Taben Malik

tm27848

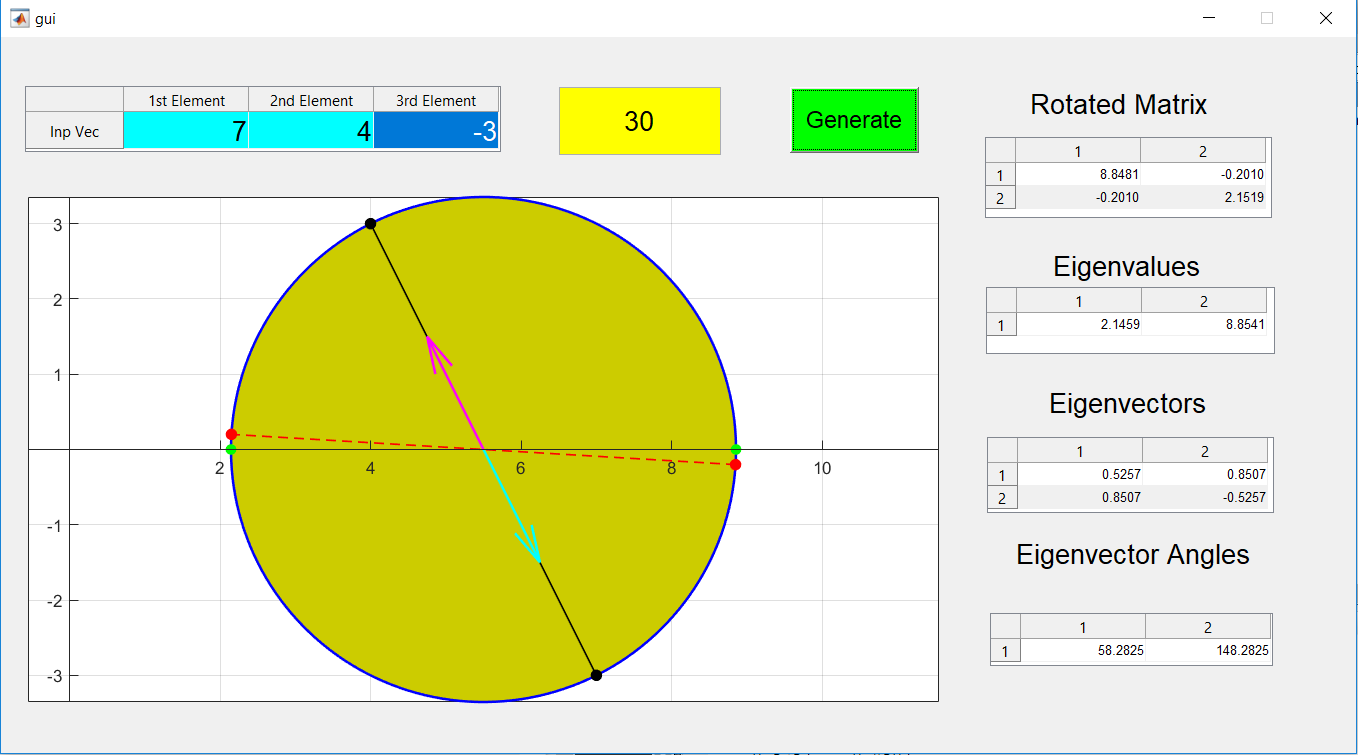
ASE301

2017/05/05

ASE 301 GUI Project

This project was rather simple and took my 24 hours to complete. The GUIDE interface was very similar to the Visual Studios GUI creator for the C# language. The only challenge of this project was finding where I used the function sin() instead of sind(). That bug evaded me for a while. Other than that, here is my results:

GUI Interface with requested input:



The answers to the requested questions are outputted to the command window. Here is the output of the above input:

Checking if eigenvalues are correct.

eig 2.145898 ?= calculated 2.145898

There is no difference.

The correct eigenvector for 2.145898 is:

-0.5257

-0.8507

The calculated eigenvector is:

0.5257

0.8507

eig 8.854102 ?= calculated 8.854102

There is no difference.

The correct eigenvector for 8.854102 is:

-0.8507

0.5257

The calculated eigenvector is:

0.8507

-0.5257

Eigenvectors can be the negative of itself without changing the meaning of the eigenvector

Sp:

2.1459 0.0000

0.0000 8.8541

The matrix Sp is not a diagonal matrix...

The diagonal components of Sp are not equal to the eigenvalues

V:

0.5257 0.8507

0.8507 -0.5257

V is an orthogonal matrix.

>> gui

Checking if eigenvalues are correct.

eig 2.145898 ?= calculated 2.145898

There is no difference.

The correct eigenvector for 2.145898 is:

-0.5257

-0.8507

The calculated eigenvector is:

0.5257

0.8507

eig 8.854102 ?= calculated 8.854102

There is no difference.

The correct eigenvector for 8.854102 is:

-0.8507

0.5257

The calculated eigenvector is:

0.8507

-0.5257

Eigenvectors can be the negative of itself without changing the meaning of the eigenvector

Sp:

2.1459 0.0000

0.0000 8.8541

Error using format

Unrecognized command option float.

Error in main\_func (line 69)

format float

Error in gui>Generate\_Callback (line 87)

[values, vectors, angles, points] = main\_func(S);

Error in gui\_mainfcn (line 95)

feval(varargin{:});

Error in gui (line 42)

gui\_mainfcn(gui\_State, varargin{:});

Error in

matlab.graphics.internal.figfile.FigFile/read>@(hObject,eventdata)gui('Generate\_Callback',hObject,eventdata,guidata(hObject))

Error while evaluating UIControl Callback

Checking if eigenvalues are correct.

eig 2.145898 ?= calculated 2.145898

There is no difference.

The correct eigenvector for 2.145898 is:

-0.5257

-0.8507

The calculated eigenvector is:

0.5257

0.8507

eig 8.854102 ?= calculated 8.854102

There is no difference.

The correct eigenvector for 8.854102 is:

-0.8507

0.5257

The calculated eigenvector is:

0.8507

-0.5257

Eigenvectors can be the negative of itself without changing the meaning of the eigenvector

Sp:

2.1459 0.0000

0.0000 8.8541

The matrix Sp is not a diagonal matrix...

The diagonal components of Sp are not equal to the eigenvalues

V:

0.5257 0.8507

0.8507 -0.5257

V is an orthogonal matrix.

Checking if eigenvalues are correct.

eig 2.145898 ?= calculated 2.145898

There is no difference.

The correct eigenvector for 2.145898 is:

-0.5257

-0.8507

The calculated eigenvector is:

0.5257

0.8507

eig 8.854102 ?= calculated 8.854102

There is no difference.

The correct eigenvector for 8.854102 is:

-0.8507

0.5257

The calculated eigenvector is:

0.8507

-0.5257

Eigenvectors can be the negative of itself without changing the meaning of the eigenvector

Sp:

2.1459 0.0000

0.0000 8.8541

The matrix Sp is a diagonal matrix!

The diagonal components of Sp are not equal to the eigenvalues

V:

0.5257 0.8507

0.8507 -0.5257

V is an orthogonal matrix.

Checking if eigenvalues are correct.

eig 2.145898 ?= calculated 2.145898

There is no difference.

The correct eigenvector for 2.145898 is:

-0.5257

-0.8507

The calculated eigenvector is:

0.5257

0.8507

eig 8.854102 ?= calculated 8.854102

There is no difference.

The correct eigenvector for 8.854102 is:

-0.8507

0.5257

The calculated eigenvector is:

0.8507

-0.5257

Eigenvectors can be the negative of itself without changing the meaning of the eigenvector

Sp:

2.1459 0.0000

0.0000 8.8541

The matrix Sp is a diagonal matrix!

The diagonal components of Sp are not equal to the eigenvalues

V:

0.5257 0.8507

0.8507 -0.5257

V is an orthogonal matrix.

Checking if eigenvalues are correct.

eig 2.145898 ?= calculated 2.145898

There is no difference.

The correct eigenvector for 2.145898 is:

-0.5257

-0.8507

The calculated eigenvector is:

0.5257

0.8507

eig 8.854102 ?= calculated 8.854102

There is no difference.

The correct eigenvector for 8.854102 is:

-0.8507

0.5257

The calculated eigenvector is:

0.8507

-0.5257

Eigenvectors can be the negative of itself without changing the meaning of the eigenvector

Sp:

2.1459 0.0000

0.0000 8.8541

The matrix Sp is a diagonal matrix!

The diagonal components of Sp are equal to the eigenvalues

V:

0.5257 0.8507

0.8507 -0.5257

V is an orthogonal matrix.

Checking if eigenvalues are correct.

eig 2.145898 ?= calculated 2.145898

There is no difference.

The correct eigenvector for 2.145898 is:

-0.5257

-0.8507

The calculated eigenvector is:

0.5257

0.8507

eig 8.854102 ?= calculated 8.854102

There is no difference.

The correct eigenvector for 8.854102 is:

-0.8507

0.5257

The calculated eigenvector is:

0.8507

-0.5257

Eigenvectors can be the negative of itself without changing the meaning of the eigenvector

Sp:

2.1459 0.0000

0.0000 8.8541

The matrix Sp is a diagonal matrix!

The diagonal components of Sp are equal to the eigenvalues

V:

0.5257 0.8507

0.8507 -0.5257

V is an orthogonal matrix.

>>

As you can see I get the same eigenvalues as eig . The eigenvectors are different only by their sign. The approach for calculating eigenvectors given in the project guarantees that the first component will be positive. However, this doesn’t mean the eigenvectors are different. The sign of the eigenvector does not change the meaning of the eigenvector and still corresponds to the same eigenvalue. Therefore, my eigenvectors are essentially the same. Calculating S prime from Sp = V \* (S\*transpose(V)) shows that Sp is a diagonal matrix. To programmatically determine this, however, required a threshold to eliminate some floating point errors since comparing them directly they could be off by 1e-14.

Now for the real fluff of the project. Here is the code separated by every file:

gui.m:

function varargout = gui(varargin)

% GUI MATLAB code for gui.fig

% GUI, by itself, creates a new GUI or raises the existing

% singleton\*.

%

% H = GUI returns the handle to a new GUI or the handle to

% the existing singleton\*.

%

% GUI('CALLBACK',hObject,eventData,handles,...) calls the local

% function named CALLBACK in GUI.M with the given input arguments.

%

% GUI('Property','Value',...) creates a new GUI or raises the

% existing singleton\*. Starting from the left, property value pairs are

% applied to the GUI before gui\_OpeningFcn gets called. An

% unrecognized property name or invalid value makes property application

% stop. All inputs are passed to gui\_OpeningFcn via varargin.

%

% \*See GUI Options on GUIDE's Tools menu. Choose "GUI allows only one

% instance to run (singleton)".

%

% See also: GUIDE, GUIDATA, GUIHANDLES

% Edit the above text to modify the response to help gui

% Last Modified by GUIDE v2.5 30-Apr-2017 21:04:32

% Begin initialization code - DO NOT EDIT

gui\_Singleton = 1;

gui\_State = struct('gui\_Name', mfilename, ...

'gui\_Singleton', gui\_Singleton, ...

'gui\_OpeningFcn', @gui\_OpeningFcn, ...

'gui\_OutputFcn', @gui\_OutputFcn, ...

'gui\_LayoutFcn', [] , ...

'gui\_Callback', []);

if nargin && ischar(varargin{1})

gui\_State.gui\_Callback = str2func(varargin{1});

end

if nargout

[varargout{1:nargout}] = gui\_mainfcn(gui\_State, varargin{:});

else

gui\_mainfcn(gui\_State, varargin{:});

end

% End initialization code - DO NOT EDIT

% --- Executes just before gui is made visible.

function gui\_OpeningFcn(hObject, eventdata, handles, varargin)

% This function has no output args, see OutputFcn.

% hObject handle to figure

% eventdata reserved - to be defined in a future version of MATLAB

% handles structure with handles and user data (see GUIDATA)

% varargin command line arguments to gui (see VARARGIN)

% Choose default command line output for gui

handles.output = hObject;

% Update handles structure

guidata(hObject, handles);

% UIWAIT makes gui wait for user response (see UIRESUME)

% uiwait(handles.figure1);

% --- Outputs from this function are returned to the command line.

function varargout = gui\_OutputFcn(hObject, eventdata, handles)

% varargout cell array for returning output args (see VARARGOUT);

% hObject handle to figure

% eventdata reserved - to be defined in a future version of MATLAB

% handles structure with handles and user data (see GUIDATA)

% Get default command line output from handles structure

varargout{1} = handles.output;

% --- Executes on button press in Generate.

function Generate\_Callback(hObject, eventdata, handles)

% hObject handle to Generate (see GCBO)

% eventdata reserved - to be defined in a future version of MATLAB

% handles structure with handles and user data (see GUIDATA)

% Get data

data = get(handles.Inp\_Vec, 'Data');

S = [data{1}, data{3}; data{3}, data{2}];

angle = str2num(get(handles.Angle, 'String'));

rm = Rotation(S, angle);

[values, vectors, angles, points] = main\_func(S);

% Set data

set(handles.Eigenvalues, 'data', values);

set(handles.Eigenvectors, 'data', vectors);

set(handles.Eigenvector\_Angles, 'data', angles);

set(handles.Rotated\_Matrix, 'data', rm);

% Graph data

graph = handles.Graph;

axes(graph);

cla reset;

% Graph circle

x = points(1,:);

y = points(2,:);

area(x, y, 'FaceColor', [.8 .8 0], 'EdgeColor', 'none');

hold on

plot(x, y, 'Color', 'b', 'LineWidth', 1.5);

% Graph original matrix

hold on

plot([S(1,1), S(2,2)], [S(1,2),-S(1,2)], '-ok', 'MarkerFaceColor', 'k', 'LineWidth',1);

% Graph rotated matrix

hold on

plot([rm(1,1), rm(2,2)], [rm(1,2),-rm(1,2)], '--or', 'MarkerFaceColor', 'r', 'LineWidth',1);

% Graph eigenvalues

hold on

plot(values(1),0, 'og', 'MarkerFaceColor', 'g');

hold on

plot(values(2),0, 'og', 'MarkerFaceColor', 'g');

% Graph vectors from angles

hold on;

scale = (((S(1,1) - S(2,2))^2 + (S(1,2) + S(1,2))^2)^.5)/4; % fourth of the diameter

centerX = (S(1,1) + S(2,2)) / 2;

centerY = (S(1,2) - S(1,2)) / 2;

u = cosd(2\*angles(1));

v = sind(2\*angles(1));

quiver(centerX,centerY,u,v, scale,'LineWidth', 1.5, 'Color', 'magenta', 'MaxHeadSize', 1);

hold on;

u = cosd(2\*angles(2));

v = sind(2\*angles(2));

quiver(centerX,centerY,u,v, scale,'LineWidth', 1.5, 'Color', 'cyan', 'MaxHeadSize',1);

% Set graph options

graph.XAxisLocation = 'origin';

graph.YAxisLocation = 'origin';

axis square equal;

grid on;

% --- Executes during object creation, after setting all properties.

function Inp\_Vec\_CreateFcn(hObject, eventdata, handles)

% hObject handle to Inp\_Vec (see GCBO)

% eventdata reserved - to be defined in a future version of MATLAB

% handles empty - handles not created until after all CreateFcns called

set(hObject, 'Data', cell(1, 3));

function Angle\_Callback(hObject, eventdata, handles)

% hObject handle to Angle (see GCBO)

% eventdata reserved - to be defined in a future version of MATLAB

% handles structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of Angle as text

% str2double(get(hObject,'String')) returns contents of Angle as a double

% --- Executes during object creation, after setting all properties.

function Angle\_CreateFcn(hObject, eventdata, handles)

% hObject handle to Angle (see GCBO)

% eventdata reserved - to be defined in a future version of MATLAB

% handles empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.

% See ISPC and COMPUTER.

if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))

set(hObject,'BackgroundColor','white');

end

% --- Executes during object creation, after setting all properties.

function Rotated\_Matrix\_CreateFcn(hObject, eventdata, handles)

% hObject handle to Rotated\_Matrix (see GCBO)

% eventdata reserved - to be defined in a future version of MATLAB

% handles empty - handles not created until after all CreateFcns called

set(hObject, 'Data', cell(2));

% --- Executes during object creation, after setting all properties.

function Eigenvalues\_CreateFcn(hObject, eventdata, handles)

% hObject handle to Eigenvalues (see GCBO)

% eventdata reserved - to be defined in a future version of MATLAB

% handles empty - handles not created until after all CreateFcns called

set(hObject, 'Data', cell(1, 2));

% --- Executes during object creation, after setting all properties.

function Eigenvectors\_CreateFcn(hObject, eventdata, handles)

% hObject handle to Eigenvectors (see GCBO)

% eventdata reserved - to be defined in a future version of MATLAB

% handles empty - handles not created until after all CreateFcns called

set(hObject, 'Data', cell(2));

% --- Executes during object creation, after setting all properties.

function Eigenvector\_Angles\_CreateFcn(hObject, eventdata, handles)

% hObject handle to Eigenvector\_Angles (see GCBO)

% eventdata reserved - to be defined in a future version of MATLAB

% handles empty - handles not created until after all CreateFcns called

set(hObject, 'Data', cell(1, 2));

main\_func.m:

function [EigValues, EigVectors, EigVecAngles, CirclePoints] = main\_func(S)

% MAIN\_FUNC the main function for the project Gui

% returns:

% EigValues - an array of eigen values

% EigVec - an array of eigenvectors that correspond to the eigvalues

% EigVecAngles - the angles of the corresponding eigenvectors

% CirclePoints - an array of (x,y) coordinates that form a circle

[sigma1, sigma2] = EigVal(S); % Get eigvalues

eigV1 = EigVec(S, sigma1); % get eigvec1

eigV2 = EigVec(S, sigma2); % get eigvec2

ang1 = angleOfVector(eigV1); % get eigvec1 angle

ang2 = angleOfVector(eigV2); % get eigvec2 angle

% Create list of points of all possible rotations of the matrix

points = [];

for a = 0:1:90

Sp = Rotation(S, a);

point1 = [Sp(1,1); Sp(1,2)];

point2 = [Sp(2,2); -Sp(1,2)];

points = [points, point1, point2];

end;

EigValues = [sigma1, sigma2];

EigVectors = [eigV1, eigV2];

EigVecAngles = [ang1, ang2];

% Adjust order of points

CirclePoints = [points(:,1:2:end),points(:,2:2:end)];

% End of calculation portion

% Start of question answering

% Check eigenvalues and eigenvectors

correctEigValues = eig(S);

[correctEigVectors, D] = eig(S);

fprintf('Checking if eigenvalues are correct.\n');

fprintf('eig %f ?= calculated %f\n', correctEigValues(1), EigValues(1));

if correctEigValues(1) == EigValues(1)

fprintf('There is no difference.\n');

else

fprintf('There IS a difference!.\n');

end;

fprintf('\nThe correct eigenvector for %f is:\n',EigValues(1));

disp(correctEigVectors(:,1));

fprintf('The calculated eigenvector is:\n');

disp(EigVectors(:,1));

fprintf('eig %f ?= calculated %f\n', correctEigValues(2), EigValues(2));

if correctEigValues(2) == EigValues(2)

fprintf('There is no difference.\n');

else

fprintf('There IS a difference!.\n');

end;

fprintf('\nThe correct eigenvector for %f is:\n',EigValues(2));

disp(correctEigVectors(:,2));

fprintf('The calculated eigenvector is:\n');

disp(EigVectors(:,2));

fprintf('\nEigenvectors can be the negative of itself without changing the meaning of the eigenvector\n');

% Check S'

Sp = EigVectors \* (S \* EigVectors');

fprintf('\nSp:\n');

disp(Sp);

threshold = 1e-14; %Eliminates floating point errors

if abs(Sp(1,2) - Sp(2,1)) <= threshold

fprintf('The matrix Sp is a diagonal matrix!\n');

else

fprintf('The matrix Sp is not a diagonal matrix...\n');

end;

diagonal = diag(Sp);

if abs(diagonal' - EigValues) <= threshold

fprintf('The diagonal components of Sp are equal to the eigenvalues\n');

else

fprintf('The diagonal components of Sp are not equal to the eigenvalues\n');

end;

fprintf('\nV:\n');

disp(EigVectors);

tf = all(abs(inv(EigVectors) - transpose(EigVectors))<=threshold);

if tf

fprintf('V is an orthogonal matrix.\n');

else

fprintf('V is not an orthogonal matrix.\n');

end;

end

EigVal.m:

function [sigma1, sigma2] = EigVal(S)

% EIGVAL returns the eigenvalues of the given symmetrix matrix

I = eye(2);

eigValEquation = @(lambda) det(S - (lambda \* I));

% Use Min-Max theorem to find guesses

u = [0, 1];

a = dot(u, S \* u');

min = a;

max = min;

% Scans the 2D plane for the min and max

for theta = 0:1:180

u = [cosd(theta), sind(theta)];

a = dot(u, S \* u');

if a < min

min = a;

elseif a > max

max = a;

end;

end;

% Use the min and max to find the eigenvalues

sigma1 = fzero(eigValEquation, min);

sigma2 = fzero(eigValEquation, max);

end

EigVec.m:

function eigVector = EigVec(S, sigma)

% EIGVEC returns the eigenvector of the given matrix and eigenvalue

syms x2;

% set up equation

x = [1;x2];

leftHand = (S - (sigma \* eye(2))) \* x;

% solve equation

solved = solve(leftHand(1) == 0, x2);

eigVector = [1;solved];

% normalize the vector (scale it to a length of 1)

eigVector = double((1 / norm(eigVector)) \* eigVector);

end

angleOfVector.m:

function angle = angleOfVector(v)

% ANGLEOFVECTOR returns the angle of the given vector

% determines the angle from the x axis between 0 and 180 degrees

x = [1, 0];

angle = acosd(dot(v, x)/(norm(v) \* norm(x)));

if v(2) < 0

angle = 180 - angle;

end;

end

Rotation.m:

function rotated = Rotation(S, angle)

% ROTATION returns the given matrix rotated by the given angle

R = [cosd(angle), -sind(angle); sind(angle), cosd(angle)];

rotated = R \* (S \* R');

end