

Matlab Workshop II

Niloufer Mackey and Lixin Shen

Department of Mathematics

Western Michigan University

Email: `{nil.mackey,lixin.shen}@wmich.edu`

M-files

- Files that contain code in the Matlab language are called *M-files*. You create M-files using a text editor

M-files

- Files that contain code in the Matlab language are called *M-files*. You create M-files using a text editor
- There are two kinds of *M-files*

M-files

- Files that contain code in the Matlab language are called *M-files*. You create M-files using a text editor
- There are two kinds of *M-files*
 - **Scripts**: *do not accept input arguments or return arguments. They operate on data in the workspace*

M-files

- Files that contain code in the Matlab language are called *M-files*. You create M-files using a text editor
- There are two kinds of *M-files*
 - **Scripts**: *do not accept input arguments or return arguments. They operate on data in the workspace*
 - **Functions**: *accept input arguments and return output arguments. Internal variables are local to the function*

M-files

- Files that contain code in the Matlab language are called *M-files*. You create M-files using a text editor
- There are two kinds of *M-files*
 - **Scripts**: *do not accept input arguments or return arguments. They operate on data in the workspace*
 - **Functions**: *accept input arguments and return output arguments. Internal variables are local to the function*
- The name of each M-file must end in “.m” and must be found on MATLAB’s search **path**.

Scripts

- A script may contain any sequence of MATLAB statements, including references to other M-files. For example, create a file called `magicrank.m` that contains these Matlab commands:

```
% Investigate the rank of magic squares
r = zeros(1,14);
for n = 3:14
    r(n) = rank(magic(n));
end
r
bar(r)
```

Scripts (cont')

To invoke the script `magiccrank.m`, just typing the statement

```
magiccrank
```

causes

- execute the commands

Scripts (cont')

To invoke the script `magiccrank.m`, just typing the statement

```
magiccrank
```

causes

- execute the commands
- compute the rank of the first 14 magic squares

Scripts (cont')

To invoke the script `magiccrank.m`, just typing the statement

```
magiccrank
```

causes

- execute the commands
- compute the rank of the first 14 magic squares
- plot bar graph of the result

Scripts (cont')

To invoke the script `magiccrank.m`, just typing the statement

```
magiccrank
```

causes

- execute the commands
- compute the rank of the first 14 magic squares
- plot bar graph of the result
- variables `n` and `r` remain in the workspace

Functions

- Functions are M-files that can accept input arguments and return output arguments.

Functions

- Functions are M-files that can accept input arguments and return output arguments.
- The names of the M-file and of the function should be the same

Functions

- Functions are M-files that can accept input arguments and return output arguments.
- The names of the M-file and of the function should be the same
- Functions operate on variables within their own workspaces

Functions

- Functions are M-files that can accept input arguments and return output arguments.
- The names of the M-file and of the function should be the same
- Functions operate on variables within their own workspaces
- Write a function called `mytridiag` that takes as input n , and returns the $n \times n$ tridiagonal matrix that has the pattern suggested by

$$\begin{pmatrix} 2 & -1 & 0 & 0 \\ -1 & 4 & -1 & 0 \\ 0 & -1 & 6 & -1 \\ 0 & 0 & -1 & 8 \end{pmatrix}.$$

Functions (Cont')

```
function [A] = mytridiag(n)
% MYTRIDIAG(n) creates an nxn tridiagonal
% matrix with the main diagonal elements
% 2, 4, ..., 2n and two off-diagonal
% elements all being -1s

A=diag(2*(1:n))+diag(-ones(1,n-1),1) ...
    +diag(-ones(1,n-1),-1);
```

Functions (Cont')

- `function [A] = mytridiag(n)`
`% MYTRIDIAG(n) creates an nxn tridiagonal`
`% matrix with the main diagonal elements`
`% 2, 4, ..., 2n and two off-diagonal`
`% elements all being -1s`

```
A=diag(2*(1:n))+diag(-ones(1,n-1),1) ...  
    +diag(-ones(1,n-1),-1);
```

- `help mytridiag`

Functions (Cont')

- `function [A] = mytridiag(n)`
`% MYTRIDIAG(n) creates an nxn tridiagonal`
`% matrix with the main diagonal elements`
`% 2, 4, ..., 2n and two off-diagonal`
`% elements all being -1s`

```
A=diag(2*(1:n))+diag(-ones(1,n-1),1) ...  
    +diag(-ones(1,n-1),-1);
```

- `help mytridiag`
- `mytridiag(12)`

Graphics

- Creating a plot: `plot`

```
x=0:pi/100:2*pi; y=sin(x); plot(x,y);
```

Graphics

- Creating a plot: `plot`

```
x=0:pi/100:2*pi; y=sin(x); plot(x,y);
```

- Multiple data sets on a graph

```
x=0:pi/100:2*pi; y=sin(x); y1=sin(x-0.25);  
y2=sin(x-0.5); plot(x,y,x,y1,x,y2);  
legend('sin(x)', 'sin(x-0.25)', 'sin(x-0.5)');
```

We also need to know the command `hold on`

Graphics

- Creating a plot: `plot`

```
x=0:pi/100:2*pi; y=sin(x); plot(x,y);
```

- Multiple data sets on a graph

```
x=0:pi/100:2*pi; y=sin(x); y1=sin(x-0.25);  
y2=sin(x-0.5); plot(x,y,x,y1,x,y2);  
legend('sin(x)', 'sin(x-0.25)', 'sin(x-0.5)');
```

We also need to know the command `hold on`

- Multiple plots in one figure: `subplot(m,n,p)`

```
subplot(2,2,1); plot(x,y);  
subplot(2,2,2); plot(x,y1);  
subplot(2,2,3); plot(x,y2);  
subplot(2,2,4); plot(x,y1+y2);
```

Graphics(cont')

- Axis labels and titles: `xlabel`, `ylabel`, `title`, `text`

```
plot(x,y); axis([0,2*pi,-1,1]);  
xlabel('x is between 0 and 2pi');  
ylabel('y=sin(x)');  
title('Graph of the sine function');  
text(3, 0.5, 'y=sin(x)');
```

Graphics(cont')

- Axis labels and titles: `xlabel`, `ylabel`, `title`, `text`

```
plot(x,y); axis([0,2*pi,-1,1]);  
xlabel('x is between 0 and 2pi');  
ylabel('y=sin(x)');  
title('Graph of the sine function');  
text(3, 0.5, 'y=sin(x)');
```

- Saving and printing figures

Graphics(cont')

- Axis labels and titles: `xlabel`, `ylabel`, `title`, `text`

```
plot(x,y); axis([0,2*pi,-1,1]);  
xlabel('x is between 0 and 2pi');  
ylabel('y=sin(x)');  
title('Graph of the sine function');  
text(3, 0.5, 'y=sin(x)');
```

- Saving and printing figures
- Some other useful commands: `gtext`, `semilogx`, `semilogy`

Exercises

- Write a script file to graph $y = \sin x$, and two of its Taylor polynomials — the linear approximation $y = x$ and the fifth degree approximation $y = x - \frac{1}{3!}x^3 + \frac{1}{5!}x^5$ for $x \in [0, 2\pi]$.

Exercises

- Write a script file to graph $y = \sin x$, and two of its Taylor polynomials — the linear approximation $y = x$ and the fifth degree approximation $y = x - \frac{1}{3!}x^3 + \frac{1}{5!}x^5$ for $x \in [0, 2\pi]$.
 - All three graphs must appear in the same window

Exercises

- Write a script file to graph $y = \sin x$, and two of its Taylor polynomials — the linear approximation $y = x$ and the fifth degree approximation $y = x - \frac{1}{3!}x^3 + \frac{1}{5!}x^5$ for $x \in [0, 2\pi]$.
 - All three graphs must appear in the same window
 - Use different line styles to distinguish the graphs

Exercises

- Write a script file to graph $y = \sin x$, and two of its Taylor polynomials — the linear approximation $y = x$ and the fifth degree approximation $y = x - \frac{1}{3!}x^3 + \frac{1}{5!}x^5$ for $x \in [0, 2\pi]$.
 - All three graphs must appear in the same window
 - Use different line styles to distinguish the graphs
 - Include a background grid

Exercises

- Write a script file to graph $y = \sin x$, and two of its Taylor polynomials — the linear approximation $y = x$ and the fifth degree approximation $y = x - \frac{1}{3!}x^3 + \frac{1}{5!}x^5$ for $x \in [0, 2\pi]$.
 - All three graphs must appear in the same window
 - Use different line styles to distinguish the graphs
 - Include a background grid
 - Label the graphs using `gtext`, and label the axes using `xlabel` and `ylabel`

Exercises

- Write a script file to graph $y = \sin x$, and two of its Taylor polynomials — the linear approximation $y = x$ and the fifth degree approximation $y = x - \frac{1}{3!}x^3 + \frac{1}{5!}x^5$ for $x \in [0, 2\pi]$.
 - All three graphs must appear in the same window
 - Use different line styles to distinguish the graphs
 - Include a background grid
 - Label the graphs using `gtext`, and label the axes using `xlabel` and `ylabel`
 - Give a title to your plot using `title`.

Exercises

- Write a script file to graph $y = \sin x$, and two of its Taylor polynomials — the linear approximation $y = x$ and the fifth degree approximation $y = x - \frac{1}{3!}x^3 + \frac{1}{5!}x^5$ for $x \in [0, 2\pi]$.
 - All three graphs must appear in the same window
 - Use different line styles to distinguish the graphs
 - Include a background grid
 - Label the graphs using `gtext`, and label the axes using `xlabel` and `ylabel`
 - Give a title to your plot using `title`.
- How would you investigate the error between $y = \sin x$ and its fifth degree approximation on the interval $[0, \pi/2]$?