Computational Methods
CMSC/AMSC 460

Ramani Duraiswami,
Dept. of Computer Science

Matlab Windows

- Command line Interface (Main Window)
- Editor Window
- Present Directory
- Directory Contents and Workspace variables
- Command line
- Command History
Matrices in Matlab

- Entering a Matrix:

  ```matlab
  >> A = [ 0 -0.8 -0.6 ; 0.8 -0.36 0.48 ; 0.6 -0.48 -0.64]
  A =
  0 -0.8000 -0.6000
  0.8000 -0.3600 0.4800
  0.6000 0.4800 -0.6400
  ```

- Matrix referencing:

  ```matlab
  >> A(1,2)
  ans =
  -0.8000
  >> A(2,:)
  ans =
  0.8000
  -0.3600
  -0.6000
  0.4800
  ```

- Matrix Operations:

  ```matlab
  >> A+A;
  >> A.*A;
  >> 3*A;
  >> A*A
  ans =
  -1.0000 0 0
  0 -0.2800 -0.9600
  0 -0.9600 0.2800
  ```

**Built-in functions**

- Inverting a Matrix

  ```matlab
  >> inv(A)
  ans =
  -0.0000 0.8000 0.6000
  -0.8000 -0.3600 0.4800
  -0.6000 0.4800 -0.6400
  ```

- Determinant

  ```matlab
  >> det(A)
  ans =
  -1.0000
  ```

- Transpose of a Matrix

  ```matlab
  >> A'
  ans =
  0 0.8000 0.6000
  -0.8000 -0.3600 0.4800
  -0.6000 0.4800 -0.6400
  ```

- Rank

  ```matlab
  >> rank(A)
  ans =
  3
  ```
Introduction to MATLAB

- Vectors, Matrices, Syntax
- Vector operations, including the `dot` commands
  - length, size, linspace, logspace, size, rand, randn, randperm
- Special vectors and matrices: zeros, ones, eye, magic
- Scripts and functions
  - Diary
- Graphing:
  - plot, special fonts, plot3, semilogx, semilogy, title, xlabel, ylabel, axis, grid, legend, subplot,
- Formatted output:
  - Sprintf, ;, disp, input
- Programming:
  - for, if, while, &, |, ~
- General/misc commands
  - ginput set, size, max, sum, close, figure, hist, any, all, floor, fix, round,
- Graphical programming and callbacks

Plotting a function

\[ y_1 = \frac{7x}{0.6 + x} \quad y_2 = \frac{5x}{0.08 + x} \]

\[
\begin{align*}
\text{>> } & x = [0:0.01:5]; \\
\text{>> } & y_1 = 7 \times x ./ (0.6 + x); \\
\text{>> } & y_2 = 5 \times x ./ (0.08 + x); \\
\text{>> } & \text{plot}(x,y_1,x,y_2) \\
\text{>> } & \text{legend('y1','y2')} \\
\end{align*}
\]
Matrix-vector product

• Matrix-vector multiplication

$$M \cdot v = \begin{bmatrix} M_{11} & M_{12} & M_{13} \\ M_{21} & M_{22} & M_{23} \\ M_{31} & M_{32} & M_{33} \end{bmatrix} \begin{bmatrix} v_1 \\ v_2 \\ v_3 \end{bmatrix}$$

• Recall how to do matrix multiplication
• How many operations does this matrix vector product take?
• How many operations does a general matrix vector product take?

Ways to implement a matrix vector product

• Access matrix
  – Element-by-element along rows
  – Element-by-element along columns
  – As column vectors
  – As row vectors
• Discuss advantages

```matlab
[m,n]=size(A);
y = zeros(m,1);
for i=1:m,
    for j=1:n,
        y(i) = y(i) + A(i,j)*x(j);
    end
end
```

```matlab
[m,n]=size(A);
y = zeros(m,1);
for i=1:m,
    y(i) = A(i,:) * x;
end
```

```matlab
[m,n]=size(A);
y = zeros(m,1);
for j=1:n,
    y = y + A(:,j)*x(j);
end
```
Good workmen know their tools

- Primitive model

Memory Hierarchy