Project **04/05/2017**

The overall goal of this project is to develop skills to program common operations in matrix algebra such as matrix multiplication, matrix diagonalization, transforming the matrix, matrix diagonalization, etc. Also, this project provides opportunities for exploring and using MATLAB built-in functions and capabilities that we did not discuss in the class.

This project asks for a graphical representation of a symmetric 2-by-2 matrix that renders all possible rotations (transformations) of the matrix in a 2-D plane. A well-known example of this problem in the field of Aerospace/Mechanical Engineering is Mohr's circle that represents the rotations of a stress tensor matrix in 2-D plane. However, here, we present this project as a more general mathematical problem in matrix algebra.

The final outputs of the projects are the GUI figure showing the results for the requested values (given at the end of this file), and the display of the answers (in the command windows) to the question asked in some parts and are marked in green. The parts marked in orange are not required to answer/explore, but is included for interested students who want to learn more about matrix algebra related to this project.

The project should include all the following steps

1- Start with a main function (main_func.m) that takes a symmetric matrix **S** as its input and provides the values asked in Parts 2, 3, 5, 6. The main function should be the one called from .m file of your GUI. Other functions mentioned below can be made as subfunctions within the main function or separate .m function.

2- Write a function (EigVal.m) that finds the eigenvalues of **S** (input of the function) using the following formula

$$\det(\mathbf{S} - \sigma\mathbf{I}) = 0$$

where **I** is the 2-by-2 identity matrix (eye(2)). Inside EigVal.m, write the left hand side of Eq. (1) in an anonymous function (you can use the built-in function 'det') and use 'fzero' to find two eigenvalues σ_1 , and σ_2 (which would be output of the function EigVal.m). You will need two initial guesses for 'fzero' to find both roots. Use the Min-max theorem to find those two initial guesses. The theorem states that all eigenvalues of a square matrix **A** will fall in the range of the scalar $a = \mathbf{u} \bullet (\mathbf{Au})$ when the unit vector $\|\mathbf{u}\| = 1$ covers the space. Find this range for the matrix **S** by scanning the 2-D plane (i.e. let $\|\mathbf{u}\|$ span 0 to 180 degree. Should you include 180 to 360 degree as well?). Use the lower and upper bounds of this range as initial guesses for 'fzero'.

3- Write a function (named EigVec.m) that finds the eigenvectors $\mathbf{x}_1$, and $\mathbf{x}_2$ of **S** from the following system of linear equations

$$(\mathbf{S} - \sigma\mathbf{I})\mathbf{x} = \mathbf{0}$$

Note that the bold **0** in the right is a 2-by-1 zero vector (zeros(2,1)). The arguments for this function will be **S** and one eigenvalue, and the output is the associated eigenvector **x**. Hint: Inside the function, set the first component x(1) equal to one and solve for the other component (Calculate $\det(\mathbf{S} - \sigma\mathbf{I})$. It is zero for both σ_1 , and σ_2.

What does it imply? Why can't both elements of **x** be determined uniquely? Do we need them to be determined uniquely anyway?). You can work-out the problem on the paper and include the analytical solutions inside the function file. Note that both equations will give the same answer for x(2) after setting x(1)=1. Scale both components at the end such that the magnitude of **x** is equal to one.

4- Use the built-in function 'eig' in MATLAB to verify your results in Parts 2 and 3. Are there any differences?

5- Find the angle θ_1 and θ_2 that $\mathbf{x}_1$ and $\mathbf{x}_2$ make with the x-axis ($\theta = 0^\circ$), respectively. Write an algorithm, or use the correct built-in function of MATLAB, to calculate these angles between 0 and π. (What is the value $|\theta_1 - \theta_2|$? What does this say about the principal directions of **S** (i.e., $\mathbf{x}_1$ and $\mathbf{x}_2$)?).

6- Write a function (Rotation.m) that uses the 2-D rotation matrix with angle θ and transforms **S** according to the following transformation rule

$$\mathbf{R} = \begin{pmatrix} \cos(\theta) & -\sin(\theta) \\ \sin(\theta) & \cos(\theta) \end{pmatrix} \qquad \mathbf{S}' = \mathbf{R}\ \mathbf{S}\mathbf{R}^{\mathrm{T}}$$

The inputs of this function will be original **S**, and θ. Store separate

points $(\mathbf{S}'_{11},\mathbf{S}'_{12})$ and $(\mathbf{S}'_{22},-\mathbf{S}'_{12})$ resulting from all rotations from $\theta = 0$ to 90 degree. (For convenience in plotting, I suggest you store the 3 independent elements (i.e., $\mathbf{S}'_{11}$, $\mathbf{S}'_{22}$, and $\mathbf{S}'_{12}$) into a 2-D array with three rows. You will have as many columns as the number of θ). Use increments of one degree.

7- Make a 2-by-2 matrix **V** with two eigenvectors. The first and second rows should be $\mathbf{x}_1$ and $\mathbf{x}_2$, respectively. Find $\mathbf{S}' = \mathbf{V}\,\mathbf{S}\,\mathbf{V}^{\mathrm{T}}$. Is $\mathbf{S}'$ a diagonal matrix? Are its diagonal components equal to eigenvalues of **S**? Is **V** an orthogonal matrix? i.e. does it satisfy $\mathbf{V}^{-1} = \mathbf{V}^{\mathrm{T}}$? Read about the diagonalization of a symmetric matrix. This is what you just did.

8- Develop a GUI that asks for two inputs: (i) three independent components of **S** in a 1-by-3 table, and (ii) an angle between 0 and 90 degree (θ_0). The three elements will be three upper-triangular elements of **S** where the first two elements are diagonal elements $\mathbf{S}_{11}$ and $\mathbf{S}_{22}$, and the next element is the remaining upper triangular element. Construct the complete matrix **S** using the symmetry of the matrix. The GUI should show the following items

- Plots the curve resulting from the points $(\mathbf{S}'_{11},\mathbf{S}'_{12})$ and $(\mathbf{S}'_{22},-\mathbf{S}'_{12})$ stored in the vector mentioned in Part 6. You will get a circle if you calculate/plot this correctly. Use blue color for this curve. You may fill this circle with some color. Also, you need to re-locate your axes to origin (look up the command). Note that by

changing θ between 0 to 90 in the rotation matrix, we are able to cover the entire 2-D plane. This means that an increment of '1' for θ in the rotation matrix corresponds to an increment of '2' in the displayed 2-D plane.

- A line connecting two original points $(\mathbf{S}_{11},\mathbf{S}_{12})$ and $(\mathbf{S}_{22},-\mathbf{S}_{12})$ with markers showing these two points. This line will be a diameter of the circle. Use black color.
- Two circular markers denoting the two points $(\mathbf{S}'_{11},\mathbf{S}'_{12})$ and $(\mathbf{S}'_{22},-\mathbf{S}'_{12})$ resulting from the rotation θ_0 with a line connecting them. Use red color with a dashed style.
- Two circular markers denoting two eigenvalues in the x-axis with y=0. Use green color.
- Two vectors originating from the origin of the circle with angles $2\theta_1$ and $2\theta_2$. Use the built-in function 'quiver' to plot these two vectors. Use magenta for the first vector and cyan for the second vector.

In addition, the GUI should display the following information in the table form

- The components for the rotated matrix under θ_0
- Eigenvalues of $\mathbf{S}$
- Eigenvectors of $\mathbf{S}$. Display Eigenvectors as a 2-by-2 matrix where the first and second columns are the first and second matrix.
- The angles θ_1 and θ_2

This is what your GUI should look like for the values $\mathbf{S}(1,1) = 5,\ \mathbf{S}(2,2) = 3,\ \mathbf{S}(1,2) = -2$, and $\theta_0 = 20^\circ$. However, you should submit the results for the values $\mathbf{S}(1,1) = 7,\ \mathbf{S}(2,2) = 4,\ \mathbf{S}(1,2) = -3$, and $\theta_0 = 30^\circ$.