MATLAB: File I/O, Functions, Plotting

Neil Rayala

November 30, 2018

1 File Input/Output

MATLAB's file I/O functionality is used to read and store data contained in the memory space.

```
f= fopen(name, ps) %Opens file with default permission (reading)
%Permission types can be found at https://tinyurl.com/y7rbw78v
fprintf(f,'10.5f \n', num) %Writes num with 5 digits after
                           %decimal and field width of 10
fscanf(filename, spec) %Command to read values from file
M = fscanf('d.txt', '%f') %Reads floating-point numbers into matrix
```

Binary files are computer-executable and include file types such as .bin. ASCII files include plain text file types such as .txt.

Algorithm 1 Binary and ASCII File Commands

```
save matlabday2.bin        %saves all data in memory space
save matlabday2.bin var1   %saves only var1 in file
save matlabday2.txt -ascii %saves data to ASCII-formatted file
load matlabday2.bin        %loads data
delete matlabday2.bin      %removes file
type matlabday2.bin        %displays file content
```

Use the sprintf command to store data passed to the file in a specific format.

Algorithm 2 Text Formatting

```
sprintf(spec, input) %formats input based on specification
sprintf("My name is %s %s", "Neil", ".") %My name is Neil.
sprintf(text) %formats escape character sequences (i.e. \n)
```

2 Functions

```
function func = matlab2(input)
```

A function definition always starts with the keyword 'function'. The variable func stores the value returned by the function and input is the local variable that contains the input value.

```
func = input − 1
end
```

The rest of the definition is stored in its own file which matches the function name (i.e. matlab2.mat). In this scenrio, the input variable is decremented and stored in the output variable. Make sure to always use the 'end' keyword to end the function.

```
z = matlab2(15)
```

The function is called by assigning an input value. The output value can be assigned to another variable, such as the 'z' in this case. Even if they have the same name, variables outside of the function will always be different from local variables inside the function.

3 Plotting

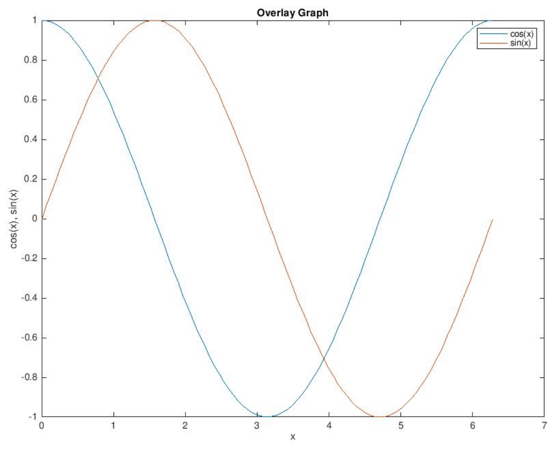

Figure 1: Overlay Plot

Algorithm 4 Trigonometric Plots

```
x = linspace(0, 2*pi, 100) %0 to 2pi with 100 points between
plot(x, cos(x))            %Makes a plot of function cos(x)
title('cos(x) Graph')      %Makes title
xlabel('x')                %labels x-axis
ylabel('cos(x)')           %labels y-axis

figure;                    %begins figure on new window
plot(x, cos(x))            %Makes a plot of function cos(x)
hold on;                   %Instructs not to overwrite plot

plot(x, sin(x))            %Overlays plot with function sin(x)
title('Overlay Graph')     %Makes title
xlabel('x')                %labels x-axis
ylabel('cos(x), sin(x)')   %labels y-axis
legend('cos(x)', 'sin(x)') %Adds legend
```

```
x = 0:0.1:2*pi;          %0 to 2pi with increment of 0.1
y = 0:0.1:2*pi;          %0 to 2pi with increment of 0.1
[X,Y] = meshgrid(x,y);   %stores 2D grid coordinates
Z = sin(X).*cos(Y);      %uses element-wise operation

figure;                  %surface plot
surf(X,Y,Z);
xlabel('x');
ylabel('y');
zlabel('sin(x)*cos(y)');
title('sin(x)*cos(y)');

figure;                  %contour plot with color bar
contourf(X,Y,Z);
colorbar;
xlabel('x');
ylabel('y');
zlabel('sin(x)*cos(y)');
title('sin(x)*cos(y)');

figure;                  %3D contour plot
contour3(X,Y,Z);
xlabel('x');
ylabel('y');
zlabel('sin(x)*cos(y)');
title('sin(x)*cos(y)');
```

Note: Use 'help plot' to develop other types of graphs if needed.

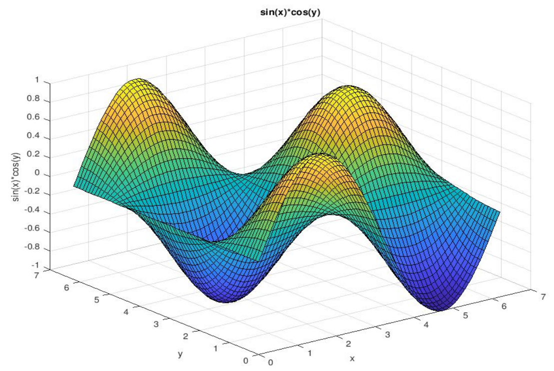

Figure 2: Surface Plot

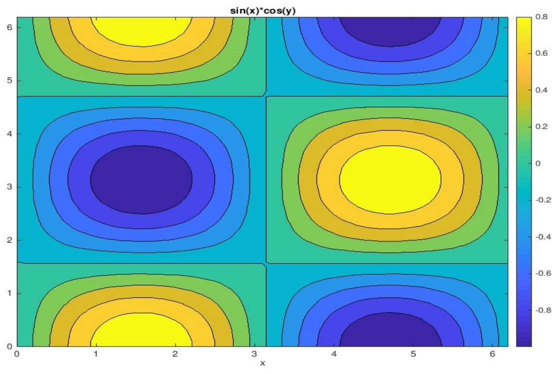

Figure 3: Contour Plot with Color Bar

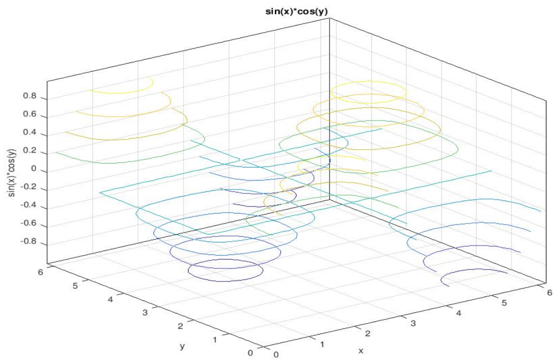

Figure 4: 3D Contour Plot