Roots of polynomials and transcendental equations (Root finding)

- Equation solving is one of the most basic problems in scientific computing.
- Roots of equations frequently occur in the area of engineering design.
- For example,

  Medical studies have established that a free falling Jumper's chances of sustaining an injury increases significantly if the free-falling velocity exceeds 36 m/s after 4 s of free fall. We want to determine the mass at which this criteria is exceeded at a given drag coefficient of 0.25.

We know the analytical solution of the free fall velocity as a function of time is given by,

$$v(t) = \sqrt{\frac{mg}{c_d}}\tanh\left(\sqrt{\frac{gc_d}{m}}\,t\right)$$

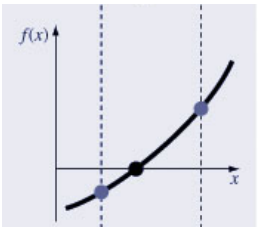
$$\Rightarrow \sqrt{\frac{mg}{c_d}}\tanh\left(\sqrt{\frac{gc_d}{m}}\,t\right) - v(t) = 0$$

Now, we see that the answer of the problem is the value of m that makes f(m)=0 i.e. the Root of the eqation.

1

Graphical method

- A simple method for obtaining the estimate of the root of the equation *f(x)*=0 is
  - plot of the function and
  - observe where it crosses the *x* -axis.



2

Graphical method

Example:

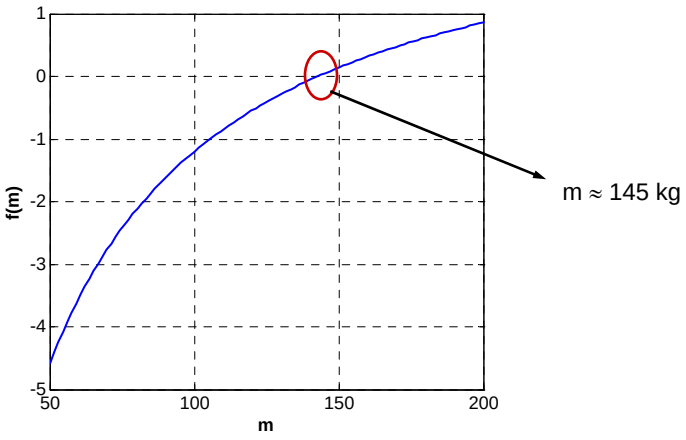
Use graphical approach to determine the mass of a Jumper with a drag coefficient of 0.25 kg/m to have a velocity of 36 m/s after 4 s of free fall. g=9.8 m/s^2 .

MATLAB code:

```
format long e; clear all;close all;
cd=0.25;  % drag coefficient
g=9.81;   % gravity in m/s2
v=36;     % velocity in m/s
t=4;      % time in s
mp=linspace(50,200);
fp=sqrt(g*mp/cd).*tanh(sqrt(g*cd./mp)*t)-v;
plot(mp,fp,'Linewidth',2);
grid;
```

3

Graphical method



Graphical method provides a rough approximation of the root.

4

Roots of polynomials and transcendental equations (Root finding)

Two major classes of methods available are distinguished by the initial guess.

1) Bracketing methods:
As the name implies, they are based on two initial guesses that "bracket" the root, That is, they are on either side of the root.

2) Open methods:
These methods can involve one or more initial guess but there is no need for them to bracket the root.

For well-posed problems, the bracketing methods always work but converge Slowly (i.e. they typically take more number of iterations)

In contrast, the open methods do not work always (i.e. they can diverge), but when they do they usually converge quickly.

5

Bisection method

- Bracketing method

Theorem

Let f be a continuous function on [a, b], satisfying $f(a)*f(b)<0$.

Then f has a root between a and b, that is, there exists a number r satisfying $a<r<b$ and $f(r)=0$.

6

Bisection method

Definition

A solution is correct with in P decimal places (significant figures) if

the absolute error is less than $0.5 * 10^{-P}$

Or, the relative error is less than $0.5 * 10^{2-P}$ %

7

Bisection method

How accurate and how fast?

Solution error

Number of function evaluations

Number of steps needed to get the desired accuracy

8

Bisection method

```
a=0;b=1;
TOL=0.5e-3;
f=inline('x^3+x-1');
[inter_number xc]=Bisect(f,a,b,TOL);
```

i	ai	f(ai)	ci	f(ci)	bi	f(bi)
0	0.00000000	-1.00000000	0.50000000	-0.37500000	1.00000000	1.00000000
1	0.50000000	-0.37500000	0.75000000	-0.17187500	1.00000000	1.00000000
2	0.50000000	-0.37500000	0.62500000	-0.13085938	0.75000000	0.17187500
3	0.62500000	-0.13085938	0.68750000	0.01245117	0.75000000	0.17187500
4	0.62500000	-0.13085938	0.65625000	-0.06112671	0.68750000	0.01245117
5	0.65625000	-0.06112671	0.67187500	-0.02482986	0.68750000	0.01245117
6	0.67187500	-0.02482986	0.67968750	-0.00631380	0.68750000	0.01245117
7	0.67968750	-0.00631380	0.68359375	0.00303739	0.68750000	0.01245117
8	0.67968750	-0.00631380	0.68164063	-0.00164600	0.68359375	0.00303739
9	0.68164063	-0.00164600	0.68261719	0.00069374	0.68359375	0.00303739

0.68212891

9

Bisection method

```
function [iter_number xc]=Bisect(f,a,b,TOL)
fa=f(a); fb=f(b);
if sign(fa)*sign(fb)>=0
    error('Root is NOT bracketed\n'); % Program STOPs
end

iter_number=0;
while(b-a)/2>TOL
    c=(a+b)/2;
    fc=f(c);
    if fc==0
        break         % c is a solution, done
    end
    if sign(fc)*sign(fa)<0
        b=c;                % b and c make the new interval
        fb=fc;
    else
        a=c;                 % a and c make the new interval
        fa=fc;
    end
    iter_number=iter_number+1;

end
xc=(a+b)/2;    % final ROOT (after coming out of the while loop).
```

10