PHYS 210: Introduction to Computational Physics MATLAB Exercises 2

Using your text editor, and working within `~/matlab`, create a file named `ex2.m` that contains `Matlab` commands to solve the exercises enumerated below. Note that `ex2.m` will be a `Matlab` *script*.

IMPORTANT! Be sure that you create/save `ex2.m` in the directory `~/matlab`.

IMPORTANT! Although all of the commands that you need to solve the problems should *eventually* be entered in `ex2.m`, you will probably find it more convenient to "experiment" interactively at the command prompt to solve some/most/all of the exercises, then copy the command(s)—using a cut-and-paste technique—that you have used to `ex2.m`. You should then re-save `ex2.m` and execute

```
>> ex2
```

to ensure that you have recorded your solutions correctly.

IMPORTANT! As was the case for the previous set of exercises, if you see the following error message

```
>> ex2
Undefined function or variable 'ex2'.
```

then it is probable that one or more of the following is true:

- You didn't start `matlab` from the command line, and from within the directory `~/matlab`
- You didn't name the file that contains the `matlab` commands `ex2.m`
- You didn't save `ex2.m` in the directory `~/matlab`.

In completing the exercises, you may find useful the following `matlab` examples, as well as the notes at

```
http://laplace.physics.ubc.ca/210/Doc/matlab/matlab2.html
```

that we covered in the last lab.

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% PHYS 210: Introduction to Compuational Physics
%
% Hints/examples for Matlab Exercises 2
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% Create a row vector with constant spacing using colon operator

>> rv1 = [2:3:14]

rv1 =

     2     5     8    11    14

% Create a row vector with constant spacing using linspace command

>> rv2 = linspace(1.0, 2.0, 6)

rv2 =

    1.0000    1.2000    1.4000    1.6000    1.8000    2.0000

% Select 4th element of rv1

>> rv1(4)

ans =   11

% Create 2 x 3 matrix
```

```
>> m1 = [sqrt(3), exp(2), pi; cosd(45), 6, 7.2]

m1 =

   1.73205   7.38906   3.14159
   0.70711   6.00000   7.20000

% Select element in second row, second column of m1

>> m1(2,2)

ans =  6

% Create a length 10 row vector with all elements = 7
% (assignment of multiple elements).

>> xrow(1:10) = 7

xrow =

   7   7   7   7   7   7   7   7   7   7

% Transpose xrow to make a length 10 column vector (single quote
% is the transpose operator).

>> xcol = xrow'

xcol =

   7
   7
   7
   7
   7
   7
   7
   7
   7
   7

% Another way to create xrow as defined above.

>> xrow1 = linspace(7, 7, 10)

xrow1 =

   7   7   7   7   7   7   7   7   7   7

% Create a length six row vector with 6th element = 5,
% elements 1 through 5 = 0 (implicit assignment).

>> v6(6) = 5

v6 =

   0   0   0   0   0   5


% Create a length 10 row vector with elements 1, 2, ... 10
% then assign 6th through last elements to 3, 5, 7, 8 and 12
% (assignment of multiple elements using vector of conformant
% size).

>> z10 = (1:10)

z10 =

    1    2    3    4    5    6    7    8    9   10

>> z10(6:end) = [3 5 7 8 12]
```

```
z10 =

    1    2    3    4    5    3    5    7    8   12

% Create a row vector with elements 1, 2, 3, 4 followed by
% 10, 9, 8, 7, 6 (concatentation of two row vectors to produce
% a third).

>> vcatrow = [1:4 10:-1:6]

vcatrow =

    1    2    3    4   10    9    8    7    6

% Create a short vector (vdoubleme), contatenate it to itself,
% then reassign to vdoubleme (note optional use of comma to
% separate vectors being concatenated).

>> vdoubleme = 1:3

vdoubleme =

   1   2   3

>> vdoubleme = [vdoubleme, vdoubleme]

vdoubleme =

   1   2   3   1   2   3

% Create a column vector with elements 0.1, 0.2, 0.3, 0.4, 0.5
% followed by 20, 18, 16, 14 (transpose and column concatentation).

>> vcatcol = [linspace(0.1, 0.5, 5)'; linspace(20, 14, 4)']

vcatcol =

    0.10000
    0.20000
    0.30000
    0.40000
    0.50000
   20.00000
   18.00000
   16.00000
   14.00000

% Create a 2 x 3 matrix with all elements = pi (assignment of
% multiple elements).

>> pi23(1:2, 1:3) = pi

pi23 =

   3.1416   3.1416   3.1416
   3.1416   3.1416   3.1416

% Create a 4 x 5 matrix with element (4,5) = -1 and all other
% elements = 0 (implicit assignment).

>> pi45(4,5) = -1

pi45 =

   0   0   0   0   0
   0   0   0   0   0
   0   0   0   0   0
   0   0   0   0  -1

% Set values of upper-left 2 x 2 sub-matrix of pi45 to 1
% (assignment of sub-matrix using conformant matrix of values).
```

```
>> pi45(1:2, 1:2) = ones(2)

pi45 =

   1   1   0   0   0
   1   1   0   0   0
   0   0   0   0   0
   0   0   0   0  -1

% Matrix defined via concatenation of conformant matrices
% along rows (constituents must have same number of columns).

>> mcatrow = [eye(3) ones(3) zeros(3)]

mcatrow =

   1   0   0   1   1   1   0   0   0
   0   1   0   1   1   1   0   0   0
   0   0   1   1   1   1   0   0   0

% Matrix defined via concatenation of conformant matrices
% along columns (constituents must have same number of rows).

>> mcatcol = [eye(3); ones(3); zeros(3)]

mcatcol =

   1   0   0
   0   1   0
   0   0   1
   1   1   1
   1   1   1
   1   1   1
   0   0   0
   0   0   0
   0   0   0

% Three ways to define 5 x 4 matrix with alternating columns
% of 1's and 0's.  First method uses transpose operator to
% create columns, and concatenates rows; second defines
% alternating rows, then transposes resulting matrix; third
% defines 5 x 2 matrix, then uses concatenation/reassignment
% of/to itself.

>> mat54a = [ones(1,5)' zeros(1,5)' ones(1,5)' zeros(1,5)']

mat54a =

   1   0   1   0
   1   0   1   0
   1   0   1   0
   1   0   1   0
   1   0   1   0

>> mat54b = [ones(1,5); zeros(1,5); ones(1,5); zeros(1,5)]'

mat54b =

   1   0   1   0
   1   0   1   0
   1   0   1   0
   1   0   1   0
   1   0   1   0

>> mat54c = [ones(1,5); zeros(1,5)]'

mat54c =

   1   0
   1   0
   1   0
   1   0
```

```
   1   0

>> mat54c(1:5, 3:4) = mat54c

mat54c =

   1   0   1   0
   1   0   1   0
   1   0   1   0
   1   0   1   0
   1   0   1   0
```

And now it's your turn ...

Problems from Gilat, Ch. 2.11

2.11) Using the colon symbol, create a row vector (assign it to a variable named `same`) with seven elements that are all -3.

2.14) Create a vector (name it `vecA`) that has 14 elements of which the first is 49, the increment is -3, and the last element if 10. Then, using the colon symbol, create a new vector (call it `vecB`) that has 8 elements. The first 4 elements are the first 4 elements of the vector `vecA`, and the last 4 are the last 4 elements of the vector `vecA`.

2.17) Create the following matrix by using vector notation (the colon symbol) for creating vectors with constant spacing and/or the `linspace` command. Do not type *any* individual elements explicitly. (Hint: Use the transpose operator, `'`)

$$r217 = \begin{bmatrix} 1 & 0 & 3 \\ 2 & 0 & 3 \\ 3 & 0 & 3 \\ 4 & 0 & 3 \\ 5 & 0 & 3 \end{bmatrix}$$

2.20) Create the following matrix by typing one command. Do not type *any* individual elements explicitly.

$$D = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 6 & 6 \\ 0 & 0 & 0 & 6 & 6 \end{bmatrix}$$

2.30) Create the following matrix B without typing individual elements

$$B = \begin{bmatrix} 18 & 17 & 16 & 15 & 14 & 13 \\ 12 & 11 & 10 & 9 & 8 & 7 \\ 6 & 5 & 4 & 3 & 2 & 1 \end{bmatrix}$$

Use the matrix B to

1. Create a six-element column vector named `va` that contains the elements of the second and fifth columns of B
2. Create a seven-element column vector names `vb` that contains the elements 3 through 6 of the the third row of B and the elements of the second column of B.

Do not type *any* individual elements explicitly.

2.37) Using the `zeros` and `ones` commands, create the following array, `r237`.

$$r237 = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & 1 \end{bmatrix}$$