ENGR105 --- Problem Set 2 --- Variables and Vectors

Due: Friday, 9/11/2020

Name:

The University of Pennsylvania code of academic integrity can be found here:

https://catalog.upenn.edu/pennbook/code-of-academic-integrity/

By submitting this assignment, you certify that you completed this assignment in compliance with the code of academic integrity, as defined above, and that the work here is your own. Collaboration is allowed and encouraged but it is in your own best interest to be able to complete all the problems on your own. Collaboration is allowed, COPYING is NOT. Be sure you are able to reason through each step.

Notes on HW submission:

- Please save the final document In the following format: "<your last name>_<your first name>_HW2.mlx
- Run the entire document (`f5`) prior to submitting. Following this, please **export to .pdf** and **submit both** the .mlx file and the .pdf file to Canvas.
- The LiveScript file may be exported to PDF using the **Save** dropdown menu.
- If it is not selected by default, select "Output Inline" at the top right of the LiveScript document to display the output below your code (rather than to the right).

Purpose of this problem set:

- Develop familiarity defining vectors, using the `help` function, and manipulating variables.
- Learn how to determine the time required for a piece of code to run.
- Practice creating, manipulating, and performing mathematical operations on vectors.

Notes about MATLAB LiveScripts:

- The gray regions are "code" regions. Entering and running your code here is the same as running it in a `.m` script file or on the command line.
- "Running a piece of code" means you are executing the code in that section only. **Press `Cntl` + `Enter` to run a given section.**
- Press `f5` to run all sections of the document.
- To remove inline figures, right click and select "Clear All Output."
- When you run a section (each section refers to a different problem or part of a problem), all variables and vectors are stored in MATLAB and may be called from the command line.

Reminder: Prior to submitting, be sure to run all sections of the document to ensure that it works properly. After running all sections, export the document to pdf.

PROBLEM 1 (15 pts): Valid MATLAB variables

It's a late night and you have been working hard on your engineering projects. You are currently creating a new script and need to define a variable, however you are also sleepy and having a hard time picking a name. The potential names that come to mind are listed below.

For each of the following, state whether or not it is a valid variable name in MATLAB. Remember, valid variables should be able to store an scalar or vector, should not overwrite built-in functions, and should not overwrite built-in variables. Please type "**Yes**" next to valid variable names and "**No**" next to invalid names. For each invalid name, add a statement as to why it is not valid.

a) `newVar`

`Yes`

b) `new Var`

`No because you cannot have spaces when creatign a variable.`

c) `var`

`No because var is already a function.`

d) `engr`

`Yes`

e) `ENGR`

`Yes`

f) `engr_`

`Yes`

g) `en_gr`

`Yes`

h) `en-gr`

`No because the character of a hyphen does not work for creating variables.`

i) `_engr`

`Yes`

j) `puppies please`

`No because the space makes it invalid.`

k) `puppiesplease`

`Yes`

l) `pizza*5`

```
No because the * cannot be in variables since it is already a built in function.
In addition it is a character that is not an underscore so it cannot function.
```

m) `1var`

```
No because variables cannot begin with a number.
```

n) `var1`

```
Yes
```

o) `one`

```
Yes
```

PROBLEM 2 (15 pts): A vector of random numbers

Consider (and run) the following code that uses a `for` loop to produce a vector of random numbers between 0 and 1.

```
% Populate an array of 10 random numbers using a for loop
for jj = 1:10
    randArr(jj) = rand;
end
```

Note that `rand` outputs a random number uniformly distributed between 0 and 1.

a) Produce a vector of random numbers, `myRandArr`, with the same dimensions (but not necessarily the same exact numbers) as in the example above using a single expression. **Your code should be in the form of a (one-line) script.**

```
myRandArr = rand(1,10);
```

Hint: the MATLAB documentation for `rand` may be helpful.

b) Calculate the computational time required to produce a `1 x 10000` vector (10000 entries) of random numbers using the approaches that produced `randArr` and `myRandArr`. The `tic` and `toc` commands may be helpful here. Which array is quicker to create?

```


tic;
myRandArr = rand(1,10000);
toc;
```

```
Elapsed time is 0.002786 seconds.
```

```

tic;
for jj = 1:10000
    randArr(jj) = rand;
end
toc;
```

```
Elapsed time is 0.003200 seconds.
```

The one that was not a for loop seemed to be faster most of the time than the for loop.

PROBLEM 3 (20 pts): Vector manipulations

Consider the vector x:

```
x = [5, 4, 2, 1, 9];
```

Using only `x`, subvectors specified by calling multiple indices of `x`, basic math operations, and vector concatenation, produce code for each of the outputs specified in **a) - f)**. You should not explicitlly define individual entries into the new vector.

For clarification, the following are examples of acceptable code for this problem:

```
a = x(2:4)
a = x([5,2,1])
a = 5*x
a = x([4,2]) + 2
a = [x(3:4).*x(1:2),x(3:4)]
```

The following are examples of unacceptable code for this problem:

```
a = [x(1),x(3),x(2)]
a = [x(1),5,x(2)]
a = [0,x([4,2,5])]
```

Your answer for each should be one line of code

a) `a = [1      2      4      5      9]`

```
a = x([4,3,2,1,5])
```

```
a = 1×5
     1     2     4     5     9
```

b) `b = [1      9      5      5]`

```
b = x([4,5,1,1])
```

```
b = 1×4
     1     9     5     5
```

c) `c = [25     16      4      1     81]`

```
c = x.^2
```

```
c = 1×5
    25    16     4     1    81
```

d) d `= [3      2      0]`

```
d = x([1:3]) - 2
```

```
d = 1×3
     3     2     0
```

e) e = [3   2   0   0   2   3]

```
e = [x(1:3) - 2, x([3,2,1]) - 2]
```

```
e = 1×6
     3     2     0     0     2     3
```

f) f = [14   5   4   5   14]

```
f = [x([1,2,3] )+x([5,4,3]), x([2,1])+x([4,5])]
```

```
f = 1×5
    14     5     4     5    14
```

You do not need to provide comments for your code in this section.

PROBLEM 4 (25 pts): Mathematical expressions with vectors

Translate the following mathematical expressions into MATLAB statements. The output of each expression should be assigned to a valid variable name of your choice. All symbolic variables represent vectors with the same dimensions of 1 x 10. For convenience, these are defined here:

```
p = 1:10;
q = 10:-1:1;
x = rand(1,10)+1;
y = linspace(1,25,10);
```

All computations are intended to be performed element-by-element (i.e. not matrix multiplication of division) such that the expressions resolve to a 1 x n array. Only utilize array operations (e.g. `.*` and `./`) when absolutely necessary.

a) $p + \frac{2}{q}$

```
p + (2./q);
```

b) $q \log(p)$ (where $\log$ is the base-10 logarithm)

```
q .* log(p);
```

c) $1 + \frac{1}{\sin(x)}$

```
1+(1./sin(x));
```

d) $x^{\frac{1}{4}}$

```
x.^25;
```

e) $50y^{p^2}$

```
50*y.^(p.^2);
```

f) x^{y^x}

```
x.^(y.^x);
```

g) $\dfrac{p+\dfrac{x}{y+q}}{p+\dfrac{x}{y-q}}$

```
(p+(x./(y+q)))./(p+(x./(y-q)));
```

h) $x-\dfrac{x^3}{3!}+\dfrac{x^5}{5!}$

```
%make vectors for 3! and 5! and then multiply up
x-(x.^3/factorial(3))+(x.^5/factorial(5))
```

```
ans = 1×10
    1.0047    0.8589    0.9334    1.0043    0.9714    1.0045    0.9095    1.0047 ···
```

```
```

Be sure to suppress your code with semicolons (`;`) when you submit this problem.

PROBLEM 5 (25 pts): Decoding a secret message

One day while walking past Van Pelt, you notice a cryptic note taped to a park bench. Intrigued, you look closer and find a number that resembles a vector written at the top of the note. This mystery vector reads:

```
% Mystery vector found at a park bench
mVec = [1, 9, 9, 1, 3, 3, 9, 3, 8, 5, 6, 1, 3, 1, 4, 7, 2, 2, 1, 9, 5, 1];
```

That isn't all, however -- a series of intructions are written below it:

1. Delete every 3rd index starting with the first ind ex (*e.g.* indices 1, 4, 7, 10, etc.)
2. Swap even and odd indices (*e.g.* index 1 goes to 2, index 2 goes to 1, index 3 goes to 4, index 4 goes to 3, etc)
3. Take the last 6 indices and move them to the front of the vector
4. Subtract 1 from every even index (*e.g.* indices 2, 4, 6, 8, etc.)
5. Remove an equal number of indices from the left and right hand side of the vector to yield a 10-entry vector

Fortunately, you are a sleuth with MATLAB skills that you can use to decode this number! **Create a script (MATLAB algorithm) below to decode the mystery vector**. Make sure your code is clear and do not suppress the final output of your algorithm. You may find the `length()` command useful when creating your decoding algorithm.

```
%Should store the length in the beginning of the script as a variable
%Part 1: Remove every third index
mVec_length = length(mVec);
mVec(1:3:end) = [];
%storing the even number so that we can later input them into the odd
%positions
%Part 2: Swith the even and odd indices
mVec_even = mVec(2:2:end);
mVec(2:2:end) = mVec(1:2:end);
mVec(1:2:end) = mVec_even;
%switches the first 6 indeces with the last six
mVec = [mVec(end-5:end), mVec(1:end-6)];
%Part 4: Remove 1 from all even indices
mVec(2:2:end) = mVec(2:2:end) - 1;

%will continue to remove entities form the front and the end until the
%length is equal 10
%Part 5
while length(mVec)~=10
    mVec(1) = [];
    mVec(end) = [];
end
disp(mVec)
```

```
     2     1     5     8     9     8     3     2     8     2
```

```
% Found out that this was a phone number for UPenn Computer Connection
```