MAT 343 Laboratory 4
Plotting and computer animation in MATLAB

In this laboratory session we will learn:

1. How to plot in MATLAB
2. The geometric properties of special types of matrices (rotations, dilations, reflections)
3. How to perform simple computer animations in MATLAB

Plotting in MATLAB

If **x** and **y** are vectors of the same size, the command `plot(x,y)` will produce a plot of all the (x_i, y_i) pairs, and each point will be connected to the next by a line segment. If the x-coordinates are taken close enough together, the graph should resemble a smooth curve. The command `plot(x,y,'o')` will mark the ordered pairs with `o`'s, but will not connect the points.

For example, to plot the function $f(x) = \dfrac{e^{x/10}\sin x}{x+1}$ on the interval $[0, 10]$, set

```
x = 0:0.2:10;
y = exp(x/10).*sin(x)./(x+1);
```

(note the `.` before the `*` and `/`, as these operations must be done component-wise).

The command `plot(x,y)` will generate the graph of the function. To compare the graph with that of $\sin x$, we could set `z = sin(x);` and use the command

```
plot(x,y,x,z)
```

to plot both curves at the same time.

It is also possible to do more sophisticated types of plots in MATLAB, including polar coordinates, contour plots, and three-dimensional surfaces.

Given the points (x_i, y_i), rather than creating the vectors **x** and **y**, an equivalent approach, which often is more convenient, is to create a $2 \times n$ matrix of data:

$$A = \begin{bmatrix} x_1 & x_2 & \dots & x_n \\ y_1 & y_2 & \dots & y_n \end{bmatrix}$$

then typing

```
plot(A(1,:),A(2,:))
```

In other words, the first row of A gives the x's and the second row the y's.

EXAMPLE 1

Consider the square whose vertices are $(0, 0), (1, 0), (1, 1), (0, 1)$.

(a) Enter the x and y coordinates of these points as MATLAB vectors by setting

```
    x = [0, 1, 1, 0, 0]
    y = [0, 0, 1, 1, 0]
```

The coordinates of the first points are repeated in the fourth column of `x` and `y` because we want to connect the fourth point back to the first. The command `plot(x,y)` draws this square in the graph window. The command `axis([-2,2,-2,2])` will rescale the axes so that the square does not take up the entire figure.

(b) Now let's store the entire figure in a matrix S by storing the x-coordinate in the first row and the y-coordinate in the second row:

```
S = [0, 1, 1, 0, 0; 0, 0, 1, 1, 0]
```

(if `x` and `y` have already been entered we can simply type `S = [x; y]`).
The square can then be plotted entering

```
 plot(S(1,:),S(2,:))
```

Again, set `axis([-2,2,-2,2])`, and add a grid by entering `grid on`.

Rotation Matrices in $\mathbb{R}^2$

Any 2-dimensional matrix of the form

$$Q = \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix}$$

is a rotation matrix. If $\mathbf{v}$ is a vector in $\mathbb{R}^2$, the product $Q\mathbf{v}$ is the vector $\mathbf{v}$ rotated θ radians in the counterclockwise direction around the origin $(0,0)$.
To rotate the vector in the clockwise direction we simply replace θ with $-\theta$, thus

$$Q^{-1} = \begin{bmatrix} \cos(-\theta) & -\sin(-\theta) \\ \sin(-\theta) & \cos(-\theta) \end{bmatrix}.$$

Since $\cos(-\theta) = \cos\theta$ and $\sin(-\theta) = -\sin(\theta)$, we have

$$Q^{-1} = \begin{bmatrix} \cos\theta & \sin\theta \\ -\sin\theta & \cos\theta \end{bmatrix} = Q^T.$$

EXAMPLE 2

We can rotate the square S from Example 1 by an angle of $\pi/4$ in the counterclockwise direction by multiplying the matrix S by

$$Q = \begin{bmatrix} \cos(\pi/4) & -\sin(\pi/4) \\ \sin(\pi/4) & \cos(\pi/4) \end{bmatrix}$$

The following code implements the rotation and the resulting picture is displayed in Figure 1.

```
clear all;   % clear all variables
clf;    % clear all settings for the plot
S=[0,1,1,0,0;0,0,1,1,0];
plot(S(1,:),S(2,:),'linewidth',2);  % plot the square
hold on;
theta =pi/4; % define the angle theta
Q=[cos(theta),-sin(theta);sin(theta),cos(theta)]; % rotation matrix
QS=Q*S; % rotate the square
plot(QS(1,:),QS(2,:),'-r','linewidth',2);   % plot the rotated square
title('Example of Rotation');  % add a title
legend('original square','rotated square') % add a legend
axis equal; axis([-1,2,-1,2]); % set the window
grid on; % add a grid
hold off
```

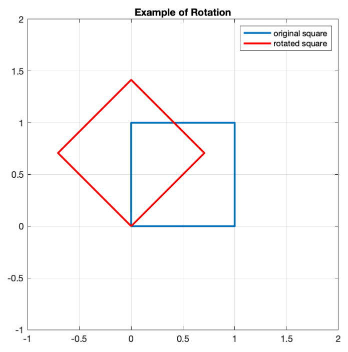

Figure 1: Original square and rotated square

Remarks:

- The `hold on` command concatenates all subsequent plot commands on the same graph. Type `hold off` to start a new plot.
- The `'-r'` plots the figure in red.
- `grid on` adds a grid to the axis
- `axis equal` sets the same scale on the x and y axes.
- `'linewidth', 2` sets the width of the line to 2 points = 2/72 inch (the default is 0.5 points).

Dilation and Contraction

Consider the matrix

$$D = \begin{bmatrix} \alpha_x & 0 \\ 0 & \alpha_y \end{bmatrix}.$$

If we multiply this matrix by any vector $\mathbf{v} = \begin{bmatrix} x & y \end{bmatrix}^T$ we obtain the vector $\begin{bmatrix} \alpha_x x & \alpha_y y \end{bmatrix}^T$ which scales the vector by a factor α_x in the x-direction and by a factor α_y in the y-direction. The scalings are a dilation or contraction, depending on whether α_x and α_y are greater or smaller than one.

EXAMPLE 3

Consider the case where the dilation factors are $\alpha_x = 2$ and $\alpha_y = 3$. We can plot the original square S together with the dilated version by executing the following commands. The output is displayed in Figure 2.

```
clf
S=[0,1,1,0,0;0,0,1,1,0];
plot(S(1,:),S(2,:),'linewidth',2)
hold on
D = [2, 0; 0, 3]; % dilation matrix
DS = D*S;
plot(DS(1,:),DS(2,:),'-r','linewidth',2)
title('Example of Dilation')
legend('original square','dilated square','location','southeast')
grid on
axis equal, axis([-1,4,-1,4]) % adjust the axis and the window
hold off
```

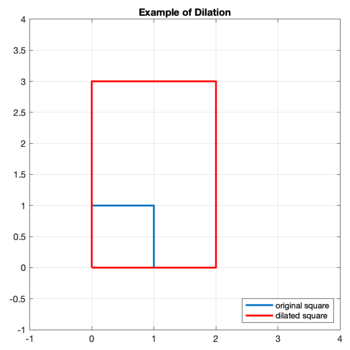

Figure 2: Original square and dilated square

Shear Matrices

A matrix of the form $T = \begin{bmatrix} 1 & t_x \\ 0 & 1 \end{bmatrix}$ is a horizontal shear transformation. When we apply this transformation, the y coordinates are unaffected, but the x coordinates are translated linearly with y.

Similarly, a matrix of the form $T = \begin{bmatrix} 1 & 0 \\ t_y & 1 \end{bmatrix}$ is a vertical shear transformation: the x coordinates are unaffected, but the y coordinates are translated linearly with x.

EXAMPLE 4

We will apply a horizontal shear transformation of 2 units. The matrix that accomplishes that is $T = \begin{bmatrix} 1 & 2 \\ 0 & 1 \end{bmatrix}$. We can plot the original square together with the sheared version by executing the following commands. The output is displayed in Figure 3.

```
clf
S=[0,1,1,0,0;0,0,1,1,0];
plot(S(1,:),S(2,:),'linewidth',2)
hold on
T=[1,2;0,1];     % shear transformation matrix
TS=T*S;
plot(TS(1,:),TS(2,:),'-r','linewidth',2);
title('Example of horizontal shear')
legend('original square','sheared square','location','southeast')
axis equal,axis([-1,4,-1,4]); grid on   %adjust the axis and the window
hold off
```

Composition of Transformations

One can perform a composition of two (or more) transformations by multiplying by the corresponding matrices in the appropriate order.

EXAMPLE 5

We can see the effect of first rotating the square 45 degrees counterclockwise and then applying the shear from EXAMPLE 3, by plotting the product `T*Q*S` . Note the order of the matrices; here we first apply the rotation (Q is the matrix that multiplies S first) and then we apply the shear. The

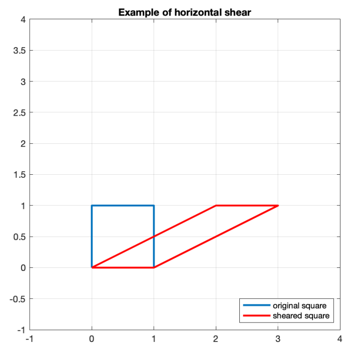

Figure 3: Original square and sheared square

output of the following code is displayed in Figure 4.

```
clf
S=[0,1,1,0,0;0,0,1,1,0];
plot(S(1,:),S(2,:),'linewidth',2)
hold on
theta=pi/4;  % define the angle
Q=[cos(theta), -sin(theta); sin(theta), cos(theta)];
T=[1,2;0,1];     % shear transformation matrix
TQS=T*Q*S;
plot(TQS(1,:),TQS(2,:),'-r','linewidth',2);
title('Example of rotation and shear')
legend('original square','modified square','location','southeast')
axis equal, axis([-1,4,-1,4]); grid on   % adjust the axis and the window
hold off
```

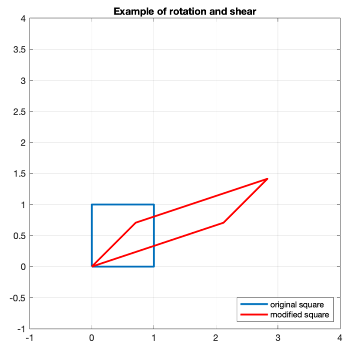

Figure 4: Original square and square first rotated and then sheared

Animation

EXAMPLE 6

The goal of this example is to draw the square in its original size, then cause it to disappear, and finally to redraw it as it looks after it is dilated by a factor of 9/8. If this operation is repeated ten times in succession, the square will appear to be expanding.
To do this we need to first plot the square and store its handle in `p`:

```
p = plot(S(1,:),S(2,:))
```

After we transform the matrix by multiplying by the appropriate transformation matrix, we can erase the original figure and draw the new figure with the command

```
set(p,'xdata',S(1,:),'ydata',S(2,:));
```

Since we are repeating the same operation ten times, we will use a `for` loop. After we have gone through this loop, we create another one that contracts the matrix by a factor of 8/9, thereby returning it to its original size.
The following are the commands that produce the effects. Enter them in MATLAB (you need not enter the comments that follow the % signs). The `pause(0.1)` command is used to adjust the speed of the plots so that we can see the graphs before they are erased. You might need to change the 0.1 to suit the speed of your machine.

```
clf                                 % clear all settings for the plot
S=[0,1,1,0,0;0,0,1,1,0];
D1 = 9/8*eye(2);                    % dilation matrix
p = plot(S(1,:),S(2,:));            % plot the square
axis([-1,4,-1,4])                   % set size of the graph
axis square, grid on                % make the display square
hold on                             % hold the current graph
for i = 1:10
    S = D1*S;                                % dilate the square
    set(p,'xdata',S(1,:),'ydata',S(2,:));    % erase original figure and plot
                                             % the transformed figure
    pause(0.1)          % adjust this pause rate to suit your computer.
end
D2 = 8/9*eye(2);        % contraction matrix
for i = 1:10
    S = D2*S;                                % contract the square
    set(p,'xdata',S(1,:),'ydata',S(2,:));    % erase original figure and plot
                                             % the transformed figure
    pause(0.1)          % adjust this pause rate to suit your computer.
end
hold off
```

EXERCISES

Instructions: For each of the following exercises, create a script M-file to store the MATLAB commands, then use the provided livescript template file to display the M-file together with the appropriate output. If the question requires written answers, include them in the livescript file in the appropriate location.

1. Determine the shearing transformation matrix that shears 1 units in the vertical direction. Plot the original square together with the sheared square. Use `axis([-1,3,-1,3]);`. Add a grid, a legend and a title (similarly to EXAMPLE 4). Include the M-file as well as the figure.

2. Consider the original square S. First apply the shear transformation from EXAMPLE 4 and then rotate the square 45° counterclockwise. Plot the resulting figure (together with the original square) and compare with the plot in Example 5. Are the results the same? Does the order of the transformations matter?
   Include the M-file as well as the figure and don't forget to answer all questions.
   Hint: Here you want to apply the shear first, then rotate, whereas in Example 5 the square is first rotated and then the shear is applied. To do this, `TQS = T*Q*S`, where Q is the rotation matrix and T is the shearing transformation matrix. So, if you want to first shear and then rotate, in what order will you multiply the matrices?

3. Adapt the procedure developed in Example 6 to rotate the square counterclockwise by increments of $\pi/9$ about the origin. Stop when the square is in its original location and then rotate it in the clockwise direction until the square is in its original location again. You may want to rescale the axis by using `axis([-2,2,-2,2])`.
Include the M-file. Do not include the figure.
Hint: Since you are performing a computation several times, you will want to use two `for` loops: one loop for the counterclockwise rotation and another one for the clockwise rotation. Think about how many times you will need to go through the loops, keeping in mind that you are rotating the square counterclockwise for a full circle by increments of $\pi/9$ and then rotating the square clockwise back again.

4. Adapt the procedure developed in Example 6 to show the square rotating in a counterclockwise direction about the origin by increments of $\pi/9$ for a total angle of 2π and expanding at the same time by a factor of 10/9, then stopping and rotating in the clockwise direction as it shrinks to its original size (with a contraction factor of 9/10). At the end of the program, the figure should have returned to its original size and original location.
You may want to rescale the axis to `axis([-8,8,-8,8]);`
Include the M-file. Do not include the figure.
Hint: Similarly to Exercise 3, you might want to use two `for` loops: one loop for the counterclockwise rotation + dilation, and another one for the clockwise rotation + contraction. Think about how many times you will need to go through the loops.

Homogeneous coordinates

Translations are not linear transformations and consequently cannot be accomplished via matrix multiplication using a 2×2 matrix. To circumvent this problem we use *homogeneous coordinates.* The homogeneous coordinates for a point $(x_1, x_2)^T$ are $(x_1, x_2, 1)^T$.
However, when the point represented by the homogeneous coordinates $(x_1, x_2, 1)^T$ is plotted, only the x_1 and x_2 are taken into consideration. The 1 is ignored. This allows us to produce the translated coordinates for a figure by matrix multiplication. Suppose we want to translate the vector c_1 units horizontally and c_2 units vertically. In homogeneous coordinates the translated vector is $(x_1 + c_1, x_2 + c_2, 1)^T$. We need a matrix M such that

$$M(x_1, x_2, 1)^T = (x_1 + c_1, x_2 + c_2, 1)^T$$

It is easy to verify that the matrix

$$M = \begin{bmatrix} 1 & 0 & c_1 \\ 0 & 1 & c_2 \\ 0 & 0 & 1 \end{bmatrix}$$

has the desired property.

EXAMPLE 7

We will apply a horizontal translation of 2 units and a vertical translation of -3 units. The matrix that accomplishes that is $M = \begin{bmatrix} 1 & 0 & 2 \\ 0 & 1 & -3 \\ 0 & 0 & 1 \end{bmatrix}$. To apply the transformation to the square S, we will need to convert the square to homogeneous coordinates. This is done by adding a row of ones to the matrix S so that the new matrix is $S = \begin{bmatrix} 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 \\ 1 & 1 & 1 & 1 & 1 \end{bmatrix}$. We can plot the original square together with the translated version by executing the following commands. The output is displayed in Figure 5.

```
clf
S=[0,1,1,0,0;0,0,1,1,0;1,1,1,1,1];    % square in homogeneous coordinates
M=[1,0,2;0,1,-3;0,0,1];     % translation matrix
MS=M*S;      % apply the translation to the square
plot(S(1,:),S(2,:),'k','linewidth',2);  % plot the original square in black
hold on
plot(MS(1,:),MS(2,:),'r','linewidth',2);  % plot the translated square in red
legend('original square','translated square','location','southwest');
axis equal, axis([-1,4,-4,4]), grid on    % adjust the axis
hold off
```

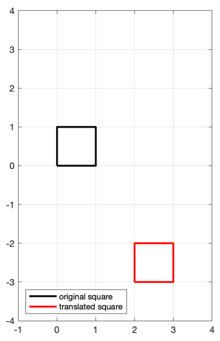

Figure 5: Original square and translated square

EXAMPLE 8

Consider the matrix S representing the original square in homogeneous coordinates. In the following M-file we translate the square horizontally using $c_1 = 0.4$ and $c_2 = 0$ for 40 times. We then translate the square vertically using $c_1 = 0$ and $c_2 = 0.4$ and 40 iterations.

```
clf
S=[0,1,1,0,0;0,0,1,1,0;1,1,1,1,1]; %  square in homogeneous coordinates
M1 = [1,0,0.4;0,1,0;0,0,1]; %  first translation matrix
M2 = [1,0,0;0,1,0.4;0,0,1]; %  the second translation matrix
p = plot(S(1,:),S(2,:)); % plot the original square
axis square, axis([-1,18,-1,18]), grid on
for i = 1:40
   S = M1*S; % compute the translated square
   set(p,'xdata',S(1,:),'ydata',S(2,:)); % plot the translated square
   pause(0.1)
end
for i = 1:40
   S=M2*S; % compute the translated square
   set(p,'xdata',S(1,:),'ydata',S(2,:)); % plot the translated square
   pause(0.1)
end
```

Reflection Matrices

The reflection matrix about the line L with direction the unit vector $\mathbf{u}$ is given by $R = 2\mathbf{u}\mathbf{u}^T - I$, where I is the identity matrix.

For instance, to obtain the reflection matrix about the line at 45 degrees we can take $\mathbf{u} = \left(\frac{\sqrt{2}}{2}, \frac{\sqrt{2}}{2}\right)^T$. Then $\mathbf{u}\mathbf{u}^T = \begin{bmatrix} 1/2 & 1/2 \\ 1/2 & 1/2 \end{bmatrix}$ and $R = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}$. (Note that this answer makes perfect sense, since the reflection of the vector $\mathbf{x} = (x_1, x_2)^T$ about the line at 45 degrees is the vector $(x_2, x_1)^T$). In homogeneous coordinates, the reflection matrix becomes:

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

The other types of basic transformations (rotations, dilations) can all be adapted to homogeneous coordinates by augmenting the matrix by the row $(0, 0, 1)$ and the column $(0, 0, 1)^T$. For instance, in homogeneous coordinates, the dilation matrix that dilates by a factor α becomes

$$D = \begin{bmatrix} \alpha & 0 & 0 \\ 0 & \alpha & 0 \\ 0 & 0 & 1 \end{bmatrix}.$$

EXERCISES

5. Consider the translated square `MS` in EXAMPLE 7. Apply to this square the reflection matrix that reflects about the line at 45 degrees. Plot the translated square in black and the reflected square in red. Use `axis([-4,4,-4,4]);` and plot the line $y = x$ using the command `plot([-4,4],[-4,4]);` (this command plots the line connecting the points $(-4, -4)$ and $(4, 4)$). Add a legend and a grid. You should reproduce Figure 6. Include the M-file and the figure.

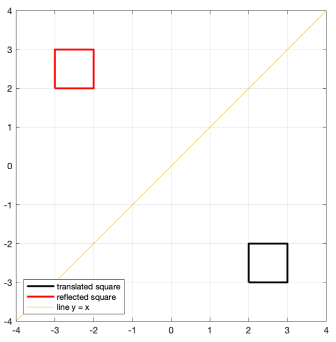

Figure 6: Output of Exercise 5

6. Modify the M-file in EXAMPLE 8 adding translations that bring the square to its original position using 40 iterations and a single additional `for` loop (for a total of three loops). Include the M-file. You do not need to include the figure.

Rotations about a point different than the origin

The rotation matrix $Q = \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix}$ performs a rotation about the origin, but what if we want to rotate about a point $P = (P_x, P_y)$ different than the origin? To accomplish this we need to

- translate the object so that P will coincide with the origin, i.e. translate $-P_x$ units horizontally and $-P_y$ units vertically
- rotate the object
- translate the object back, i.e. translate P_x units horizonally and P_y units vertically

To compose the three transformations above, we multiply the appropriate matrices:

$$\begin{bmatrix} 1 & 0 & P_x \\ 0 & 1 & P_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & -P_x \\ 0 & 1 & -P_y \\ 0 & 0 & 1 \end{bmatrix}$$

Note that the rotation matrix is written in homogeneous coordinates.

EXAMPLE 9

In this example we first translate the square horizontally 16 units using increments of 0.4. We then rotate the resulting square clockwise $\pi/2$ radians around the vertex $(17,0)$ using increments of $\pi/12$ radians.

```
clf
S=[0,1,1,0,0;0,0,1,1,0;1,1,1,1,1]; % square in homogeneous coordinates
M1 = [1,0,0.4;0,1,0;0,0,1]; % first translation matrix
theta = pi/12;  % define the angle theta
Q=[cos(theta),-sin(theta),0;sin(theta),cos(theta),0;0,0,1]; % rotation matrix about (0,0)
QP=[1,0,17;0,1,0;0,0,1]*Q'*[1,0,-17;0,1,0;0,0,1];   % rotation matrix about (17,0)
p = plot(S(1,:),S(2,:)); % plot the original square
axis equal, axis([-0.5,19,-2,5]), grid on
for i = 1:40
    S = M1*S; % compute the translated square
    set(p,'xdata',S(1,:),'ydata',S(2,:)); % plot the translated square
    pause(0.1)
end
for i = 1:6
    S=QP*S; % compute the rotated square
    set(p,'xdata',S(1,:),'ydata',S(2,:)); % plot the rotated square
    pause(0.1)
end
```

EXERCISES

7. Consider the square in EXAMPLE 9. The goal of this exercise is to bring back the square to its original position by first translating it horizontally to the left 16 units using 40 iterations, and then rotating it counterclockwise $\pi/2$ radians around the point $(1,0)$ using 6 iterations. This can be done by modifying the code in EXAMPLE 9 by adding two `for` loops. The first loop should translate the square while the second should rotate it around the point $(1,0)$. Note that the rotation is counterclockwise, while in EXAMPLE 9 it was clockwise. Include the M-file. You do not need to include the figure.