INTRODUCTION TO SCIENTIFIC & ENGINEERING COMPUTING BIL 108E, CRN24023

Dr. S. Gökhan Karaman

Technical University of Istanbul

March 22, 2010

TENTATIVE SCHEDULE

Week	Date	Topics
1	Feb. 10	Introduction to Scientific and Engineering Computing
2	Feb. 17	Introduction to Program Computing Environment
3	Feb. 24	Variables, Operations and Simple Plot
4	Mar. 03	Algorithms and Logic Operators
5	Mar. 10	Flow Control, Errors and Source of Errors
6	Mar. 17	Functions
6	Mar. 20	Exam 1
7	Mar. 24	Arrays
8	Mar. 31	Solving of Simple Equations
9	Apr. 07	Polynomials Examples
10	Apr. 14	Applications of Curve Fitting
11	Apr. 21	Applications of Interpolation
11	Apr. 24	Exam 2
12	Apr. 28	Applications of Numerical Integration
13	May 05	Symbolic Mathematics
14	May 12	Ordinary Differential Equation (ODE) Solutions with Built-in Functions

LECTURE # 7

LECTURE # 7

LINEAR EQUATIONS cont'd.

1. INVERSE OF A MATRIX
2. DETERMINANT

NONLINEAR EQUATIONS

1. BRACKETING METHODS
   - BISECTION
   - FALSE POSITION(REGULA–FALSI)
2. OPEN METHODS
   - NEWTON METHOD
   - SECANT METHOD
   - FIXED POINT METHOD
3. MATLAB FUNCTIONS

SOME MATRIX FUNCTIONS

SOME MATRIX FUNCTIONS

- `zeros`: creates a matrix that all elements are equal to zero.
- `ones`: creates a matrix that all elements are equal to one.
- `size`: returns the dimension of the matrix.
- `eye`: creates an identity matrix.
- `diag`: creates a diagonal matrix
- `inv`: creates the inverse of a given matrix.
- `trace`: returns the sum of the diagonal terms of a matrix.
- `det`: returns the determinant of a matrix.
- \: left division
- /: right division

LINEAR EQUATIONS

LINEAR EQUATIONS

$a_{11}x_1 + a_{12}x_2 + \ldots + a_{1n}x_n = b_1$
$a_{21}x_1 + a_{22}x_2 + \ldots + a_{2n}x_n = b_2$
$\ldots$
$a_{m1}x_1 + a_{m2}x_2 + \ldots + a_{mn}x_n = b_n$

$$A\,x = b$$

Unknown variables can be calculated with matrix operations.
If $m = n$
$x = A^{-1} \times b$

INVERSE OF A MATRIX

Introduction to Scientific and Engineering Computing, BIL108E

Karaman

INVERSE OF A MATRIX

Inverse of matrix A is A^{-1}.

$$A\,A^{-1} = A^{-1}\,A = I$$

$A\,x = b$

$A^{-1}\,A\,x = A^{-1}\,b$

So, the solution of $A\,x = b$ is

$$x = A^{-1}\,b$$

DETERMINANT OF A MATRIX

Introduction to Scientific and Engineering Computing, BIL108E

Karaman

DETERMINANT OF A MATRIX

$a = |A|$: Determinant of the matrix A.

If the determinant a of a square matrix $A = a_{ij}$ is different from zero, then the inverse matrix A^{-1} of A exists and is obtained by $A^{-1} = \beta_{ij}$
$\beta_{ij} = \frac{\alpha_{ij}}{a}$

Here α_{ij} is the cofactor of a_{ji} in the determinant a of the matrix A.

DETERMINANT OF A MATRIX

Introduction to Scientific and Engineering Computing, BIL108E

Karaman

DETERMINANT OF A MATRIX

$$det(A) = \sum_{j=1}^{n} \alpha_{ij} a_{ij}$$

where $n \geq 1$, $i = 1, \ldots, n$
The $(n-1)$ rowed determinant obtained from the determinant a by striking out the jth row and ith column in a, and then multiplying the result by $(-1)^{i+j}$

$$A = \begin{bmatrix} a_{11} & a_{12} & \ldots & a_{1n} \\ a_{21} & a_{22} & \ldots & a_{2n} \\ \vdots & \vdots & \vdots & \vdots \\ a_{m1} & a_{m2} & \ldots & a_{mn} \end{bmatrix}$$

DETERMINANT OF A MATRIX

$$A = \begin{bmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{bmatrix}$$

For a 2×2 matrix
$det(A) = a_{11}\, a_{22} - a_{12}\, a_{21}$

For a 3×3 matrix
$$det(A) = a_{11}\, a_{22}\, a_{33} + a_{31}\, a_{12}\, a_{23} + a_{21}\, a_{13}\, a_{32}$$
$$-a_{11}\, a_{23}\, a_{32} - a_{21}\, a_{12}\, a13 - a_{31}\, a_{13}\, a_{22}$$

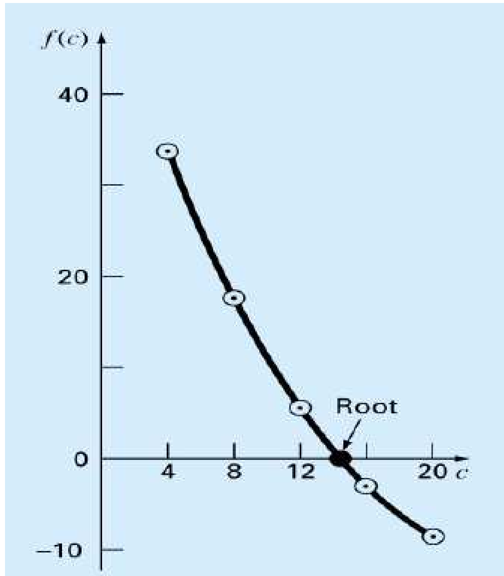
NONLINEAR EQUATIONS

- The Problem: Computing the roots of a real function.
- If the degree of the polynomial is greater than four, there exists no explicit form to obtain the roots.
- If the function is not in the form of a polynomial, finding roots is more difficult.
- Solution: Iterative Methods.
- Start from initial value and converge(hopefully) to a zero value α of the function.

ROOT FINDING

- Nonlinear equations can be written as $f(x) = 0$
- Example: If $f(x) = x\, e^x$, solve $f(x) = x\, e^x = 0$

GRAPHICAL INSPECTION



ROOT FINDING

- Finding the roots of a nonlinear equation is equivalent to finding the values of x for which $f(x)$ is zero.
- Any function of one variable can be put in the form $f(x) = 0$.
- We examine several methods of finding the roots for a general function $f(x)$.

ROOT FINDING

- A fundamental principle in computer science is iteration. As the name suggests, a process is repeated until an answer is achieved.
- Iterative techniques are used to obtain the roots of equations, solutions of linear and nonlinear systems of equations, and solutions of differential equations.
- A rule or function for computing successive terms is needed, together with a starting value.
- Then a sequence of values is obtained using the iterative rule $x_{k+1} = g(x_k)$

EXAMPLE:

To find the x that satisfies $cos(x) = x$

Find the zero crossing of $f(x) = cos(x) - x = 0$

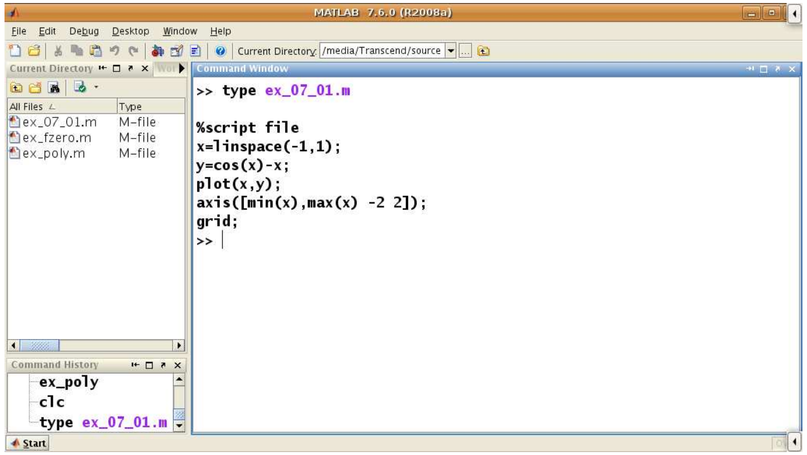
EXAMPLE:

filename: ex_ 07_ 01.m

```
%script file
x=linspace(-1,1);
y=cos(x)-x;
plot(x,y);
axis([min(x),max(x) -2 2]);
grid;
```

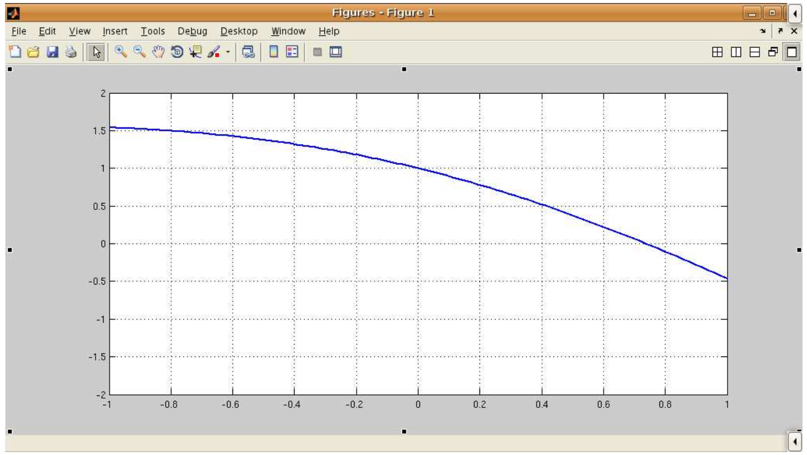
ROOT FINDING

EXAMPLE:



ROOT FINDING

EXAMPLE:



ROOT FINDING

ROOT FINDING

The basic strategy for root-finding procedure

1. **Plot the function.**
   The plot provides an initial guess and an indication of potential problems.
2. **Select an initial guess.**
3. Iteratively refine the initial guess with a root finding algorithm.
   If x_k is the estimate to the root on the k^{th} iteration, then the iterations converge.

NONLINEAR EQUATIONS

METHODS

1. BRACKETING METHODS
   - BISECTION(INTERVAL HALVING)
   - FALSE POSITION(REGULA–FALSI)

   These methods are applied after initial guesses on the root(s) that are identified with bracketing (or guesswork).
2. OPEN METHODS
   - NEWTON METHOD(NEWTON–RAPHSON)
   - SECANT METHOD
   - FIXED POINT METHOD

   These methods may involve one or more initial guesses, however there is no need to bracket the root.

BISECTION METHOD

- f: continuos function within $[a, b]$ which satisfies $f(a)f(b) \leq 0$
- f has at least one zero(α) in (a, b).
- If f has several zeros, use `fplot` command to locate an interval, which contains only one of them.

BISECTION METHOD

- Divide the given interval in halves.
- Select the subinterval, where f features a sign change.
- Intervals named as $I^{(i)}$.
- In each step the interval contains α.

BISECTION METHOD

The method starts by setting:

$a^{(0)} = a, \;\; b^{(0)} = b, \;\; I^{(0)} = (a^{(0)}, \;\; b^{(0)})$

$x^{(0)} = (a^{(0)} + b^{(0)})/2$

At each step $(k \geq 1)$ we select the subinterval

$I^{(k)} = (a^{(k)}, \;\; b^{(k)})$

of the interval $I^{(k-1)} = (a^{(k-1)}, \;\; b^{(k-1)})$

The iteration $(k-1)$, $x^{(k-1)} = (a^{(k-1)}, \;\; b^{(k-1)})/2$ and

if $f(x^{(k-1)}) = 0$ then $\alpha = x^{(k-1)}$

BISECTION METHOD

otherwise

if $f(a^{(k-1)})f(x^{(k-1)}) < 0$ set $a^{(k)} = a^{(k-1)}$, $b^{(k)} = x^{(k-1)}$

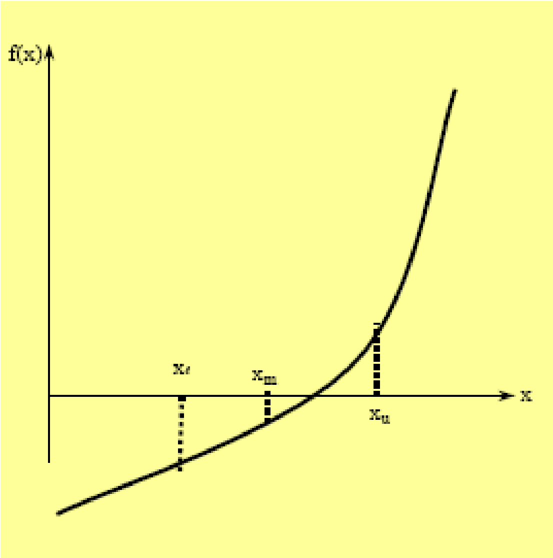
if $f(x^{(k-1)})f(b^{(k-1)}) < 0$ set $b^{(k)} = b^{(k-1)}$, $a^{(k)} = x^{(k-1)}$

Define

$x^{(k)} = (a^{(k)} + b^{(k)})/2$ and interval $I^{(k+1)}$

BISECTION METHOD

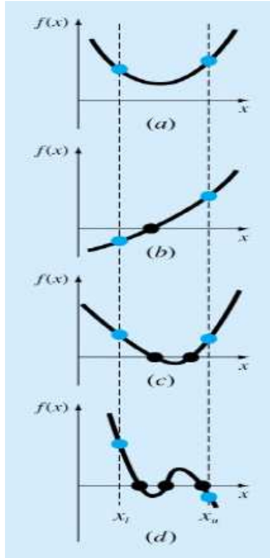
BISECTION METHOD $x^{(k)} = (a^{(k)} + b^{(k)})/2$



Introduction to Scientific and Engineering Computing, BIL108E
Karaman

BISECTION METHOD

BISECTION METHOD $f(a^{(k)})f(b^{(k)}) < 0$



Introduction to Scientific and Engineering Computing, BIL108E
Karaman

BISECTION METHOD

BISECTION METHOD

- Each interval contains the zero α
- The interval halves in each step

$|e^{(k)}| = |x^{(k)} - \alpha| \leq \frac{1}{2}|I^{(k)}| = (\frac{1}{2})^{k+1}(b-a)$
The number of minimum iterations for a given tolerance ϵ:

$k_{min} \geq log_2(\frac{b-a}{\epsilon}) - 1$

$|\epsilon| \geq |\frac{x^{(k+1)} - x^{(k)}}{x^{(k+1)}}|$

Introduction to Scientific and Engineering Computing, BIL108E
Karaman

BISECTION METHOD

BISECTION METHOD

Advantages

- Always convergent.
- The root bracket gets halved with each iteration.

Drawbacks

- Slow convergence.
- If one of the initial guesses is close to the root, the convergence is slower.

In spite of its simplicity, the bisection method does not guarantee a monotone reduction of the error, but simply the search interval is halved from one iteration to the next.

BISECTION METHOD

BISECTION METHOD

```
initialize: a = ..., b = ...
for k = 1, 2, ...
  x_m = a + (b − a)/2
  if sign (f(x_m)) = sign (f(x_a))
    a = x_m
  else
    b = x_m
  end
  if converged, stop
end
```

The statement `eval(f)` is used to evaluate the function at a given value of x.

BISECTION METHOD

EXAMPLE

Hand Calculation Example

Bisection Method

Example : $f(x) = x^2 - 2x - 3 = 0$

initial estimates $[x_a, x_b] = [2.0,\ 3.2]$

iter	x_a	x_b	x_c	$f(x_c)$	Δx
1	2.0	3.2	2.6	–1.44	1.2
2	2.6	3.2	2.9	–0.39	0.6
3	2.9	3.2	3.05	0.2025	0.3
4	2.9	3.05	2.975	–0.0994	0.15
5	2.975	3.05	3.0125	0.0502	0.075
6	2.975	3.0125	2.99375	–0.02496	0.0375

$f(2) = -3,\ f(3.2) = 0.84$

REGULA–FALSI METHOD

REGULA–FALSI METHOD

- From geometry, similar triangles have similar ratios of sides.

$$slope = \frac{f(x_b) - f(x_a)}{x_b - x_a} = \frac{f(x_b) - f(x_c)}{x_b - x_c}$$

- The new approximation for the root: $f(x_r) = 0$
- This can be rearranged to yield Regula–Falsi equation.

$$x_c = x_b - \frac{x_a - x_b}{f(x_a) - f(x_b)} f(x_b)$$

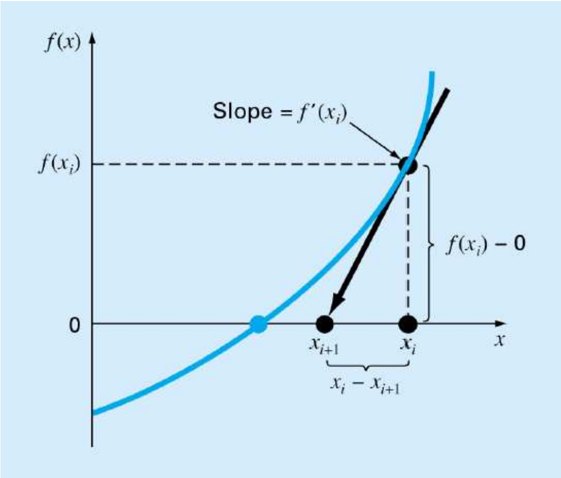
NEWTON METHOD

NEWTON METHOD

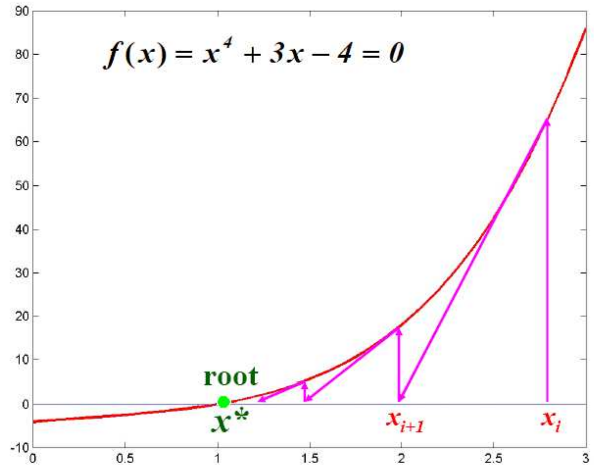
The definition for the derivative is used to find the zero α
$y(x) = f(x^{(k)}) + f'(x^{(k)})(x - x^{(k)})$
with the equation of the tangent to the curve $(x, f(x))$ at the point $x^{(k)}$
$x^{(k+1)} = x^{(k)} - \frac{f(x^{(k)})}{f'(x^{(k)})}$
$f'(x^{(k)}) \neq 0$
This is the simple form of the function f represented in Taylor series.

When the f function is linear it converges in a single step. (Example: $f(x) = a_1 x + a_0$)

NEWTON METHOD



NEWTON METHOD



NEWTON METHOD

The Newton method in general does not converge for all possible choices of $x^{(0)}$, but only for those values of $x^{(0)}$ which are sufficiently close to α. In practice initial value can be obtained:

- with a few iterations of the bisection method or
- with the graph of function f.

NEWTON METHOD

- The error at step $(k+1)$ behaves like the square of the error at step k multiplied by a constant which is independent of k.
- The iterations can be terminated at the smallest value of k_{min} for a given tolerance ϵ

  $|\alpha - x^{(k_{min})}| \leq \epsilon$

  $|x^{k_{min}} - x^{(k_{min}-1)}| \leq \epsilon$

NEWTON METHOD

EXAMPLE:

- Use the Newton Raphson method to determine the mass of the bungee jumper with a drag coefficient of 0.25kg/m to have a velocity of 36m/s after 4s of free fall ($g = 9.81 m/s^2$).
- The function to be evaluated and its derivative is shown below:

$$f(m) = \sqrt{\frac{gm}{c_d}} \tanh\left(\sqrt{\frac{gc_d}{m}} t\right) - v(t)$$

$$\frac{df(m)}{dm} = \frac{1}{2}\sqrt{\frac{g}{mc_d}} \tanh\left(\sqrt{\frac{gc_d}{m}} t\right) - \frac{g}{2m} t \sec h^2\left(\sqrt{\frac{gc_d}{m}} t\right)$$

NEWTON METHOD FOR SYSTEM OF NONLINEAR EQUATIONS

NEWTON METHOD FOR THE SYSTEM OF NONLINEAR EQUATIONS

Consider a system of nonlinear equations of the form

$f_1(x_1, x_2, \ldots, x_n) = 0$

$f_2(x_1, x_2, \ldots, x_n) = 0$

$\ldots$

$f_n(x_1, x_2, \ldots, x_n) = 0$

$\mathbf{f} = (f_1, f_2, \ldots, f_n)^T$

$\mathbf{x} = (x_1, x_2, \ldots, x_n)^T$

$f(x) = 0$

NEWTON METHOD FOR SYSTEM OF NONLINEAR EQUATIONS

NEWTON METHOD FOR THE SYSTEM OF NONLINEAR EQUATIONS

- Extend the Newton's method, replace the first derivative of the scalar function f
  with the Jacobian matrix J_f
  $(J_f)_{ij} = \frac{\partial f_i}{\partial x_j}$
  $i, j = 1, \ldots, n$
- The method stops when the difference between two consecutive iterates has an euclidean norm smaller than ϵ

FIXED POINT ITERATION

FIXED POINT ITERATION

Given a function

$\alpha = \phi(\alpha)$

if such an alpha exist, it is called a fixed point of ϕ

Algorithm:

$x^{(k+1)} = \phi(x^{(k)}),\ k \geq 0$

Fixed point iteration

ϕ Iteration function

Example:

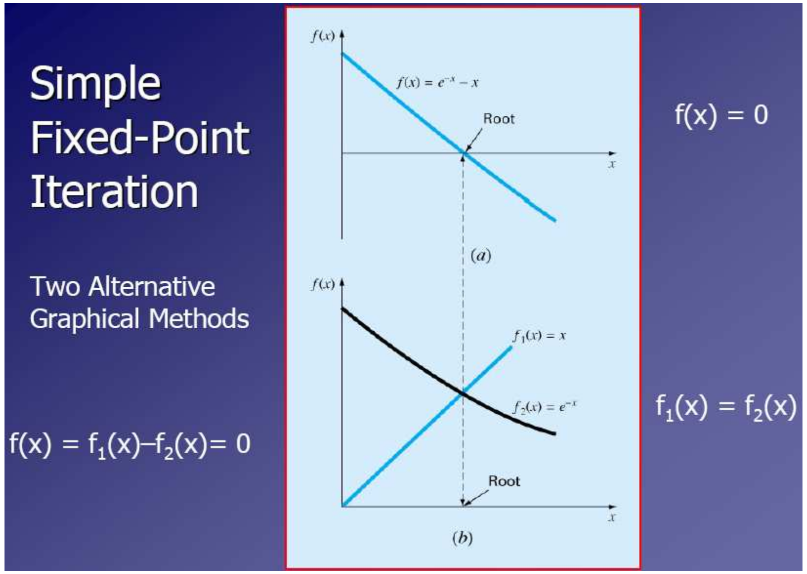
The Newton method can be regarded as an algorithm of fixed point iterations whose iteration function is ϕ_N

$\phi(x) = x - \frac{f(x)}{f'(x)}$

All the functions do not have fixed points.

FIXED POINT ITERATION

EXAMPLE:



Introduction to Scientific and Engineering Computing, BIL108E
Karaman

SECANT METHOD

SECANT METHOD

- Use secant line instead of tangent line at $f(x_i)$
- The formula for the secant method is

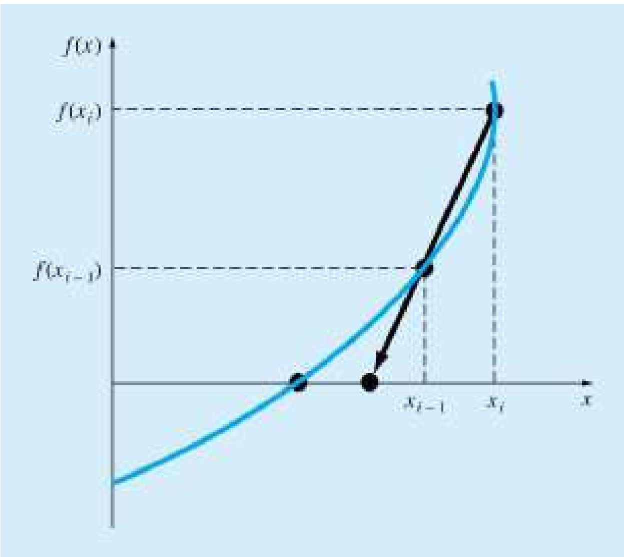
$$x_{i+1} = x_i - \frac{f(x_i)(x_{i-1} - x_i)}{f(x_{i-1}) - f(x_i)}$$

- Notice that this is very similar to the False Position(Regula Falsi) method in form.
- Still requires two initial estimates
- But it does not bracket the root at all times - there is no sign test.

Introduction to Scientific and Engineering Computing, BIL108E
Karaman

SECANT METHOD

SECANT METHOD



Introduction to Scientific and Engineering Computing, BIL108E
Karaman

MATLAB FUNCTION `fzero`

MATLAB FUNCTION `fzero`

Solution with Dekker –Brent method.

- Bracketing methods: reliable but slow.
- Open methods: fast but possibly unreliable.
- MATLAB fzero: fast and reliable.
- fzero: find real root of an equation (not suitable for double root).
- When output argument `flag` is negative it means that, `fzero` cannot find the zero.

`fzero`(*function*, x_0)
`fzero`(*function*, $[x_0 x_1]$)

MATLAB FUNCTION fzero

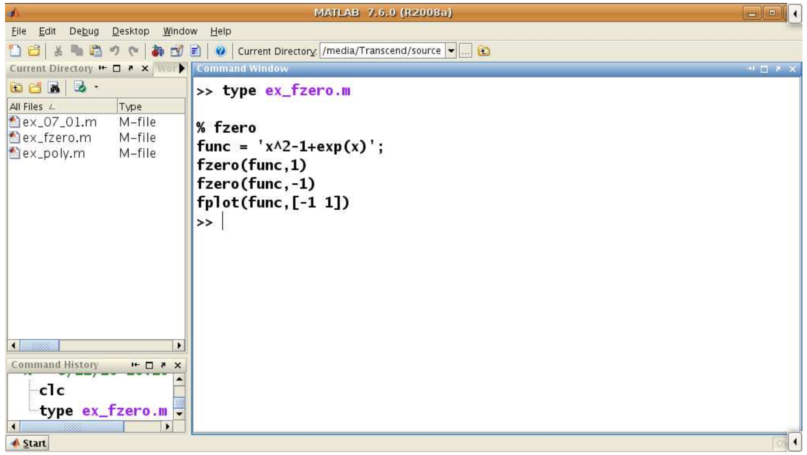
EXAMPLE:

filename: ex_ fzero.m

```
% fzero
func = 'x^2-1+exp(x)';
fzero(func,1)
fzero(func,-1)
fplot(func,[-1 1])
```

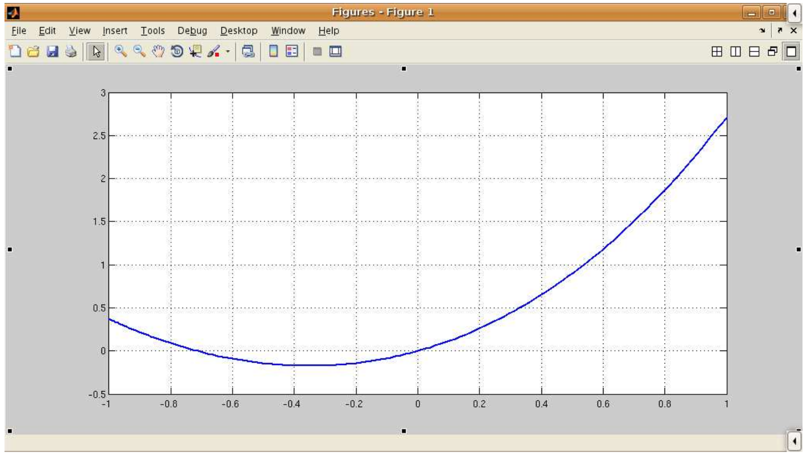
MATLAB FUNCTION fzero

EXAMPLE:



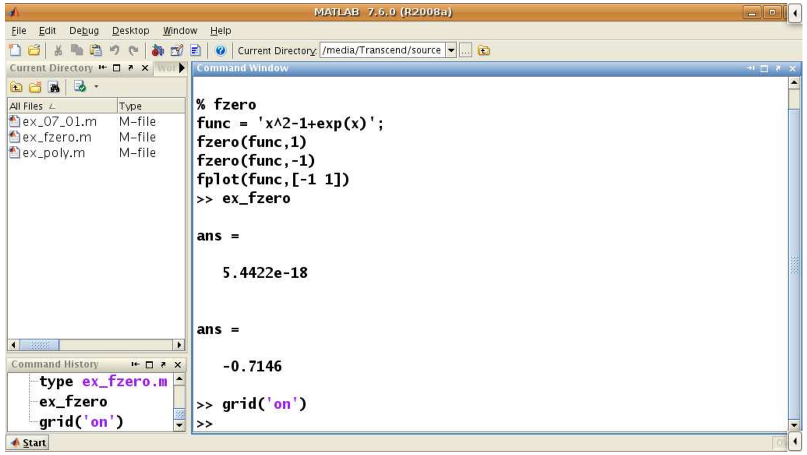
MATLAB FUNCTION fzero

EXAMPLE:



MATLAB FUNCTION fzero

EXAMPLE:



MATLAB FUNCTION `roots`

- Zeros of n^{th} – order polynomial
  $p(x) = c_n\,x^n + c_{n-1}\,x^{n-1} + \ldots + c_2\,x^2 + c_1\,x + c_0$
- Coefficient vector $c = [c_n,\, c_{n-1}, \ldots,\, c_2,\, c_1,\, c_0]$

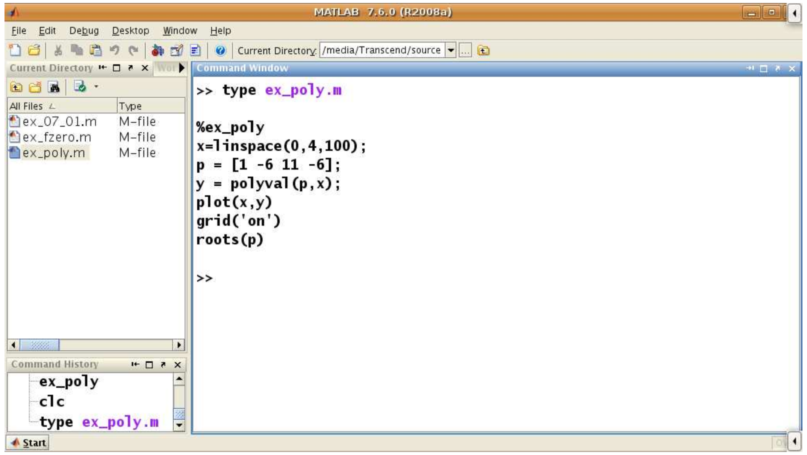
```
c = poly(r)
x = roots(c)
```


EXAMPLE:

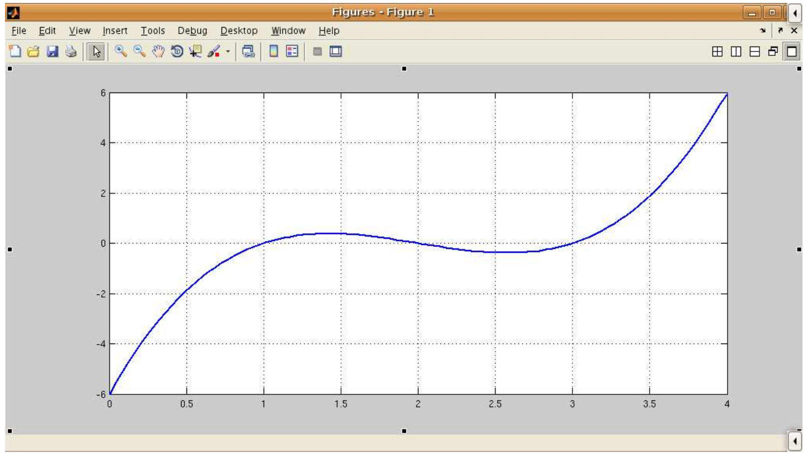
filename: ex_ fzero.m

```
%ex_poly
x=linspace(0,4,100);
p = [1 -6 11 -6];
y = polyval(p,x);
plot(x,y)
grid('on')
roots(p)
```


EXAMPLE:

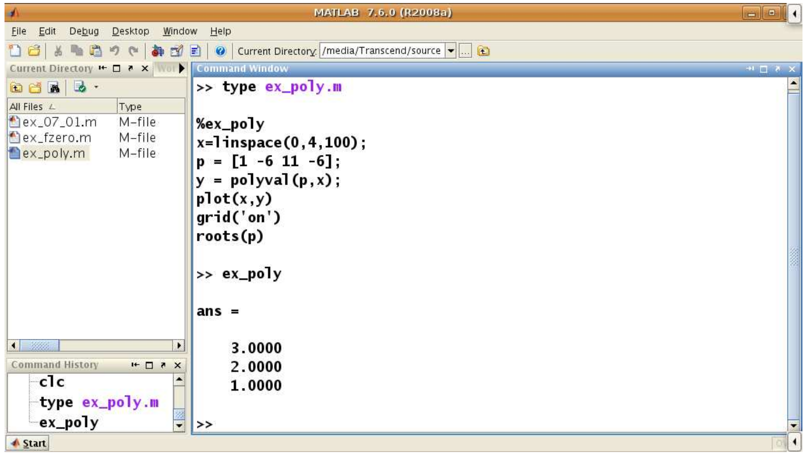


EXAMPLE:



MATLAB FUNCTION roots

EXAMPLE:



BISECTION METHOD

SOURCE:

```
function [zero ,res ,niter ]= bisection(fun ,a,b,tol ,...
                                        nmax ,varargin)
%BISECTION Find function zeros.
% ZERO=BISECTION(FUN ,A,B,TOL ,NMAX) tries to find a zero
% ZERO of the continuous function FUN in the interval
% [A,B] using the bisection method. FUN accepts real
% scalar input x and returns a real scalar value. If
% the search fails an errore message is displayed. FUN
% can also be an inline object.
```

BISECTION METHOD

SOURCE cont'd.:

```
% ZERO=BISECTION(FUN ,A,B,TOL ,NMAX ,P1 ,P2 ,...) passes
% parameters P1 ,P2 ,... to the function FUN(X,P1 ,P2 ,...
% [ZERO ,RES ,NITER ]= BISECTION(FUN ,...) returns the val
% of the residual in ZERO and the iteration number at
% which ZERO was computed.
x = [a, (a+b)*0.5 , b]; fx = feval(fun ,x,varargin {:});
if fx (1)*fx(3) > 0
    error ([' The sign of the function at the ' ,...
            'endpoints of the interval must be different '
elseif fx(1) == 0
    zero = a; res = 0; niter = 0; return
elseif fx(3) == 0
    zero = b; res = 0; niter = 0; return
end
```

BISECTION METHOD

SOURCE cont'd.:

```
niter = 0;
I = (b - a)*0.5;
while I >= tol & niter <= nmax
    niter = niter + 1;
    if fx (1)* fx(2) < 0
        x(3) = x(2); x(2) = x(1)+(x(3)-x(1))*0.5;
        fx = feval(fun ,x,varargin {:}); I = (x(3)-x(1))*0
    elseif fx (2)* fx(3) < 0
        x(1) = x(2); x(2) = x(1)+(x(3)-x(1))*0.5;
        fx = feval(fun ,x,varargin {:}); I = (x(3)-x(1))*0
    else
        x(2) = x(find(fx ==0)); I = 0;
    end
end
```

BISECTION METHOD

SOURCE cont'd.:

```
if niter > nmax
    fprintf(['bisection stopped without converging ' ,...
             'to the desired tolerance because the ' ,...
             'maximum number of iterations was ' ,...
             'reached\n']);
end
zero = x(2); x = x(2); res = feval(fun ,x,varargin {:});
return
```

NEWTON METHOD

SOURCE:

```
function [zero ,res ,niter ]= newton(fun ,dfun ,x0 ,tol ,..
                                    nmax ,varargin)
%NEWTON Find function zeros.
% ZERO=NEWTON(FUN ,DFUN ,X0 ,TOL ,NMAX) tries to find the
% zero ZERO of the continuous and differentiable
% function FUN nearest to X0 using the Newton method.
% FUN and its derivative DFUN accept real scalar input
% x and returns a real scalar value. If the search fails
% an errore message is displayed. FUN and DFUN can also
% be inline objects.
```

NEWTON METHOD

SOURCE cont'd.:

```
% ZERO=NEWTON(FUN ,DFUN ,X0 ,TOL ,NMAX ,P1 ,P2 ,...) passe
% parameters P1 ,P2 ,... to functions: FUN(X,P1 ,P2 ,...)
% and DFUN(X,P1 ,P2 ,...).
% [ZERO ,RES ,NITER ]= NEWTON(FUN ,...) returns the value
% the residual in ZERO and the iteration number at which
% ZERO was computed.
```

NEWTON METHOD

SOURCE cont'd.:

```
x = x0;
fx = feval(fun ,x,varargin {:});
dfx = feval(dfun ,x,varargin {:});
niter = 0; diff = tol +1;
while diff >= tol & niter <= nmax
    niter = niter + 1; diff = - fx/dfx;
    x = x + diff; diff = abs(diff );
    fx = feval(fun ,x,varargin {:});
    dfx = feval(dfun ,x,varargin {:});
end
```

SOURCE cont'd.:

```
if niter > nmax
    fprintf (['newton stopped without converging to ' ,...
              'the desired tolerance because the maximum '
              'number of iterations was reached\n']);
end
zero = x; res = fx;
return
```

References

Introduction to Scientific and Engineering Computing, BIL108E

Karaman

References for Week 7

1 Alfio Quarteroni, Fausto Saleri, Wissenschaftliches Rechnen mit Matlab, Springer, 2006.