

Fixed Point Method Using Matlab

Huda Alsaud

King Saud University

The main purpose of these slides is to demonstrate how to solve a fixed point problem in MATLAB. The MATLAB program of the fixed point algorithm can be done in various ways. However, since these slides were prepared for students who didn't learn MATLAB before, I tried to present the method as easy as possible by using the command window instead of building M-files. To learn more advanced MATLAB programming and more details about MATLAB we refer to the references [1] and [2].

Outline

- (1) How to use the function *ezplot* to draw a two dimensional graph.
- (2) Create a M-file to calculate Fixed Point iterations.
- (3) Introduction to Newton method with a brief discussion.
- (4) A few useful MATLAB functions.
- (5) Homework.

The function *ezplot*

The function *ezplot* can be used to draw a two dimensional plot in the x -rang from -2π to 2π with the expression as the title. The domain for the function *ezplot* can also be specified, for example, to change the x -axis to the rang 0 to π , it is specified as a vector.

The expression

```
>> ezplot('cos(x)')
```

produces the graph that shown in Figure 1, and

```
>> ezplot('cos(x)',[0 pi])
```

is shown in Figure 2.

How to use the function `ezplot` to draw a two-dimensional graph

Create a M-file to calculate Fixed Point iterations.
Introduction to Newton method with a brief discussion.
A few useful MATLAB functions.

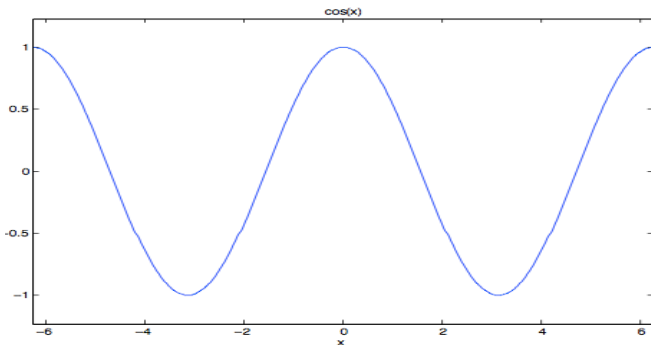


Figure: `ezplot('cos(x)')`

How to use the function *ezplot* to draw a two-dimensional graph

Create a M-file to calculate Fixed Point iterations.
Introduction to Newton method with a brief discussion.
A few useful MATLAB functions.

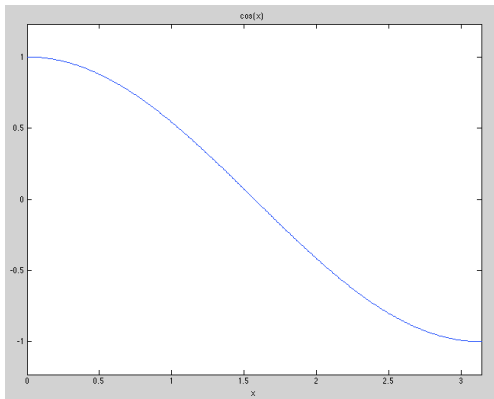


Figure: `ezplot('cos(x)',[0 pi])`

To draw two graphs in one figure window use `hold on` and `hold off`.

For example

```
>> hold on
```

```
>> ezplot('cos(x)',[0 pi])
```

```
>> ezplot('x',[0 pi])
```

```
>> xlabel('x')
```

```
>> ylabel('f(x)')
```

```
>> hold off
```

The result is given in Figure 3.

How to use the function `ezplot` to draw a two-dimensional graph

Create a M-file to calculate Fixed Point iterations.

Introduction to Newton method with a brief discussion.

A few useful MATLAB functions.

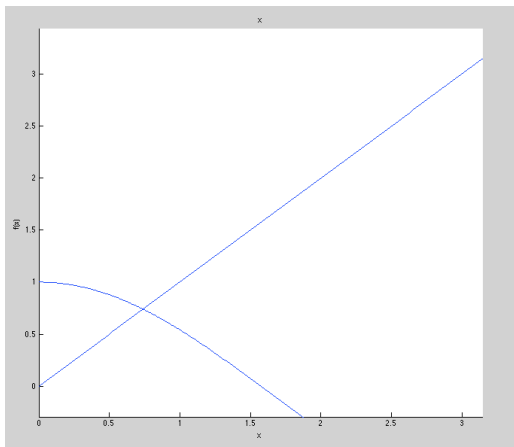


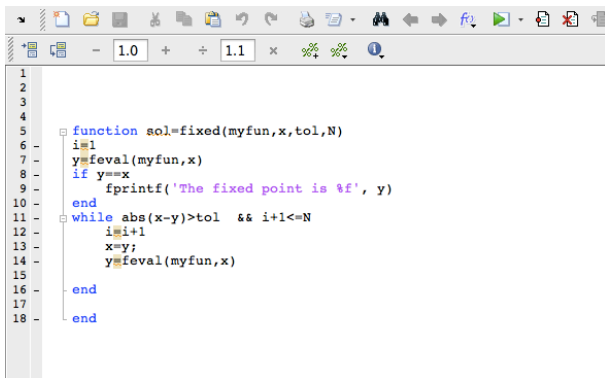
Figure: The graph of $y = \cos x$ and $y = x$

Create a M-file to calculate Fixed Point iterations.

To create a program that calculate fixed point iteration open new M-file and then write a script using Fixed point algorithm. One of the Fixed point program is

```
function sol=fixed(myfun,x,tol,N)           x=y;
i=1                                         y=feval(myfun,x)
y=feval(myfun,x)                           end
if y==x                                     end
fprintf('The fixed point is %f', y)
end
while abs(x-y)>tol && i+1<=N
i=i+1
```

See Figure 4.



The image shows a screenshot of the MATLAB editor interface. At the top, there is a toolbar with various icons for file operations, editing, and execution. Below the toolbar is a numeric keypad with values 1.0 and 1.1. The main area of the editor displays a MATLAB function named 'sol=fixed(myfun,x,tol,N)'. The function code is as follows:

```
1  
2  
3  
4  
5 function sol=fixed(myfun,x,tol,N)  
6     i=1  
7     y=feval(myfun,x)  
8     if y==x  
9         fprintf('The fixed point is %f', y)  
10    end  
11    while abs(x-y)>tol && i+1<=N  
12        i=i+1  
13        x=y;  
14        y=feval(myfun,x)  
15    end  
16 end  
17  
18 end
```

Figure: Fixed point m-file.

In the command window the contents of the script can be executed. To run or execute the script the name of the file is entered at the prompt. Once the script has been executed, you may find that you want to make changes to it especially if there are errors. To run the function for certain initial data call the function. Note that one of the input variables of the function *fixed* is a function. Hence, you need to define the function before you run the program. The simplest way to do this is to use the command window. To define a function in the command window use $@(x)$ between the name of the function and its expression, where x here is the variable.

For sample,

```
>> f=@(x) cos(x)
```

or

```
>> y=@(r) 1-r2
```

To run the function *fixed.m* with one of these functions just call the function with input *f* or *y*.

For example,

```
>> fixed(f,pi,0.0001,7 )
```

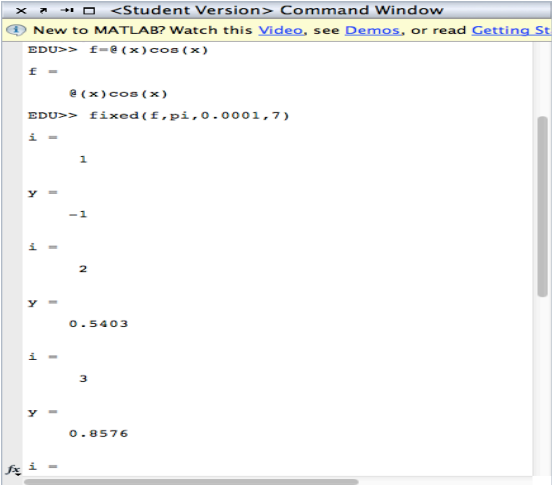
Executing the script produces the following output;

How to use the function *ezplot* to draw a two dimensional graph

Create a M-file to calculate Fixed Point iterations.

Introduction to Newton method with a brief discussion.

A few useful MATLAB functions.



```
<Student Version> Command Window
New to MATLAB? Watch this Video, see Demos, or read Getting St
EDU>> f=@(x)cos(x)
f =
    @(x)cos(x)
EDU>> fixed(f,pi,0.0001,7)
i =
    1
y =
   -1
i =
    2
y =
    0.5403
i =
    3
y =
    0.8576
fx i =
```

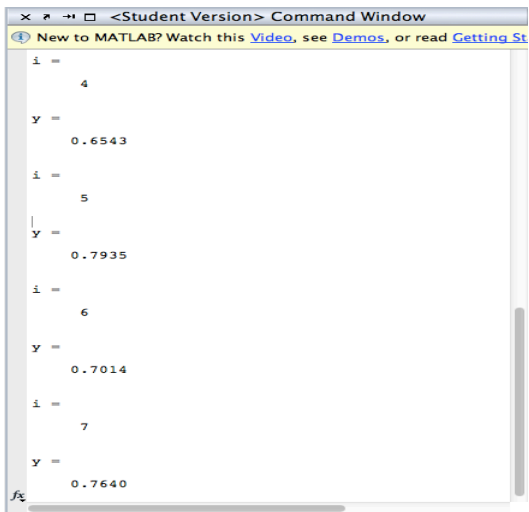
Figure: Result of `fixed(f,pi,0.0001,7)`.

How to use the function *ezplot* to draw a two dimensional graph

Create a M-file to calculate Fixed Point iterations.

Introduction to Newton method with a brief discussion.

A few useful MATLAB functions.



```
<Student Version> Command Window
New to MATLAB? Watch this Video, see Demos, or read Getting St

i =
    4

y =
    0.6543

i =
    5

y =
    0.7935

i =
    6

y =
    0.7014

i =
    7

y =
    0.7640

fx
```

Figure: Result of `fixed(f,pi,0.0001,7)`.

Introduction to Newton method with a brief discussion.

A program for Newton method can be written in similar way. Note that in Newton method we need the derivative of the function. To differentiate the function f use the function $diff(f)$. For example,

```
>>syms f x
```

```
>>f=cos(x)
```

```
>> y=diff(f)
```

```
>>y=-sin(x)
```

Then you need to define f and f' as a functions in the command window,

```
>>f=@(x) cos(x)
```

```
>>df=@(x) -sin(x)
```

Then run your program, for example

```
>>new(f,df,p,tol,N)
```

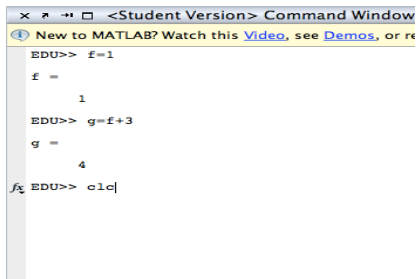
The theorems about Newton's method generally start off with the assumption that the initial guess is "close enough" to the solution. Since you don't know the solution when you start, how do you know when it is "close enough?" The answer is basically that if you stay away from places where the derivative is zero, then any number is OK. More to the point, if you know that the solution lies in some interval and on that interval, then the Newton iteration will converge to the solution, starting from any point in the interval. When there are zeros of the derivative nearby, Newton's method can display highly erratic behavior and may or may not converge.

A few useful MATLAB functions.

`>>clear` : to clear out all variables so they no longer exist.

`>>clc` : delete everything in the command window.

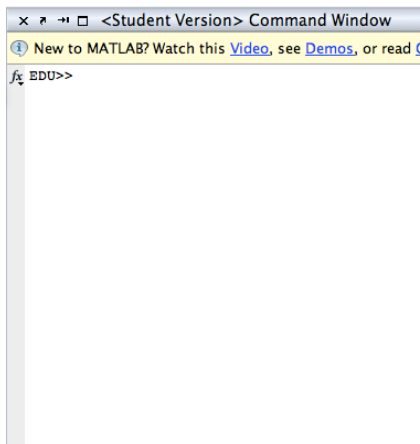
For example,



```
<Student Version> Command Window
New to MATLAB? Watch this Video, see Demos, or re
EDU>> f=1
f =
    1
EDU>> g=f+3
g =
    4
fx EDU>> clc
```

Figure: How to use *clc* function.

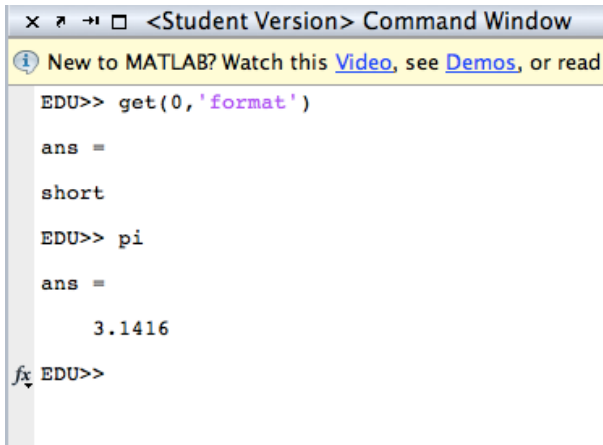
The result will be



```
<Student Version> Command Window
New to MATLAB? Watch this Video, see Demos, or read Help
> cfc
EDU>>
```

Figure: The result of the function *cfc*.

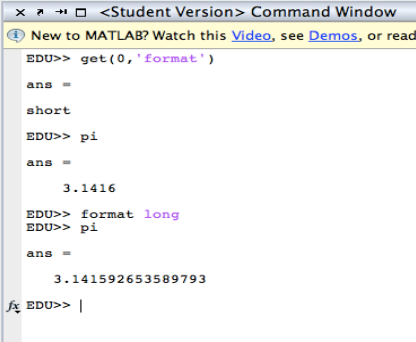
>>get(0,'format') : to see the current format.



```
<Student Version> Command Window
New to MATLAB? Watch this Video, see Demos, or read
EDU>> get(0,'format')
ans =
short
EDU>> pi
ans =
3.1416
fx EDU>>
```

Figure: The current format.

>>format long : to change the format.



```
<Student Version> Command Window
New to MATLAB? Watch this Video, see Demos, or read
EDU>> get(0, 'format')
ans =
short
EDU>> pi
ans =
3.1416
EDU>> format long
EDU>> pi
ans =
3.141592653589793
fx EDU>> |
```

Figure: The new format.



Ctrl + c: To stop an endless loop.

Homework

Q: Writing a Matlab program to approximate a zero of the following functions using Newton method. Approximate the root within 10^{-5} .

1. $f(x) = x^3 - 3x^2 + x - 1$.
2. $f(x) = x^3 - 7$.
3. $f(x) = \sin x - e^{-x}$.

References

-  B.R. Hunt, R.L. Lipsman, and J.M. Rosenberg. A Guide to MATLAB, for beginners and experienced users. Cambridge University Press, 2001.
-  Stormy Attaway. MATLAB: A Practical Introduction to Programming and Problem Solving. Elsevier Inc, 2012.