

MATLAB: Introduction

Part 2

Bruno Abreu Calfa

Last Update: January 29, 2014

Table of Contents

Outline

Contents

1	MATLAB Classes	1
2	Elements of Programming	2
3	Plotting	4
3.1	2-D Plotting	4
3.2	3-D Plotting	6

1 MATLAB Classes

ND Arrays

- MATLAB allows *multidimensional* arrays (n dimensions)
 - `>> nd1 = zeros(2,3,4) % 2-by-3-by-4 full of 0s`
 - `>> nd2 = ones(10,5,8,7) % 10-by-5-by-8-by-7 full of 1s`
 - `>> nd1(:,1,2) = 1:2 % Replaces column 1 of page 2 by [1, 2]`
 - `>> nd2(:, :, 5, 7) = rand(10,5) % Replaces rows and columns of page 5 and chapter 7 by random 10-by-5 matrix`

Cell and Structure Arrays

- Cell Arrays (`cell`): generic *containers* (store any type of data)
 - `>> cell1 = {'aaa', 1, rand(2,3)} % Use curly braces to retrieve/assign values`
 - `>> a = cell1(1) % 'a' is the first container (also a cell)`
 - `>> b = cell1{1} % 'b' is the first content (a char array)`
 - `>> cell1{:} % {:} generates a comma-separated list`
 - `>> [a,b,c] = cell1{:} % Assigns each content to a variable`
- Structure Arrays (`struct`): data types with *fields* and *values*
 - `>> methane.omega = .012; % Methane's acentric factor`
 - `>> methane.Tc = 190.6; % Its critical temperature, K`
 - `>> methane.Pc = 45.99; % Its critical pressure, bar`
 - `>> methane % Display methane fields and values`

2 Elements of Programming

Relational and Logical Operators

- Relational Operators:

>, <	greater than, smaller than
>=, <=	greater or equal than, smaller or equal than
==, ~=	equal to, not equal to

- Logical Operators:

&&, &	short-circuiting AND, element-wise AND
,	short-circuiting OR, element-wise OR
~	element-wise NOT

if-elseif-else Statements: Flow Control

- General form:

```
if expression1
    statements1
elseif expression2
    statements2
else
    statements3
end
```

- Example:

```
r = rand;
if (r < .3)
    r = r*2;
elseif (r >= .3 && ...
    r < .6)
    r = r*3;
else
    r = r*4;
end
```

switch-case Statements: Flow Control

- General form:

```
switch switch_expr
    case case_expr
        statement, ..., statement
    case {case_expr1, case_expr2, case_expr3, ...}
        statement, ..., statement
    otherwise
        statement, ..., statement
end
```

- Example:

```

method = 'Bilinear';
switch lower(method)
    case {'linear', 'bilinear'}
        disp('Method is linear')
    otherwise
        disp('Unknown method')
end

```

for Loop Statements

- General form:

```

for var = init:step:end
    statement
    statement
    ...
end

```

- Example:

```

a = zeros(10);
for i = 1:10
    for j = 1:10
        a(i,j) = 1/(i+j-1);
    end
end

```

while Loop Statements

- General form:

```

while expression
    statement
    statement
    ...
end

```

- Example:

```

x0 = .5;
x = x0 - tan(x0);
while (sqrt(x^2 - x0^2) > 1E-3)
    x0 = x;
    x = x0 - tan(x0);
end
sprintf('x_end = %g', x)

```

try-catch Statements: Error Handling

- General form:

```
try
    statement
    ...
catch [ME] % Optional
    statement
    ...
end
```

- Example:

```
try
    fid = fopen('a.txt', 'r');
    d_in = fread(fid);
catch EX
    disp('Exception caught!')
    EX
end
```

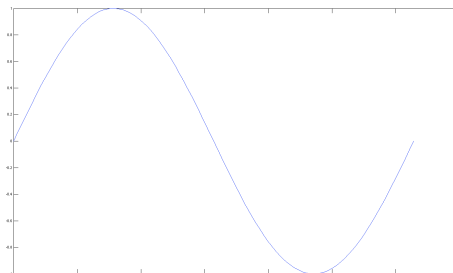
3 Plotting

3.1 2-D Plotting

2-D Plotting

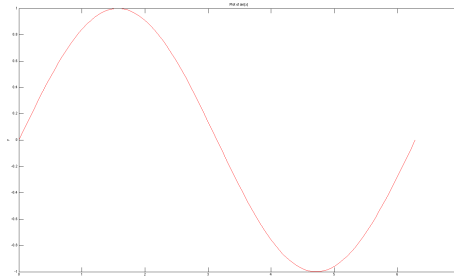
- The plotting commands in MATLAB work in a similar way: `command(data1,data2,...,['Prop1Name',Prop1Value,...])` where `data1,data2,...` are arrays of data to be graphed and `'Prop1Name', Prop1Value,...` are the plotting properties' names and respective values (optional)
- See MATLAB's Help for a description of all `lineseries` properties
- Some plotting commands: `plot`, `loglog`, `semilogx`, `semilogy`
- Basic example: `plot sin(x)` between $[0, 2\pi]$

```
x = linspace(0,2*pi);
y = sin(x);
figure
plot(x,y);
```



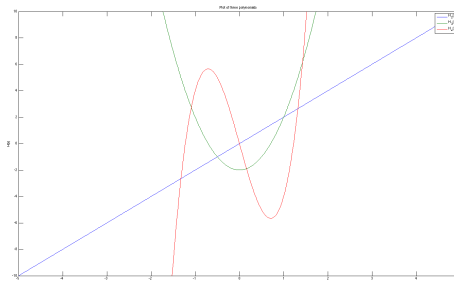
- Adding more information to the plot of $\sin(x)$ between $[0, 2\pi]$

```
x = linspace(0,2*pi);
y = sin(x);
figure
plot(x,y,'Color','red');
title('Plot of sin(x)');
xlabel('x');
ylabel('y');
```



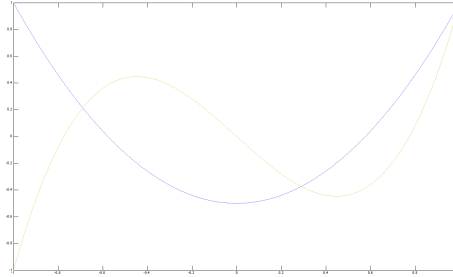
- Plotting multiple data on the same figure

```
x = linspace(-10,10,1000);
y = 2*x;
z = 4*x.^2 - 2;
w = 8*x.^3 - 12*x;
figure
plot(x,y,x,z,x,w);
title('Plot of three polynomials');
xlabel('x');
ylabel('H(x)');
ylim([-10 10]);
legend('H_2(x)', 'H_3(x)', 'H_4(x)');
```



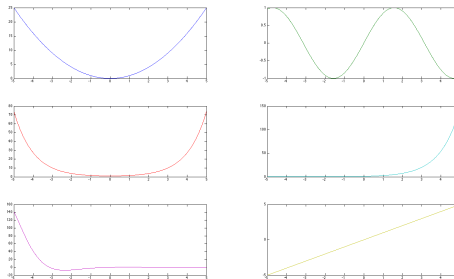
- Plotting multiple data on the same figure with hold on and hold off

```
x = linspace(-1,1,1000);
y = (3*x.^2 - 1)/2;
z = (5*x.^3 - 3*x)/2;
figure
plot(x,y,'Color',rand(1,3));
hold on;
plot(x,z,'Color',rand(1,3));
hold off;
```



- Adding multiple plots on the same figure: subplot

```
x = linspace(-5,5,1000).';
y = [x.^2, sin(x), cosh(x), exp(x), exp(-x).*sin(x), x];
colors = lines(6);
figure('Name','3-by-2 Plots','Color','white');
for i = 1:6
    subplot(3,2,i);
    plot(x,y(:,i),'Color',colors(i,:));
end
```

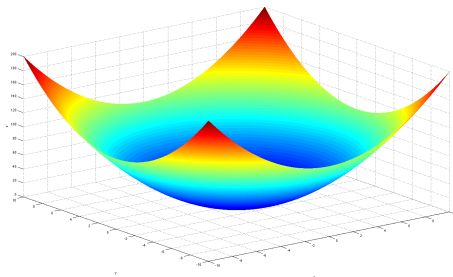


3.2 3-D Plotting

3-D Plotting

- In three dimensions, you can plot lines (plot3) and surfaces (surf, surfc, mesh, meshc)
- See MATLAB's Help for a description of all surface properties
- Set the current color map with the command colormap
- Basic example: plot $z = x^2 + y^2$

```
x = linspace(-10,10,1000);
y = x;
[X,Y] = meshgrid(x,y);
Z = X.^2 + Y.^2;
figure
surf(X,Y,Z,'EdgeColor','none');
xlabel('x');
ylabel('y');
zlabel('z');
```



- Adding contours to $z = x^2 - y^2$

```
x = linspace(-5,5,50);  
y = x;  
[X,Y] = meshgrid(x,y);  
Z = X.^2 - Y.^2;  
figure  
colormap('cool');  
meshc(X,Y,Z);  
xlabel('x');  
ylabel('y');  
zlabel('z');
```

