 0];              %Note I made this a column array so the dimensions work
X=A\B                       %Using left division
```

```
X = 3×1
```

```
   -1.8049
    0.2927
    2.6341
```

```
A*X                          %Confirming I did it right, should be equal to B
```

```
ans = 3×1
    8.0000
    4.0000
         0
```

Since X = [x y z], I know x=8, y=4, and c=0 in the system of equations I started with.

Problem 1:

Solve the following system of linear equations

$$-4x + 3y + z = 18.2$$
$$5x + 6y - 2z = -48.4$$
$$2x - 5y + 4.5z = 92.5$$

Write an additional line of code to prove you are correct.

```
A = [-4, 3, 1; 5, 6, -2; 2, -5, 4.5];
B = [18.2; -48.4; 92.5];

X = A\B
```

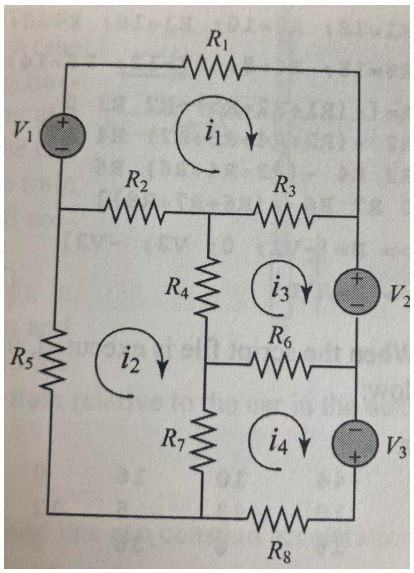
```
X = 3×1
   -0.5138
   -1.1285
   19.5301
```

```
A*X
```

```
ans = 3×1
   18.2000
  -48.4000
   92.5000
```

Problem 2:

In this problem, you will analyze the following circuit:



Kirchoff's voltage law states that the sum of the voltage around a closed circuit is zero. The sources and sinks of voltage along the loops can be batteries (labeled as V above) and resistors labeled as R. The change over a resistor is always negative and given by Ohm's Law, $\Delta V = iR$.

The **mesh current law** states that the current going through any resistor is equal to the sum of the current in connecting "meshes" or loops. For example, R_2, in the top loop connected to two meshes (or rectangles of wire). It is standard to assume that the current is moving in the clockwise direction over each loop. That means there is current moving from right to left through R_1 from the top mesh and left to right through R_1 from the bottom mesh. Therefore, if we call the current moving through the top mesh i_1 and the bottom mesh (connected to R_1) i_2 the current going through R_1 is,

$$i_1 = (i_1 - i_2)$$

Lets use this convention along the whole top loop, Kirchoff's law gives us the following equation,

$$V_1 - R_1 i_1 - R_3(i_1 - i_3) - R_2(i_1 - i_2) = 0\,.$$

Since we are writing the equation for the top loop or loop 1, the current going through that loop, i_1, is always positive and the opposing current is always subtracted from it.

We also have to worry about the direction that the current goes through the batteries (label V). Current is always assumed to be positive charge, so when it moves from the negative to the positive terminal of the battery we assume it "gained" voltage and the V is positive, like in the example for the top loop. When current moves from the positve to the negative terminal we assume that it "losses" voltage and the V is negative. For example, Kirchoff's law gives us the following for the loop with current i_3.

$$-V_2 - R_6(i_3 - i_4) - R_4(i_3 - i_2) - R_3(i_3 - i_1) = 0$$

Let's analyze this system:

Step 1: By hand, write out two more equations describing the sum of voltages along the other 2 closed loops. Have the instructor check your work.

Step 2: Rearrange these equations (by hand) to that they can be written in the following form

$$RI = V$$

where R and V are matrices that include the resistance (R) and voltage (V) terms and

$$I = \begin{pmatrix} i_1 \\ i_2 \\ i_3 \\ i_4 \end{pmatrix}$$

You shoud get something like:

$$[a\ 4\text{x}4\ \text{array with}\ R's\ \text{in it}] \begin{bmatrix} i_1 \\ i_2 \\ i_3 \\ i_4 \end{bmatrix} = \begin{bmatrix} -V_1 \\ 0 \\ V_2 \\ -V_3 \end{bmatrix}$$

Step 3: In MATLAB, define the following variables

$$V_1 = 20V, V_2 = 12V, V_3 = 40V$$

$$R_1 = 18\Omega, R_2 = 10\Omega, R_3 = 16\Omega, R_4 = 6\Omega, R_5 = 15\Omega, R_6 = 8\Omega, R_7 = 12\Omega, R_8 = 14\Omega$$

Step 4: Use matrix algebra to solve for the current in each loop (given by the array defined above).

```
V1 = 20;
V2 = 12;
V3 = 40;
R1 = 18;
R2 = 10;
R3 = 16;
R4 = 6;
R5 = 15;
R6 = 8;
R7 = 12;
R8 = 14;

V = [V1; 0; V2; V3];
A = [(R1+R2+R3), -R2, -R3, 0; -R2, (R4+R2+R5+R7), -R4, -R7; R3, R4, -(R3+R4+R6), R6; 0, -R7, -R6, (R6+R7+R8)];

X = A\V
```

```
X = 4×1
    0.8411
    0.7206
    0.6127
    1.5750
```

```
A*X
```

```
ans = 4×1
```

```
    20.0000
     0.0000
    12.0000
    40.0000
```

Problem 3: The idea gas law

The ideal gas law is

$$P = \frac{nRT}{V}$$

where P is the pressure, V is the volume, T is the temperature, R is the gas constant (R=0.08206 (L atm)/(mol K)), and n is the number of moldes. Real gases, especially at high temp, deviate from this behavior. Their response can be modeled with the van der Waals equation

$$P = \frac{nRT}{V - nb} - \frac{n^2 a}{V^2}$$

where a and b are material constants. Consider 1 mole (n=1) of nitrogen gas at T=50 K. For nitrogen, a=1.39 (L^2 atm)/mol^2 and b=0.0391 L/mol.

Step 1: Creat a vector with values of volume (V) for $0.1 \leq V \leq 1$L, using increments of 0.02 L.

Step 2: Using this vector calculate P twice for each value of V, once using the ideal gas equation and once using the van der Waals equation.

Step 3: Using the two sets of values for P, calcualte the percent error $\left|\left(\frac{P_{ideal} - P_{waals}}{P_{waals}}\right) \times 100\right|$. You should use the built-in function `abs`.

Step 4: Finally, using MATLAB's built-in fuction `max`, determine the maximum error and the corresponding volume. (Hint: Look at the help file for the max function to see how you can use it to determine both the highest value and its location in the array.)

```
% define gas constants
% moles of gas
n = 1;
% gas temperature [Kelvin]
T = 50;
% gas constant [L atm / (mol K)]
R = 0.08206;
% [L^2 atm / mol^2]
a = 1.39;
% [L / mol]
b = 0.0391
```

```
b = 0.0391
```

```

% define a list of volumes
volumes = 0.1 : 0.02 : 1;
sqr_vol = volumes .^ 2;

% pre-calculate some values
nRT = n*R*T;
nb = n*b;

% calculate the pressure using the ideal gas law
p_ideal = nRT ./ volumes;

% calculate the pressure using the van der Waals equation
p_van = (nRT ./ (volumes - nb)) - (a * n.^2) ./ sqr_vol;

% find the percent error
percent_error = abs( (p_ideal - p_van) ./ p_van ) * 100;

% determine which volume element corresponds to the max error
[max_er, max_in] = max(percent_error)
```

```
max_er = 4.7518e+03
max_in = 11
```