Introduction to MATLAB

SAMSI Undergraduate Workshop

May 21, 2007
Overview

1 Basics
   • Defining Vectors and Matrices
   • Operators

2 Programming in MATLAB
   • Scripts vs. Functions
   • Programming Syntax

3 MATLAB Functions
   • For Vectors and Matrices
   • For Solving ODEs
   • For Optimization

4 Other Useful Info
   • Displaying Results
   • Uploading/Saving Data
   • Miscellaneous
   • Getting Help
Defining Vectors and Matrices

Row Vector:

```
>> r = [1 2 3 4];
>> r = [1,2,3,4];
```

Column Vector:

```
>> c = [1;2;3;4];
>> c = [1
    2
    3
    4];
```

Matrix:

```
>> M = [1 2 3;4 5 6;7 8 9];
>> M = [1 2 3;
    4 5 6;
    7 8 9];
```
Defining Vectors and Matrices

- To change the 4th element of the vector \( v \) to 7, type:
  \[
  \text{>> } v(4) = 7
  \]
- To change the \((5, 2)\) entry of the matrix \( M \) to 3, type:
  \[
  \text{>> } M(5, 2) = 3
  \]
Operators

• `+`, `−`, `∗`, `^` work as expected

• examples
  
  ```matlab
  >> [1 2 3 4]*[5;6;7;8]

  ans =

  70
  
  >> [1;2;3;4]*[5 6 7 8]

  ans =

  5   6   7   8
  10  12  14  16
  15  18  21  24
  20  24  28  32
  ```
• Dotted operators (.*,./,.^) work componentwise
• example

```matlab
>> [1;2;3;4].*[5;10;15;20]
```

```
ans =

  5
 20
 45
 80
```
Operators

• Notice the difference:
  • \( A = B = \begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix} \)
  • \( A \ast B = \begin{bmatrix} 7 & 10 \\ 15 & 22 \end{bmatrix} \)
  • \( A \ast B = \begin{bmatrix} 1 & 4 \\ 9 & 16 \end{bmatrix} \)
The ' operator is used to transpose a matrix or vector. Examples:

\[ x = \begin{bmatrix} 1 \\
2 \\
3 \\
4 \end{bmatrix} \quad \Rightarrow \quad x' = \begin{bmatrix} 1 & 2 & 3 & 4 \end{bmatrix} \]

\[ M = \begin{bmatrix} 1 & 2 & 3 & 4 \\
5 & 6 & 7 & 8 \end{bmatrix} \quad \Rightarrow \quad M' = \begin{bmatrix} 1 & 5 \\
2 & 6 \\
3 & 7 \\
4 & 8 \end{bmatrix} \]
A common mistake is to try to perform an operation with vectors/matrices of the wrong dimensions. For example, if you try to multiply two column vectors, you will get the following error message:

```
??? Error using ==> mtimes
Inner matrix dimensions must agree.
```
Operators

- The : operator
  - To access the $i$th row of the matrix $M$, use $M(i,:)$
  - To access the $j$th column of the matrix $M$, use $M(:,j)$
  - $[3:7]$ is the same as $[3,4,5,6,7]$
  - $[3:2:11]$ is the same as $[3,7,9,11]$
At this point you should be able to complete section 1 of the practice problems.
• Often, we need to run the same commands multiple times
  • Because we have different cases to look at
  • Because we did something wrong the first time
• m-files allow us to save chunks of code which can be used again later
• There are two types of m-files: Scripts and Functions.
Scripts vs. Functions

- **Scripts**
  - Made up of a list of commands as if you were typing them directly on the command line
  - No special syntax required
  - example:
    ```
    x = linspace(-2*pi,2*pi,49);
    y = sin(x);
    plot(x,y);
    max(y)
    ```
  - **NOTE:** The following are equivalent:
    - `x = linspace(-2*pi,2*pi,49);`
    - `x = [-2*pi:pi/12:2*pi];`
Scripts vs. Functions

- Functions
  - First line specifies that the m-file is a function, the return values, and the input parameters
  - Makes code easier to read
  - Example:

```matlab
function out = exfun(a,b,c)
    % solves quadratic polynomial
    % of the form ax^2 + bx + c = 0
    x1 = (-b + sqrt(b*b - 4*a*c))/(2*a);
    x2 = (-b - sqrt(b*b - 4*a*c))/(2*a);
    out = [x1, x2];
```

Programming Syntax - The “if” statement

Allows different actions based on the state of certain variables.
Example:

```matlab
if j==1
    x=1;
elseif j>1 & i~=3
    x=2;
else
    x=3;
end
```
Programming Syntax - The “for” loop

Allows the repetition of a section of code a set number of times. Example:

```matlab
x = zeros(100,100);

for i=1:100
    for j=1:100
        x(i,j) = cos(j)*sin(i);
    end
end
```
count = 0;
tol = 1e-10;
err = 100;   %something > tol
xold = 0;
while (err>tol) & (count<=500)
    fofx = (xold^2+2*xold-1)*exp(xold);  %(x^2+2x-1)e^x
    fprimeofx = (xold^2+4*xold+1)*exp(xold);
    xnew = xold-fofx/fprimeofx;  %Newton step
    err = xnew-xold;  %how close is xold to xnew?
    count = count+1;  %increment counter
    xold = xnew;
end
count
At this point you should be able to complete section 2 of the practice problems.
MATLAB Functions for Vectors and Matrices

- Creating Common Matrices
  - $m \times n$ matrix of all zeros: `zeros(m,n)`
  - $m \times n$ matrix of all ones: `ones(m,n)`
  - $n \times n$ identity matrix: `eye(n)`

- Determining Size
  - length of a vector $v$: `length(v)`
  - size of a matrix or vector $x$: `size(x)`
MATLAB Functions for Vectors and Matrices

- The solution to the linear system $Ax = b$ where $x$ is a vector of unknowns is $x = A^{-1}b$
- In MATLAB, the `inv` function will return the inverse of $A$
- Assuming $A$ and $B$ have been defined previously, $x$ can be found using the following:

$$x = \text{inv}(A) \times b$$
MATLAB Functions for Vectors and Matrices

- **Statistics**
  - Mean: `mean(x)`
  - Standard Deviation: `std(x)`
  - Variance: `var(x)`

- **Other**
  - `norm(x)`
  - `max(x)`, `min(x)`
  - `sum(x)`
  - `eig(x)` (calculates eigenvalues of a matrix)
MATLAB Functions for Solving ODEs

1. Syntax

\[ [t, y] = \text{ode45}(\text{@fun}, [t0 \ tf], y0, \text{options}, a1, a2, \ldots) \]

- **Input**
  - `@fun` - name of a function
  - `[t0 tf]` - timespan
  - `y0` - initial condition
  - `options` - ODE solver options
  - `a1, a2, \ldots` - arguments to be passed to your function

- **Output**
  - `t` - times at which solutions were calculated
  - `y` - solutions at given times
The function defining the ODE

- Consider the ODE $t^2y' = \sin(at)$ where $a$ is a parameter
- Rewrite the ODE in the form $y' = f(t, y(t))$:

$$y' = \frac{\sin(at)}{t^2}$$

- The function defining your ODE would then be as follows:

```matlab
function dy = fun(t,y,a)
    dy = sin(a*t)/(t^2);
end
```

What do you do when the ODE is 2nd order or higher?
MATLAB Functions for Solving ODEs

Handling ODEs of 2nd order or higher

\[ m\ddot{y}(t) + c\dot{y}(t) + ky(t) = \cos(t) \]
\[ y(0) = 0 \]
\[ \dot{y}(0) = 0 \]

If we let \( y_1 = y \) and \( y_2 = \dot{y}_1 \). Then \( \dot{y}_2 = \ddot{y}_1 = \ddot{y} \). Which means we can rewrite our original ODE as

\[
\begin{bmatrix}
\dot{y}_1 \\
\dot{y}_2
\end{bmatrix} = \begin{bmatrix}
0 & 1 \\
-\frac{k}{m} & -\frac{c}{m}
\end{bmatrix}
\begin{bmatrix}
y_1 \\
y_2
\end{bmatrix} + \begin{bmatrix}
0 \\
\frac{1}{m} \cos(t)
\end{bmatrix}.
\]
In this case, the function defining our ODE would be as follows:

```matlab
function dy = fun(t,y,m,c,k)
A = [ 0 1
    -k/m -c/m];
b = [0;cos(t)/m];
dy = A*y+b;
```
The function `fminsearch` can be used to find a local minimum of a scalar function.

Syntax: \( x = \text{fminsearch}(\text{@fminfun}, x0) \)
- `fminfun` - scalar function taking a scalar input
- `x0` - initial condition

NOTE: `fminsearch` may or may not produce accurate results depending on the starting point you give it. A minimum will be reached, but it may not be global.
At this point you should be able to complete section 3 of the practice problems.
Displaying Results

```matlab
x = linspace(-pi,pi,101);
plot(x,sin(x),x,cos(x))
```
x = linspace(-pi,pi,101);
plot(x,sin(x),'-x',x,cos(x),'r-o')
xlabel('x')
ylabel('y')
title('Sine and Cosine near x=0')
legend('Sine','Cosine')
axis([-pi pi -1 1])
Displaying Results

Sine and Cosine near x=0

- Sine
- Cosine
Let’s say we have a data file `data.in` that looks like

1 10
2 20
3 30
4 40
5 50

We can load this into MATLAB using the command

```
load data.in
```

The data will be stored in the variable `data`. 
What if you have results that you’ve calculated in MATLAB and you want to save them for later use.

```matlab
save mystuff
```

will save all variables currently in memory to a file named mystuff.mat in the current directory.
• clear, clear x, clear x*
• close, close all, clf
Getting Help

- **help function**
  - for more information about a specific function
  example: `help plot`
- “Full Product Family Help”
  - can be accessed from the “Help” menu
At this point you should be able to complete section 4 of the practice problems.
Questions?