

MATLAB® R2009b
The Language of Technical Computing

**Engineering Computing
and Problem Solving with Matlab**

Implementing the bisection algorithm with a
Matlab function
Using Matlab's debugging tools

**Writing functions with function-name
arguments**

Example: write a Matlab function that creates
an x-y plot of a function $y = f(x)$ for $a \leq x \leq b$

First, create the function $f(x)$

```
function y=f(x)  
y=exp(-x/4).* (2-x)-1;
```

We would like to have a general plotting
function,

```
plotxy(function_to_plot,xlow,xhigh)
```

This would look like

```
plotxy(@f,0,3) for example
```

The @ indicates that f is a function, not a variable

**Writing functions with function-name
arguments**

```
function result=plotxy(fun,x1,x2)  
x=linspace(x1,x2);  
y=feval(fun,x);  
plot(x,y,'k')  
grid
```

This takes advantage of the special Matlab
function, feval, which evaluates a given function,
here represented by fun, for an argument value x.

Then, to test the function,

```
>> plotxy(@f,0,3)
```

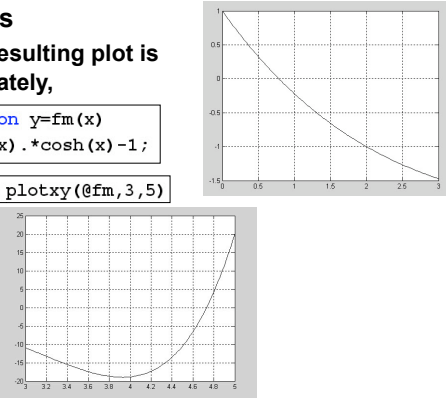
Writing functions with function-name arguments

and the resulting plot is or, alternately,

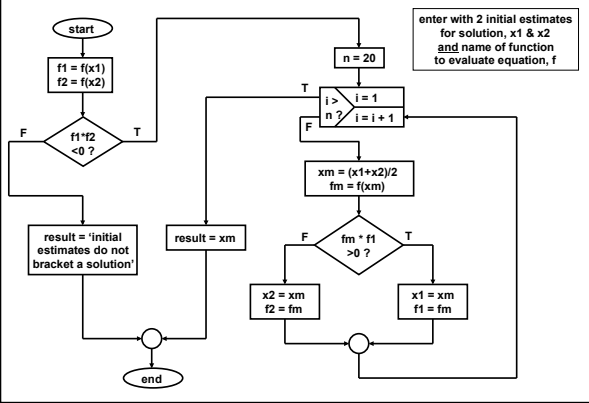
```
function y=fm(x)
y=cos(x).*cosh(x)-1;
```

and `>> plotxy(@fm,3,5)`

yields



Implementing the bisection algorithm as a Matlab function

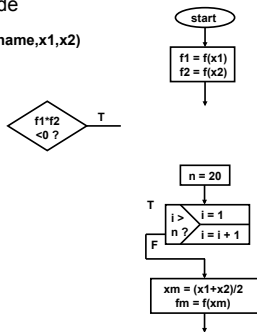


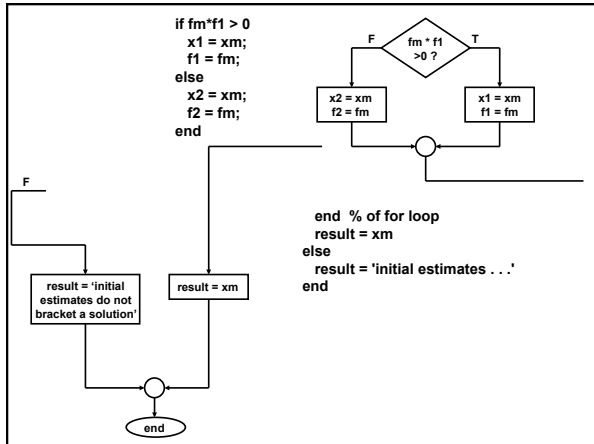
Write the Matlab function code

```
function result = bisect(eqname,x1,x2)
f1 = feval(eqname,x1);
f2 = feval(eqname,x2);
```

```
if f1*f2 < 0
n = 20;
```

```
for i = 1:n
xnew = (x1+x2)/2;
fm = feval(eqname,xm);
```





```

function result = bisect(eqname,x1,x2)    the complete
f1 = feval(eqname,x1);                  bisect function
f2 = feval(eqname,x2);
if f1*f2 < 0
    n = 20;
    for i = 1:n
        xmnew = (x1+x2)/2;
        fm = feval(eqname,xm);
        if fm*f1 > 0
            x1 = xm;
            f1 = fm;
        else
            x2 = xm;
            f2 = fm;
        end
    end
    end % of for loop
    result = xm
else
    result = 'initial estimates do not bracket a solution';
end
  
```

Create **bisect.m** using Matlab Editor

```

function result=bisect (eqname ,x1 ,x2)
f1=feval (eqname ,x1) ;
f2=feval (eqname ,x2) ;
if f1*f2 < 0
    n=20;
    for i=1:n
        xm=(x1+x2) /2;
        fm=feval (eqname , xm) ;
        if fm*f1>0
            x1=xm;
            f1=fm;
        else
            x2=xm;
            f2= fm;
        end
    end
    result = xm;
else
    result = 'initial estimates do not bracket a solution';
end
  
```

Create a function for the equation to be solved

$$f(x) = e^{-\frac{x}{4}}(2-x) - 1$$

```
function y=f(x)
y=exp(-x/4)*(2-x)-1;
```

Test the **bisect** function

```
>> bisect(@f,0.5,1)

ans =

    0.7836
```

and with bad initial estimates


```
>> bisect(@f,0.5,0.7)

ans =


initial estimates do not bracket a solution
```

Using Matlab's debugging tools [very similar to VBA]


set a breakpoint to stop execution of **bisect.m**

 or F12 key

```
function root=bisect(eqname,x1,x2)
● f1=feval(eqname,x1);
f2=feval(eqname,x2);
```

execute the function  it will stop at the breakpoint

```
function root=bisect(eqname,x1,x2)
● f1=feval(eqname,x1);
f2=feval(eqname,x2);
```


single-step the code using F10 or 

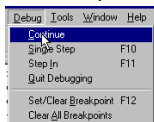
examine variable values with cursor

```
function root=bisect(eqname,x1,x2)
● f1=feval(eqname,x1);
f2=feval(eqname,x2);
▶ f2 = -0.2212
```

There is no "run to cursor" command (like there was in VBA)

You must set an addition breakpoint and run


to that with  or




Note: all debugging commands are also available on Debug menu – not that convenient though

F10 and  will not step "into" a function called by **bisect**.

Use F11 or  to do this.

Clear a single breakpoint by placing cursor on the line and using F12 or 

When done debugging, clear all breakpoints with 

You can quit debugging execution at any time with 