ELE 215 – Exercise 1 – MatLab Programming

Objectives

- To review MatLab, writing a program to solve any one of three given problems.

Notes

- Each student works independently on this assignment; if you have questions, ask me and not your classmates or the lab TAs. Plagiarism will be punished. Note that I am very willing to answer questions, but will not debug your code.

Procedure

1. Choose a problem and develop your code:

   a) Below are descriptions for 3 different programming tasks: "Resistor Network", "Particle Movement," and "Spiral Matrix Construction." You are to choose one.
   b) Whichever you choose, you are to write a single MatLab function to solve the problem; I have provided simple dummy functions (for the HW ID 999) on the website for each problem; use this as a template for your solutions. Keep the filename the same, just replacing the 3 digits of the HW ID with your own final 3 digits.
   c) I will test your function on 5 different sets of inputs to establish that it performs correctly; the scoring will be 15 points per trial for the total of 75 points.
   d) Beyond defining the output variable(s) requested, your function should not print anything to the command window. In fact, I will penalize extra outputs. Also, it must perform as a MatLab function as that is how I will test it.

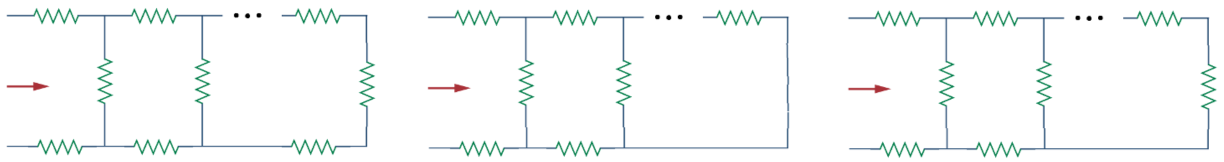
2. Submission:

   a) Check your result extensively before submitting !! Programs that do not perform as a function or abort during running will receive a score of 0.
   b) Submit your program (as a .m file) by uploading it to the ELE 215 Brightspace site
   c) The program is due by 9 AM on April 6, quite a while from now to give you a chance to revisit your EGR 106 material is needed.
   d) Late submissions will not be accepted.

Resistor Network

Your function will get as its single input a vector of positive numbers which is to be interpreted as resistor values; the goal of the function is to compute the equivalent resistance of a circuit containing those resistors.

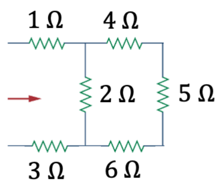
The specific circuit is one of 3 forms depending upon the number of resistors:



In the first (left) of these figures the number of resistors is a perfect multiple of 3; in the second (middle) it is one more than a multiple of 3; and in the third (right) it is 2 more than a multiple of 3. The equivalent resistance is to be computed for the two terminals on the left (the red arrow).

Assumptions:

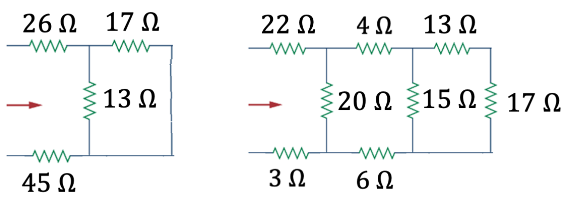
- The number of resistors is unspecified, but larger than 3
- The list of resistors fills the reversed the circuit in reversed C patterns from left to right. To explain this, consider the input vector [1 2 3 4 5 6]. Then the circuit would be as shown to the right.
- The resistor values will NOT be not be in order, 1, 2, etc.; that is purely for demonstration of the ordering



Some simple examples: for the circuit above series and parallel combining results in $\frac{98}{17} =$ 5.7647 Ω; hence, running your function should results with

```
>> resistor_999([1 2 3 4 5 6])
ans =
  5.7647e+00
```

More realistically for 4 or 8 resistors:



```
>> resistor_999([ 26, 13, 45, 17 ])
ans =
  7.8367e+01
>> resistor_999([ 22, 20, 3, 4, 15, 6, 13, 17 ])
ans =
   35
```

Particle Movement

Your function will get as its single input a matrix of positive numbers which is to be interpreted to describe movements of a particle on the (x, y) plane; the goal of the function is to compute the ending position of the particle.

Assumptions:

- The size of the input matrix is unknown but at least 2-by-2 (and it might not be square)
- The important entries may be positive or negative; zeros are to be ignored
- The individual entries, read left to right and top to bottom, described in an alternating fashion the movements in x and y
- The particle starts at the origin, $(x, y) = (0,0)$
- The first movement is in the x direction
- Zero entries do not result in a change or direction (in other words, we never have 2 movements in either direction in a row)

As a first example if the given matrix is $\begin{bmatrix} 1 & 4 & 0 & -2 \\ 1 & 0 & 3 & 0 \end{bmatrix}$ then the particle's movement is

$$(0,0) \underset{\text{"1 in x"}}{\longrightarrow} (1,0) \underset{\text{"4 in y"}}{\longrightarrow} (1,4) \underset{\text{" }-2\text{ in x"}}{\longrightarrow} (-1,4) \underset{\text{"1 in y"}}{\longrightarrow} (-1,5) \underset{\text{3 in x}}{\longrightarrow} (2,5)$$

in which the zeros are skipped. The final result is $(x, y) = (2.5)$ and running your function I should see

```
>> [x,y] = particle_999([1,4,0,-2;1,0,3,0])
x =
    2
y =
    5
```

And a second example for matrix s $\begin{bmatrix} 1 & 3 & 3 \\ 4 & 0 & 5 \\ 6 & 7 & 8 \end{bmatrix}$ we should see

```
>> [x,y] = particle_999([1,2,3;4,0,5;6,7,8])
x =
   16
y =
   20
```

Spiral Matrix Construction

Your function will get as its single input a vector of integers; the goal of the function is to construct a square spiral matrix (defined below) using as many of the values as possible.

Define a square spiral matrix as a matrix in which we enter the values in a clockwise spiral fashion starting at the top left. For example, for 9 entries in our input vector $x_1, x_2, \dots\ x_9$T then the matrix would be

$$\begin{bmatrix} x_1 & x_2 & x_3 \\ x_8 & x_9 & x_4 \\ x_7 & x_6 & x_5 \end{bmatrix}$$

Clearly the number of elements in such a matrix is a perfect square.

Assumptions:

- The size of the input vector is unknown but has at least one element
- If the number of elements in the input vector is NOT a perfect square, the function should also output a vector of the unused elements; if the number is a perfect square, this second output should be the empty vector

Some simple examples: for the 10 elements [1 2 3 4 5 6 7 8 9 10] then the output should be a 3-by-3 matrix and a scalar remainder:

```
>> [mat,rem] = spiral_999( [ 1 2 3 4 5 6 7 8 9 10 ])
mat =
     1     2     3
     8     9     4
     7     6     5
rem =
    10
```

Another example:

```
>> [mat,rem] = spiral_999( [ 1 3 5 67 40 12 2 1 0 0 0 10 20 30 22 -4 9 9 7 9  ])
mat =
     1     3     5    67
    10    20    30    40
     0    -4    22    12
     0     0     1     2
rem =
     9     9     7     9
```