MATLAB
INTRODUCTION

January 22, 2014
CMDA 3606
Outline

• What is MATLAB
• Matrix/vector creation & manipulation
• Built-in functions
• Solving systems of equations
• Plotting data
• Programming logic
MATLAB® is a high-level language and interactive environment for numerical computation, visualization, and programming. Using MATLAB, you can analyze data, develop algorithms, and create models and applications.
MATLAB

- Matrix-based computational language
  - MATLAB stands for MATrix LABoratory

- Excellent for Linear Algebra

- Contains libraries of built-in functions

- Powerful visualization capabilities
MATLAB Interface

```matlab
for blah = 3:3:@nuem(N_vec)
    N = N_vec(blah);
    dx = length/N;
    x = dx/2:dx:length-dx/2;
    delta = 2; % times dx
    M = zeros(N); b = zeros(N,1);
    [~,left] = min(abs((1/3)*length - x));
    [~,right] = min(abs((2/3)*length - x));
    b(left) = 2; b(right) = -2;
    for i = 1:N
        index = i - linspace(-delta,delta,2*delta+1);
        index(index < 1 | index > N | index == 1) = [];
        for p = 1:nuem(index)
            if abs(i - index(p)) == delta
                M(i,p) = M(i,p) + 1/(dx*2);
                M(i,i) = M(i,i) - 1/(dx*2);
            else
        ```
Building a Matrix

- Directly entering the elements:

\[ A = [2 \ 8 \ -4; \ 1 \ 0 \ 6; \ -2 \ -8 \ 3] \]

- Use brackets ‘[…]' to begin and end matrices
- Use spaces or commas to separate elements in the same row
- Use semicolons to separate rows

- Matrix size is written \( m \times n \)
  - Matrix has \( m \) rows and \( n \) columns

NOTE: The matrix above is now stored as “A” and can be recalled using this variable name. Also, variable names are case sensitive.
Building a Vector

• Vectors can be viewed as $1 \times n$ or $m \times 1$ matrices

```plaintext
>> v = [5; -2; 4; 0]
```

• Create vectors directly:

```plaintext
v =

5
-2
4
0
```

• Create vectors of equally spaced intervals:

```plaintext
>> v = 1:2:10
```

```plaintext
v =

1  3  5  7  9
```
Other Matrix & Vector Constructions

- \texttt{zeros(m,n)}: \textit{m} x \textit{n} matrix of zeros

- \texttt{ones(m,n)}: \textit{m} x \textit{n} matrix of ones

- \texttt{eye(n)}: \textit{n} x \textit{n} identity matrix

- \texttt{linspace(x0,x1,n)}: \textit{1} x \textit{n} vector of uniformly spaced points from \textit{x0} to \textit{x1}
Vector/Matrix Manipulation

- Access specific elements in a matrix or vector
- Change specific elements in a matrix or vector
- Concatenating matrices/vectors
- Delete row/column of a matrix
- Delete elements in a vector
- Transpose a matrix or vector
  - Careful when using complex values
Matrix Operations

- Addition: +
- Subtraction: –
- Multiplication: *
- Division: /
- Exponential: ^
- Left division: \ 
  - Solves system $Ax = b$ for $x$ by entering $x = A\b$
Array Operations

• Operations are performed element-wise rather than in the traditional sense
• Referred to as “dot-” operation

  • Multiplication: . *

  • Division: . /

  • Exponential: . ^

NOTE: when performing array operations, both arrays must be the same size or one of the arrays must be a scalar
Built-in Functions

- \text{size}(A): \text{size of } A

- \text{sum}(A, \text{dim}): \text{sum of the values of } A \text{ along dimension } \text{dim}

- \text{min}(A, [], \text{dim})/\text{max}(A, [], \text{dim}): \text{find minimum/maximum elements along dimension } \text{dim}

- \text{mean}(A, \text{dim}): \text{find mean of elements along dimension } \text{dim}

- \text{det}(A): \text{calculates determinant of } A

- \text{inv}(A): \text{returns } A^{-1} \text{ if it exists}
Built-in Functions, continued…

- \( \exp(A) \): raises \( e \) to each element in \( A \)
  - NOTE: this is NOT the same as matrix exponentiation

- \( \log(A) \): performs \( \log_e( \cdot ) \) to each element of \( A \)

- \( \sin(A)/\cos(A)/\tan(A) \): performs appropriate trig function to each element of \( A \)

- \( \sqrt[\wedge]{A} \): finds square root of each element of \( A \)

- \( \text{dot}(a, b) \): dot product of vectors \( a \) and \( b \)
  - Can also be found using \( * \)

- \( \text{cross}(a, b) \): cross product of vectors \( a \) and \( b \)
Plotting Data

• Want to plot $y = f(x)$
  • Build vector of $x$ points
  • Calculate $f$ at each $x$ value
  • Plot data using `plot(x, y)` function

NOTE: the $x$ and $y$ must be vectors of the same length

• Can create title and label axes
  • `title('Type title in single quotes here','FontSize',##)`
  • `xlabel('Type x-axis label in single quotes here')`
  • `ylabel('Type y-axis label in single quotes here')`
Plotting Data

• Other ways to edit plots:

  • Specifying plot colors, line styles, and marker types

    \texttt{Plot(x,y,color/line\ style/marker\ type\ info\ here)}

    | Colors | Line Styles | Marker Types |
    |--------|-------------|--------------|
    | r      | Red         | +            | Plus Sign   |
    | g      | Green       | o            | Circle      |
    | b      | Blue        | *            | Asterisk    |
    | y      | Yellow      | .            | Point       |
    | k      | Black       | x            |             |

    \textbf{NOTE:} color/line style/marker type info must be contained in quotes
    \textbf{Example:} \texttt{plot(x,y,'k--x')}

• Plotting multiple functions on one graph

    \texttt{Plot(x1,y1,x2,y2,x3,y3,...)}
Creating Multiple Plots

• Assume I want 2 separate plots with 2 pieces of info

• What happens if I type in the following:
  
x = linspace(0,1,20);
y1 = x.^2;
y2 = x + 1;
plot(x,y2)
plot(x,y2)

• Solutions:
  • Use the command **hold on** before the 2\textsuperscript{nd} plot
    • Saves previous graph when adding 2\textsuperscript{nd} graph
  • Use the command **figure** before the 2\textsuperscript{nd} plot
    • Creates new figure which will contain the 2nd graph
.m Files

• A .m file is the basic MATLAB file containing code

• .m files come in 2 forms:
  • Scripts
    • No name, inputs, outputs
    • Cannot be called by any other function
  
• Functions
  • Has the following form:
    \[
    \text{function} \ [\text{outputs}] = \text{function\_name}(\text{inputs})
    \]
Programming Logic

• MATLAB contains many tools to help control the logic within a program

  • for loops

```matlab
for i = 1:10
    Do something 10 times
end
```

NOTE: it is common for a for loop to perform a computation using the index `i`

• while loops

```matlab
while logical statement
    Do something repetitively until logical statement becomes false
end
```
Programming Logic

- **if/else/elseif** conditional statements:

  ```
  if logical statement
      Do something
  elseif different logical statement
      Do something
  else
      Do something
  end
  ```

**NOTE:** the **elseif** and **else** parts are optional & you can use as many **elseif** statements as desired
Example

- Open the population.m file from the course website

- Make it into a function with $A$ as an input and $P$ as an output

- Build a 5x5 matrix $A$ and run the function by entering
  \[ P = \text{population}(A) \]

- Change $s_3$ to 0.9 and reduce $f_1$, $f_2$, and $f_3$ by 0.1 and rerun the program

- How many females between 15 and 29 are in the 10th, 15th, and 20th generations?
Example, continued…

• Add a title to the graph and give it font size 22

• Create a new variable `pop` which is a vector containing the total population at each generation

• Using `pop`, create a vector `growth` which is the ratio of each generation’s population to the previous population

• Plot `growth` against each generation to see the trend towards a constant growth factor