A MATLAB Exercise Book

Ludmila I. Kuncheva and Cameron C. Gray

Contents

Chapter 5

Plotting

5.1 Plotting Commands

Figures 5.1 and 5.2 detail the `plot` command and the appearance of MATLAB figures.

```
x = -5:10; % values of the argument
y = x.^2 - 20; % values of the function

figure
plot(y)

figure
plot(x,y)

figure
plot(x,y,'k.-')
```

Plots only the function versus the index: 1,2,3...

Plots the function versus the argument: -5,-4,-3...

Plots the function versus the argument with a black line and a black dot marker

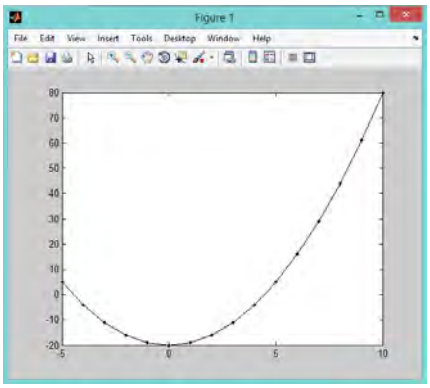


Figure 5.1: Plot command

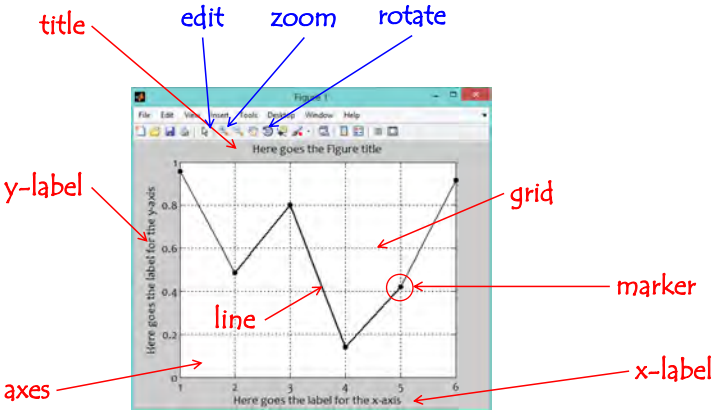
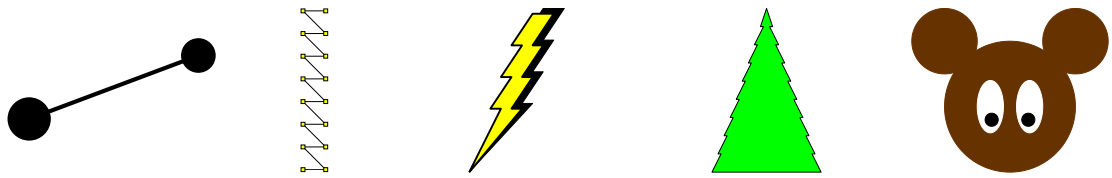


Figure 5.2: Figure with axes.

In addition, get familiar with the `fill` and `subplot` commands.

5.2 Examples

Reproduce the shapes and plots in Figure 5.3.



(a) Weights (b) Zigzag (c) Lightning bolt (d) Christmas tree (e) Bear's face

Figure 5.3: MATLAB plot examples.

Plot (a) can be constructed with one thick line and circle markers. To create the illusion of perspective, the ball which is closer to the viewer should be larger. This can be achieved by plotting a larger circular marker over the smaller one. The line width and the sizes of the two markers are determined by trying out different values until the figure meets the designer's approval. The code is shown below:

```
% Weights
figure, hold on, axis equal off % set up the figure and format the axes
plot([0 0.8],[0 0.3],'k.-','markersize',200,'linewidth',8)
plot(0,0,'k.','markersize',250) % plot a second, larger marker
```

The zigzag in plot (b) is based on a repeated pattern of x coordinates, e.g., [0 1 0 1 0 1 ...], while the y coordinate must increase as [1 1 2 2 3 3 ...]. Both patterns can be created using matrix manipulation as shown in the code below:

```
% Zigzag
figure, hold on, axis equal off % set up the figure and format the axes
x = repmat([0,1],1,8); y = [1:8;1:8];
plot(x,y(:),'ks-','markersize',8,'MarkerFaceColor','y')
```

For the lightning bolt in plot (c), the fill command should be used. The shadow has the same shape as the yellow bolt, and is displaced on both x and y. The bottom point of the shadow is 'stretched' to match the tip of the yellow spear. Note; that the shadow must be plotted first. The code is shown below;

```
% Lightning bolt
figure, hold on, axis equal off
x = [-2 3 2 4 3 5 4 6 4,2,3,1,2,0,1,-2]; % yellow X
y = [-1 1 1 2 2 3 3 4 4 3 3 2 2 1 1 -1]*3; % yellow Y
xsh = x + 1; xsh(1) = -2; xsh(end) = -2; % shadow X
ysh = y + 0.5; ysh(1) = -3; ysh(end) = -3; % shadow Y
fill(xsh,ysh,'k') % plot shadow first
fill(x,y,'y','edgecolor','k','linewidth',3) % plot yellow bolt
```

The Christmas tree in plot (d) is formed from two symmetrical parts. The x-values can be constructed for one of the parts and flipped for the other part. In the code below, both x and y are constructed initially as arrays with two rows. Then, with the help of the colon operator, the two rows are concatenated column-by-column to make the needed sequence of vertices. For example, if x has values [0 2 4 6] in the first row and [0 1 3 5] in the second row, the concatenated (and transposed) vector will be [0 0 2 1 4 3 6 5]. This gives the sawtooth pattern for the periphery of the tree. The code for the Christmas tree is as follows:

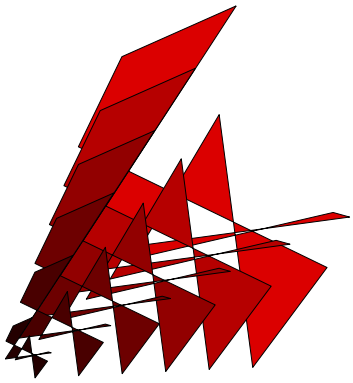
```
% Christmas tree
x = [0:2:18;0 1:2:17]; y = [20:-2:1;20:-2:1]*3;
x = [-fliplr(x(:)') x(:)']; y = [fliplr(y(:)') y(:)'];
figure, hold on, axis equal off, fill(x,y,'g')
```

The Bear's face in plot (e) would difficult to plot with markers of different sizes. It is better to use filled circles as shown in the code below:

```
% Bear's face
brown = [0.4 0.2 0]; % colour definition
figure, hold on, axis equal off
t = linspace(0,2*pi,100);
fill(sin(t),cos(t),brown,'EdgeColor',brown) % face
fill(sin(t)*0.5+1,cos(t)*0.5+1,brown,'EdgeColor',brown)
fill(sin(t)*0.5-1,cos(t)*0.5+1,brown,'EdgeColor',brown)
fill(sin(t)*0.2+0.3,cos(t)*0.4,'w','EdgeColor','w')
fill(sin(t)*0.2-0.3,cos(t)*0.4,'w','EdgeColor','w')
fill(sin(t)*0.1+0.28,cos(t)*0.1-0.2,'k','EdgeColor','k')
fill(sin(t)*0.1-0.28,cos(t)*0.1-0.2,'k','EdgeColor','k')
```

Random art can be created using filled polygons, not necessarily convex, with random vertices. For example, `fill(rand(10,1),rand(10,1),rand(1,3))` will create a random shape of joined line segments, where some of the closed spaces will be filled with a random colour. Figure 5.4 shows the output of the following piece of code:

```
x = rand(10,1); y = rand(10,1); % vertices of the polygon
figure, hold on, axis equal off
k = 6; % # of repetitions of the same shape
for i = 1:k
    fill(x*(k-i+1),y*(k-i+1),[k-i+1 0 0]/(k+1))
end
```



Note: To plot the shape at a different position, add the desired offset to the x and the y coordinates, respectively.

Figure 5.4: Repetitions of a shrinking random shape filled with progressively darkening red colour.

5.3 Exercises

1. Plot the six European flags as in Figure 5.5. The names of the countries should be displayed as well. All flags should be plotted in one figure. This task should be completed using the the `subplot` command rather than adjusting spacing between flags yourself.

Figure 5.5: Six European flags.

2. Write a function which will draw a circle in an open figure. The input arguments are x, y, r, c; x and y are the coordinates of the centre, r is the radius, and c is a three-element vector with the colour. Demonstrate your function by using it to plot 30 circles at random positions, with random radii, and with random colours, as in Figure 5.6.

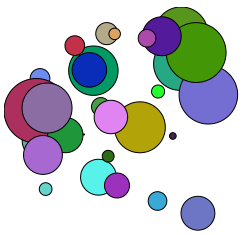


Figure 5.6: Illustration of the function for plotting circles.

3. Create an art figure by plotting 10 filled squares with jagged edges as shown in Figure 5.7. The fill colours should be random. The squares should be arranged approximately as in the example in the figure.

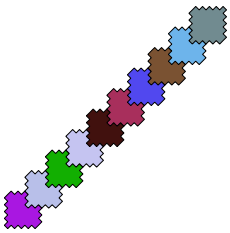


Figure 5.7: Jagged edge art example.

4. Plot a Random Art Square similar to the examples in Figure 5.8. There should be 20 random forms with random colours in the square. Note that some of the forms are not contained fully in the figure. Each form must have between 3 and 6 vertices. The number of vertices should be random.

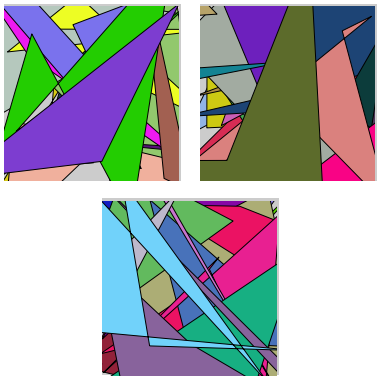


Figure 5.8: Three examples of a Random Art Square.

5. Create a loop to plot 10 triangles with random vertices in the unit square. Not every triangle should be plotted. Plot *only* triangles which are nearly equilateral. To do this, check whether the three edges differ by less than a chosen small constant, for example $e = 0.01$. The triangles should be filled with random colours. An example of the desired output is shown in Figure 5.9.

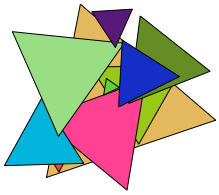


Figure 5.9: Random equilateral triangles.

6. Write a MATLAB function which takes an integer k as its input argument and plots k filled squares of random colours, nested as shown in Figure 5.10.

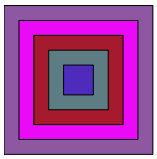


Figure 5.10: Five filled nested squares.

7. Using the function from the previous problem, reproduce Figure 5.11. The number of squares k varies from 3 to 12. All colours are random.

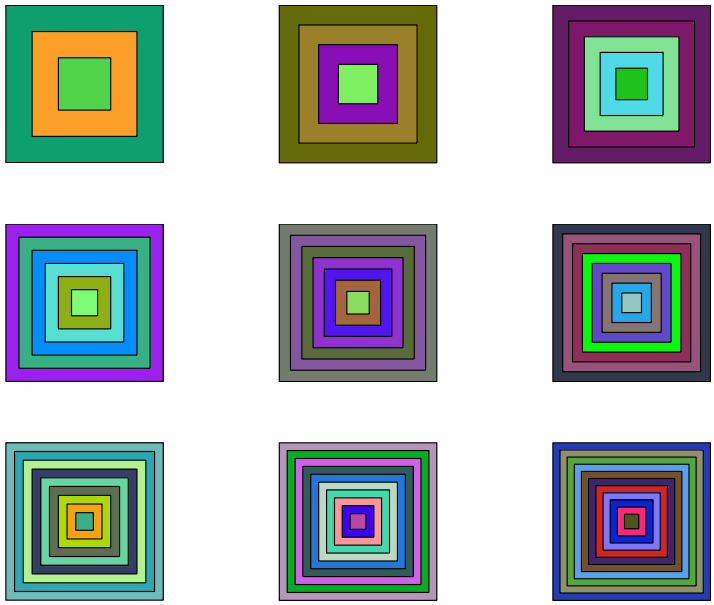


Figure 5.11: 3-12 filled nested squares.

8. Plot a shape consisting of four filled polygons. The polygons are mirror versions of one polygon with k random vertices, where k is a parameter. The figure should be symmetrical about the x *and* y axes. The polygons should touch in the middle point as shown in the examples in Figure 5.12 ($k = 10$). Try to accomplish this task using matrix operations, not geometric functions such as `fliplr`.

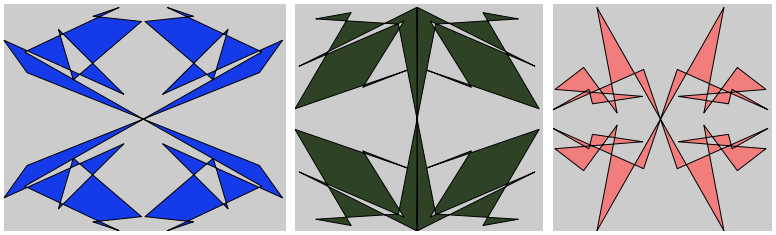


Figure 5.12: Shapes made of four filled polygons.

9. Write MATLAB code to produce a 4-part logo. Nine examples are shown in Figure 5.13. You can use your own design of the basic shape, or pick it at random. The top left part should be black and the bottom right should be white. The other two colours should be chosen randomly by your program. The four shapes should have a common edge on the x-axis and on the y-axis. The length of each of these edges should be at least half of the span of the shape on the respective axis.

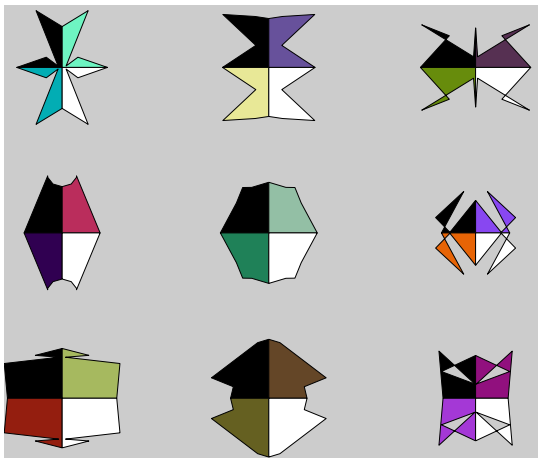
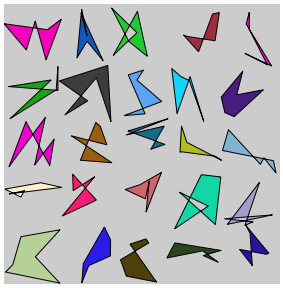


Figure 5.13: Examples of logos.

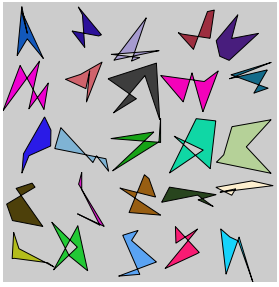
10. Random Shapes on a Grid

    (a) Create a set of 25 random shapes, each one having 6 random vertices and filled with a random colour. Plot the shapes on a 5-by-5 grid as shown in Figure 5.14 (a). This should not be achieved using the `subplot` command and can be accomplished with a single loop.

    (b) Make a figure with two subplots. The first subplot should contain the original shapes, and the second subplot should contain the same shapes, in a random order on the grid. An example is shown in Figure 5.14 (b).



(a) Original grid



(b) Shuffled grid

Figure 5.14: Grids of Shapes.

11. Birds, Butterflies or Flying Baba Yagas

    (a) Figure 5.15 shows a field with randomly distributed copies of a filled small shape or creature. The shape is random, but fixed for the figure, while the colours and the positions of the replicas are random. Write MATLAB code to produce a similar figure.

    (b) Subsequently, design a battle scene, where four 'armies' of creatures are distributed in four parts of the space as shown in Figure 5.16. The creatures from each army should have the same (random) shape and colour. The positions of the creatures within the regions are random too. (Note: The creatures are allowed to overlap near the borders)

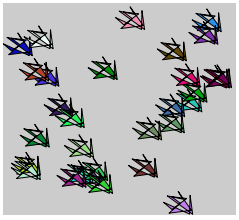


Figure 5.15: Small-shape art.

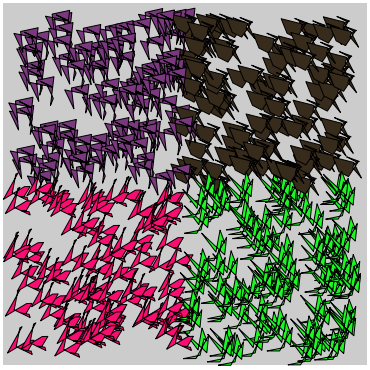
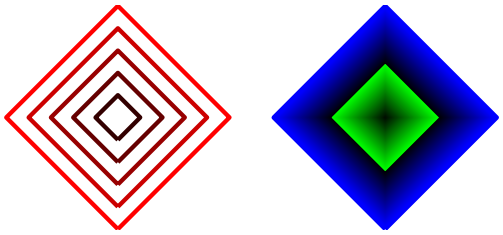


Figure 5.16: Four armies of creatures.

12. Diamonds in a Loop

    (a) Use a loop to create 5 diamonds as in Figure 5.17(a) (one diamond in each pass through the loop). The innermost diamond is black, and the outermost is red. Each diamond has its own fixed colour. The colours of the diamonds go gradually from black to red.

    (b) Subsequently, use one loop to create the pattern in Figure 5.17(b). The colour goes gradually from black to green followed by black to blue. The figure should be plotted as succession of diamonds.



(a) (b)

Figure 5.17: Diamonds.

13. Try to reproduce the row of Christmas trees in Figure 5.18.

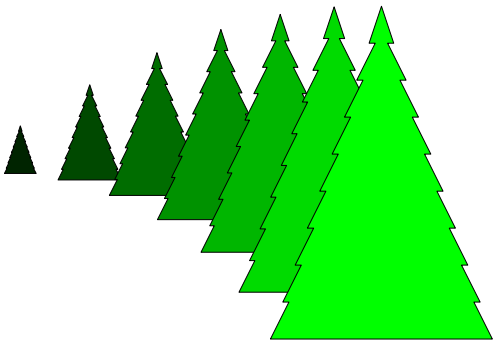


Figure 5.18: A row of Christmas trees.

14. Write a MATLAB function called 'dice'. The function should open a figure and display the given face of a regular six-sided die. The face 'number' is the only input argument, k. Examples of the desired output are shown in Figure 5.19. Notice the rounded corners. The orientation of faces 2, 3 and 6 does not matter as long as the white dots form the desired pattern.

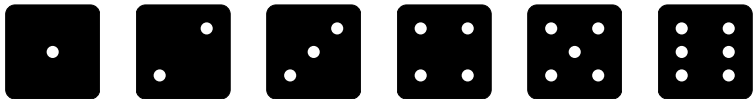


Figure 5.19: The 6 dice faces.

start CHALLENGE

Dice Face Golf

The challenge for this problem is to write the shortest possible code for the Dice function. The length of the code is the number of characters ignoring the white spaces and new lines. (In real competitions, the variable names of any length are counted as one character, and comments are not counted at all. In our competition both of these will count.)

end CHALLENGE

15. Hall of Fame

    Create a MATLAB function called `hall_of_fame`. The function should take 4 input arguments: the top 10 scores, the respective names, the new score and the new name. The output will be a figure with the top 10 names and scores. The new score should be displayed with a different colour. The font chosen and colours are not important as long as the new entry is a distinct colour. An example of the output is shown in Figure 5.20.

Figure 5.20: Example of the Hall of Fame.