International University
School of Electrical Engineering

Introduction to Computers for Engineers

Dr. Hien Ta

Lecturely Topics

Lecture 1 - Basics – variables, arrays, matrices
Lecture 2 - Basics – matrices, operators, strings, cells
Lecture 3 - Functions & Plotting
Lecture 4 - User-defined Functions
Lecture 5 - Relational & logical operators, if, switch statements
Lecture 6 - For-loops, while-loops
→ Lecture 7 - Review on Midterm Exam
Lecture 8 - Solving Equations & Equation System (Matrix algebra)
Lecture 9 - Data Fitting & Integral Computation
Lecture 10 - Representing Signal and System
Lecture 11 - Random variables & Wireless System
Lecture 12 - Review on Final Exam

References: H. Moore, *MATLAB for Engineers*, 4/e, Prentice Hall, 2014
G. Recktenwald, *Numerical Methods with MATLAB*, Prentice Hall, 2000
A. Gilat, *MATLAB, An Introduction with Applications*, 4/e, Wiley, 2011

Review Examples

1. User-defined function functions
2. Pitfalls in piece-wise function definitions
3. Finite **for-loop & while-loop** examples
4. Terminating divergent **while-loops**
5. Summing infinite series with prescribed accuracy
6. Vectorized forms of **fprintf / fscanf**

Example 1: User-defined function functions

function handle

M-file: **fmaxbnd.m**

```
function [xmax,fmax] = fmaxbnd(f,a,b)

[xmax,fmax] = fminbnd(@(x) -f(x), a, b);

fmax = -fmax;
```

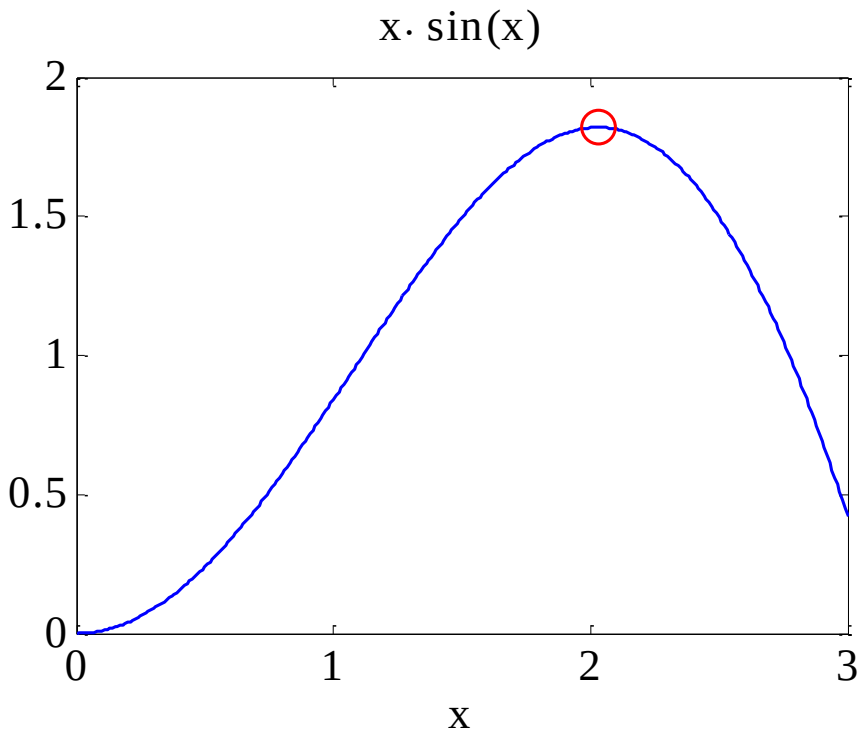
M-file: **xsin.m**

```
function y = xsin(x)

y = x.*sin(x);
```

anonymous definition

```
f = @(x) x.*sin(x);
```

```
[xmax,fmax] = fmaxbnd(@xsin,0,3);
[xmax,fmax] = fmaxbnd(f,0,3);

x = linspace(0,3,301);
y = xsin(x);
plot(x,y,'b', xmax,fmax,'ro');

[xmax,fmax]

ans =
    2.0288  1.8197
```



Example 2: Pitfalls in piece-wise function definitions

method 1 – vectorized – using relational operators
method 2 – vectorized – using the function **`find`**
method 3 – not vectorized – used with a for-loop

$$y = f(x) = \begin{cases} \exp(x+1), & \text{if } x < -1 \\ |x|, & \text{if } -1 \le x \le 1 \\ \exp(-x+1), & \text{if } x > 1 \end{cases}$$

implemented in the M-files: **`f1.m, f2.m, f3.m`**

method 1 – vectorized – using relational operators

M-file: **f1.m**

```
function y = f1(x)

y = exp(x+1).*(x<-1) + ...
    abs(x).*(abs(x)<=1) + ...
    exp(-x+1).*(x>1);
```

shape of x is automatically preserved

anonymous definition is best

```
f1 = @(x) exp(x+1).*(x<-1) + ...
          abs(x).*(abs(x)<=1) + ...
          exp(-x+1).*(x>1);
```

method 2 – vectorized – using the function **find**

M-file: **f2.m**

```
function y = f2(x)

y = zeros(size(x));    % preserve shape of x

i = find(x<-1);
y(i) = exp(x(i)+1);

i = find(x>1);
y(i) = exp(-x(i)+1);

i = find(abs(x)<=1);
y(i) = abs(x(i));
```

method 2 – vectorized – equivalent to using **find**

M-file: **f2b.m** – does not use **find** but it's hard to read

```
function y = f2b(x)

y = zeros(size(x));    % preserve shape of x

y(x<-1) = exp(x(x<-1)+1);

y(x>1) = exp(-x(x>1)+1);

y(abs(x)<=1) = abs(x(abs(x)<=1));
```

method 3 – not vectorized – must be used with a for-loop

scalar **x**

M-file: **f3.m**

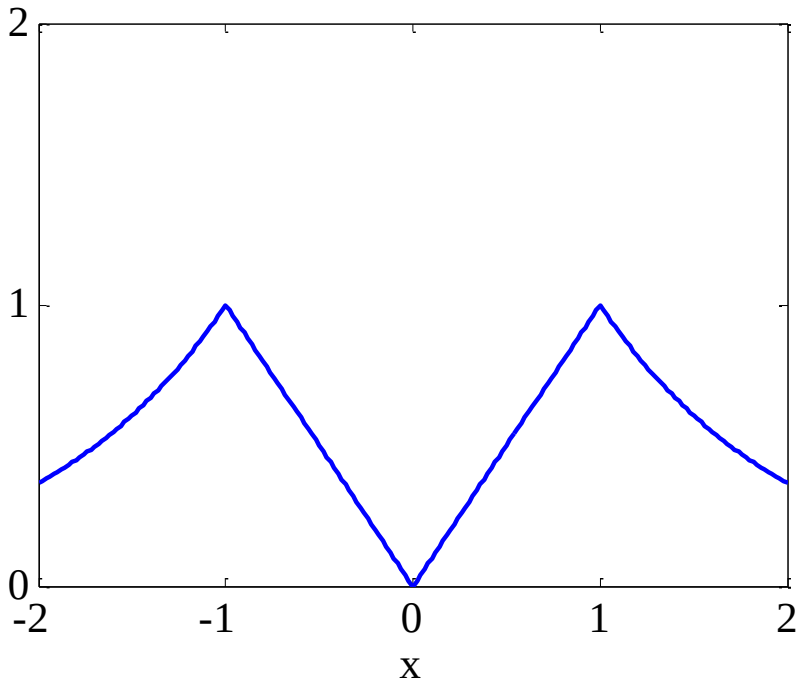
```
function y = f3(x)

if x<-1
     y = exp(x+1);
elseif x>1
     y = exp(-x+1);
else
     y = abs(x);
end
```

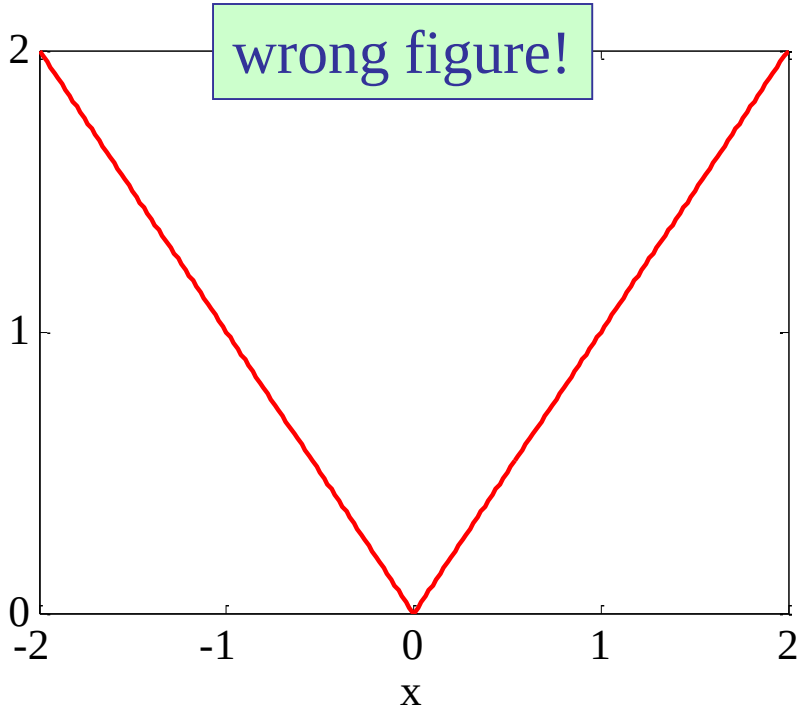
Pitfall:

although it most clearly displays the logic of the function definition, it produces the wrong result if a vector **x** is used as input

see explanation below

```
x = linspace(-2,2,201);

figure; plot(x,f1(x),'b');  % f2(x) is the same
yaxis(0,2,0:2);

figure; plot(x,f3(x),'r');  % f3(x), vector x
yaxis(0,2,0:2);
```





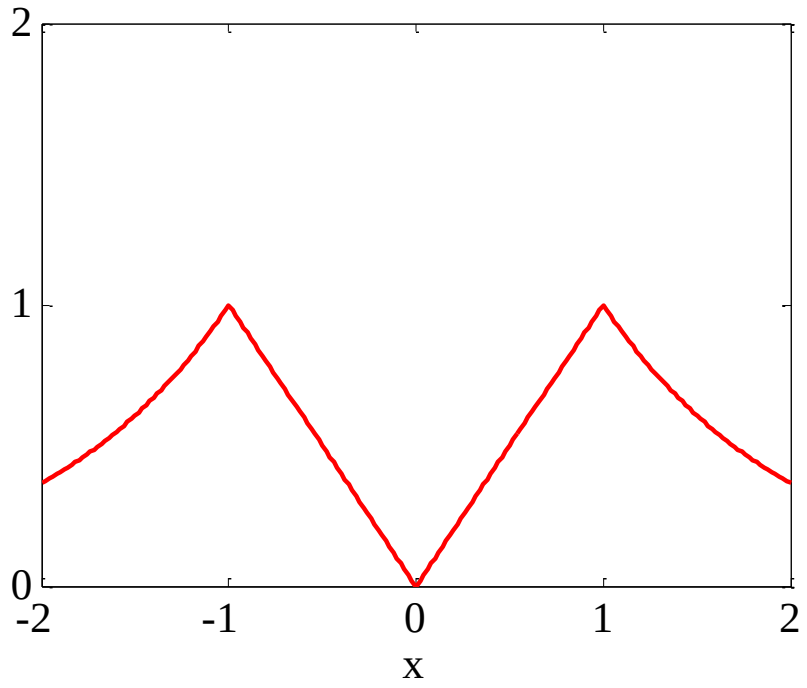
```
>> x = -2:0.5:2
x =
     -2   -1.5   -1   -0.5   0   0.5   1   1.5   2

>> x<-1
ans =
      1    1    0    0    0    0    0    0    0

>> x>1
ans =
      0    0    0    0    0    0    0    1    1
```

explanation of pitfall:

`if x<-1` test fails, then **`elseif x>1`** test also fails, hence, **`else`** is executed, so result is **`|x|`** for all **`x`**.

```
if x<-1
    y = exp(x+1);
elseif x>1
    y = exp(-x+1);
else
    y = abs(x);
end
```

```
% correct usage of f3(x)

for n=1:length(x)
   y(n) = f3(x(n));
end

figure; plot(x,y,'r');
yaxis(0,2,0:2);
```



Example 3: Finite for-loop & while-loop examples

Imitate the function **`S = sum(x)`**

$$\mathbf{x} = [x(1),\ x(2),\ x(3),\ \ldots,\ x(N)]$$

$$S = \sum_{n=1}^{N} x(n) = x(1)+x(2)+x(3)+\cdots+x(N)$$

convert to a recursion for the partial sums:

$$S_k = S_{k-1} + x(k)\,, \quad k = 1, 2, \ldots, N\,, \quad S_0 = 0$$

```
x = [10, -20, 30, 50, 5, 25];  % sum(x) = 100
```

```
S = 0;
for k=1:length(x)
    S = S + x(k);
end
```

for-loop

```
S = 0; k=1;
while k<=length(x)
    S = S + x(k);
    k = k+1;
end
```

conventional while-loop

```
S = 0; k=1;
while 1
    if k>length(x), break; end
    S = S + x(k);
    k = k+1;
end
```

forever while-loop

Imitate the function **`y = cumsum(x)`**

$$y(1) = x(1)$$

$$y(2) = x(1) + x(2) = y(1) + x(2)$$

$$y(3) = x(1) + x(2) + x(3) = y(2) + x(3)$$

$$y(4) = x(1) + x(2) + x(3) + x(4) = y(3) + x(4), \quad \text{etc.}$$

difference equation algorithm:

$$y(1) = x(1)$$

$$y(k) = y(k-1) + x(k), \quad k = 2, 3, \ldots, N$$

```
x = [10, -20, 30, 50, 5, 25];
```

```
y(1) = x(1);
for k=2:length(x)
    y(k) = y(k-1) + x(k);
end
```

for-loop

```
>> y
y =

    10   -10    20    70    75   100

>> y = cumsum(x)
y =

    10   -10    20    70    75   100
```

```
x = [10, -20, 30, 50, 5, 25];
```

```
y(1) = x(1); k = 2;
while k <= length(x)
    y(k) = y(k-1) + x(k);
    k = k+1;
end
```

conventional
while-loop,
it computes the values,
  y(2), k=3;
  y(3), k=4; etc.

```
y(1) = x(1); k = 2;
while 1
   if k > length(x),
     break;
   end
   y(k) = y(k-1) + x(k);
   k = k+1;
end
```

forever
while-loop

Imitating the function **find** using a for-loop

```
% consider the grades of 10 students in the vector g,
% find number of B+ grades in g, i.e., in range 85 <= g < 90
% find their locations within g, i.e., their student number
% find their actual numerical values

g = [67,85,95,87,88,75,89,70,76,86]     % numerical grades
G = grade(g)                            % letter grades

% vectorized method:

i = find(g >= 85 & g < 90)      % indices of B+'s
count = length(i)               % number of B+'s
Bp = g(i)                       % actual numerical values

% i =
%     2   4   5   7  10
% count =
%     5
% Bp =
%    85  87  88  89  86
```

Imitating the function **find** using a for-loop

```
% for-loop method:

count = 0;                              % initialize count
i = [];                                 % initialize indices
Bp = [];                                % initialize grades

for k = 1:length(g)
  if g(k) >= 85 & g(k) < 90             % check grade range
    i = [i,k];                          % append index
    count = count + 1;                  % update count
    Bp = [Bp,g(k)];                     % append numerical value
  end
end

% i =
%     2   4   5   7  10
% count =
%     5
% Bp =
%    85  87  88  89  86
```

double for-loops

```
N = 4; M = 5;

for i=1:N
    for j=1:M
        A(i,j) = 2^i+2^j;
    end
end

>> A

A =
     4     6    10    18    34
     6     8    12    20    36
    10    12    16    24    40
    18    20    24    32    48
```

arbitrary example:

$$A(i,j) = 2^i + 2^j$$

```
% vectorized row-wise

N=4; M=5; j=1:M;

for i=1:N
   A(i,:) = 2^i+2.^j;
end
```

```
% vectorized column-wise

N=4; M=5; i=1:N;

for j=1:M
   A(:,j) = 2.^i+2^j;
end
```

```
% fully vectorized

N=4; M=5;

i=1:N; j=1:M;

[J,I] = meshgrid(j,i);

A = 2.^I + 2.^J;
```

```
A =
     4     6    10    18    34
     6     8    12    20    36
    10    12    16    24    40
    18    20    24    32    48
```

why `(j,i)` instead of `(i,j)`?

Example 4: Terminating divergent while-loops

divergent geometric series

$$\sum_{k=0}^{\infty} = 1 + a + a^2 + a^3 + \cdots = \infty\,, \quad \text{if } |a| > 1$$

recursive calculation:

$$S_0 = 1$$

$$S_k = S_{k-1} + a^k\,, \quad k \geq 1$$

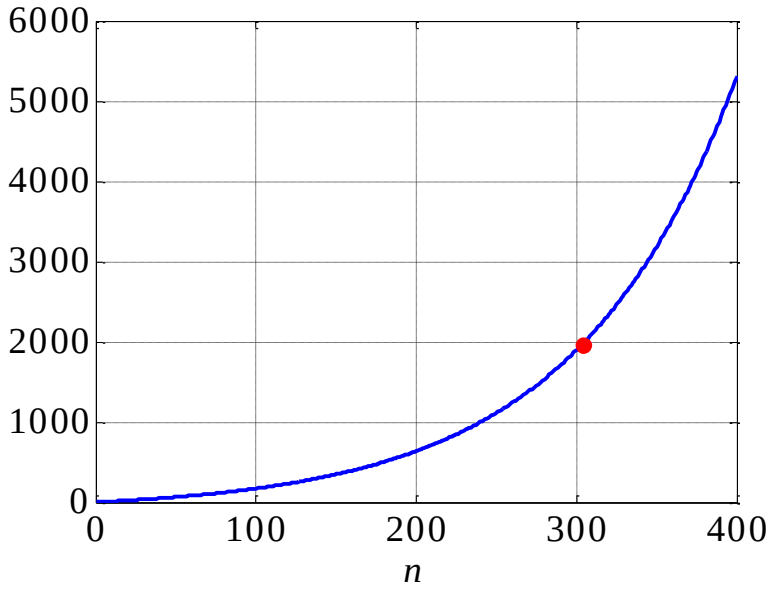
Problem: Find maximum k such that, $S_k < S_{\max}$

```
a = 1.01; Smax = 2000;

S = 1; k=0;

while S <= Smax
    k = k+1;
    S = S + a^k;
end

k=k-1, S, sum(a.^(0:k))

k =
    304
S =
   2000.6
ans =
   1979.8
```

loop terminates only after
k has been increased by one
and S has become S > Smax

```
a = 1.01; Smax = 2000;

S = 1; k=0;

while 1
    if S > Smax
       break;
    end
    k = k+1;
    S = S + a^k;
end

k=k-1, S, sum(a.^(0:k))

n=0:400;
y = cumsum(a.^n);
plot(n,y,'b', k,y(k),'r.')
```

forever while-loop



savings account

R = annual percentage rate

$$r = \frac{R}{1200} = \text{monthly rate}$$

y_0 = opening balance

x = monthly deposit

recursive calculation:

$$y(1) = y_0$$

$$y(k) = (1 + r)y(k - 1) + x, \quad k \geq 2$$

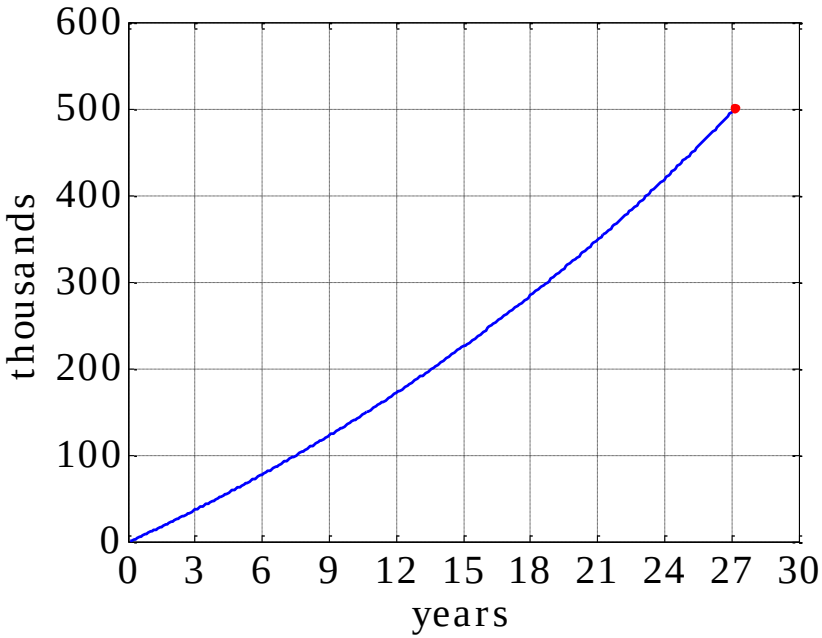
Problem: Find minimum k such that, $y(k) \geq y_{\max}$

```
R = 3; r = R/1200; a=1+r;
y0 = 1000; x = 1000;
ymax = 500000;
```

```
y(1) = y0; k=1;

while y(k) < ymax
    k = k+1;
    y(k) = a*y(k-1) + x;
end
```

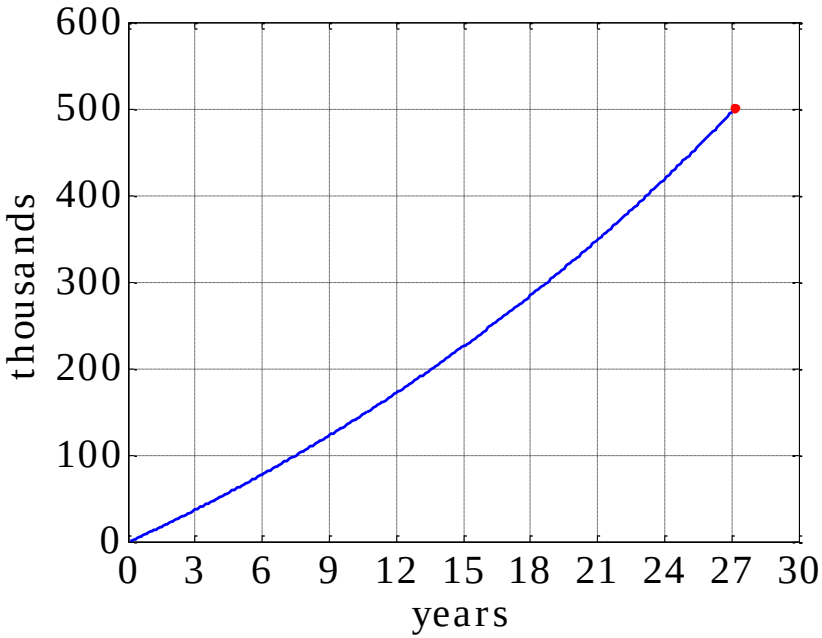
conventional while-loop

```
k, y(k)

% k=325 = 27 yrs + 1 mo
% y(k) = $ 500500.40
% x*k = $ 325000

n=1:k;
plot(n/12, y/1e3,'b')
hold on
plot(k/12, y(k)/1e3,'r.')
```



```
y(1) = y0; k=1;
while 1
    k = k+1;
    y(k) = a*y(k-1) + x;
    if y(k) >= ymax
        break;
    end
end
k, y(k)
% k=325 = 27 yrs + 1 mo
% y(k) = $ 500500.40
n=1:k;
plot(n/12, y/1e3,'b')
hold on
plot(k/12, y(k)/1e3,'ro')
```

forever while-loop



Find Midterm Exam Sample for your preparation to Midterm