Robot Learning
2. Numerical Method

Jeong-Yean Yang

2020/10/22

Dept. of Intelligent Robot Eng. MU

0

Numerical Method

1. Equation Solver
2. Multiple Nonlinear Equation
3. Linear Regression
4. Nonlinear Regression
5. Stochastic Regression (RANSAC)
6. Optimization

1

Deterministic Vs. Stochastic Method

First Start with Numerical Methods

Deterministic Vs. Stochastic World

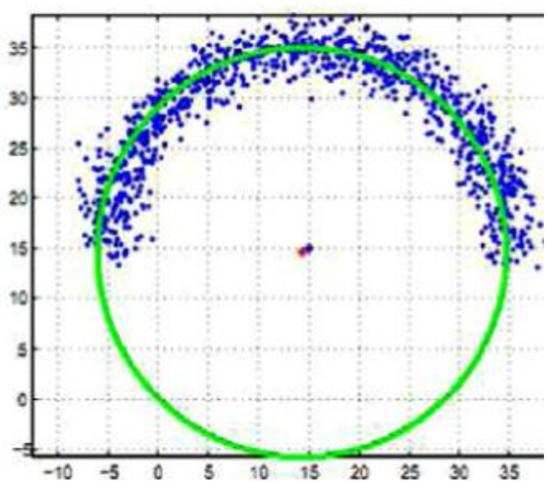
- Deterministic World
  - Everything must be determined.
  - Everything is Understood by Modeling
  - Manipulator-based Robotics, PID or even Robust Control
  - Pseudo code : $a = 3$
- Stochastic World
  - Everything is PROBABILISTICALLY determined
  - Every phenomenon occurs by Probabilistic Results
  - Autonomous locomotion(SLAM), Learning, and so on
  - Pseudo code:

$$a \sim N(3, \sigma^2) = \frac{1}{\sqrt{2\pi\sigma}} \exp \frac{-1}{2} \left(\frac{x-3}{\sigma} \right)^2$$

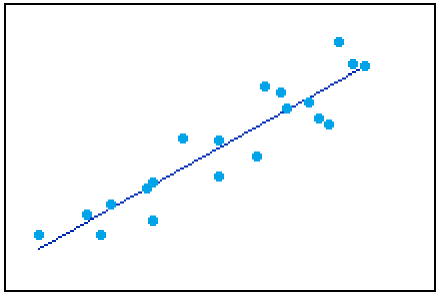
Equality "=" is
NOT allowed.

Why we learn first Deterministic Method?

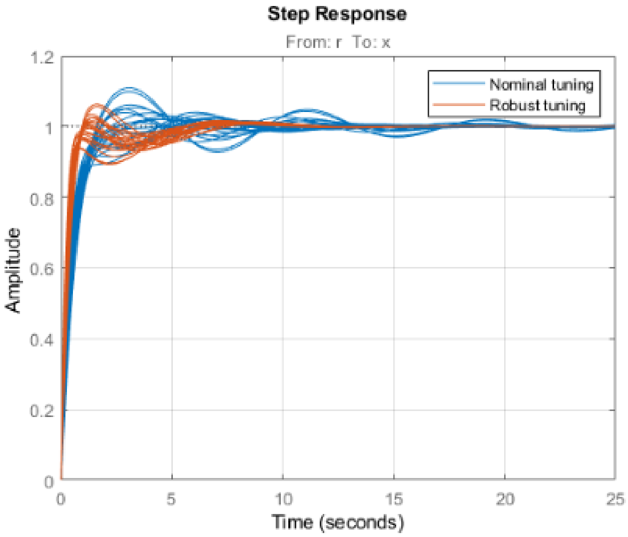
- Some methods look like Non-Deterministic Problem.
- Fitting, Regression, Control



Regression



Curve fitting



Control

- Ex) Control tries to be in the desired goal in spite of unnecessary system dynamics, Is it probabilistic?
- Absolutely, Not.

Many Deterministic Methods are based on Mathematical Model

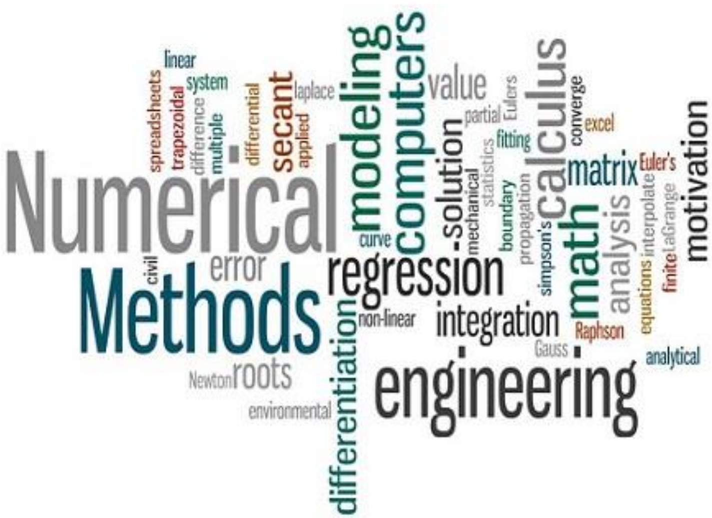
- Controller is well designed to OVERCOME marginal error → Deterministic method
- Fitting or Regression is to Minimize Errors → Deterministic Method

- Then, What is Stochastic Process?
  – Probabilistically, next state is determined.
- 1.Many Deterministic Methods are applied to Learning Methods
- 2. We learn the differences of

Deterministic and Stochastic Methods.

2

Numerical Methods for Solving Equation, f(x)=0



Numerical Method: The goal is to find the solution

- Analytical Solution

$$2a = 3$$

$$\therefore a = 1.5$$

- Numerical Solution(or Computational Solution)
- Solve 2a=3 equation by using computer program

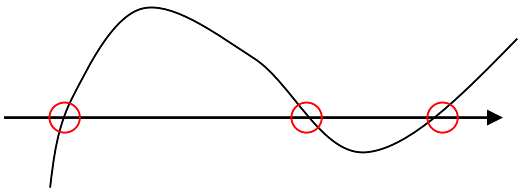
$$Equation : 2a = 3$$

$$Function : f(a) = 2a - 3$$

$$How\ to\ find\ f(a) = 2a - 3 = 0\ ?$$

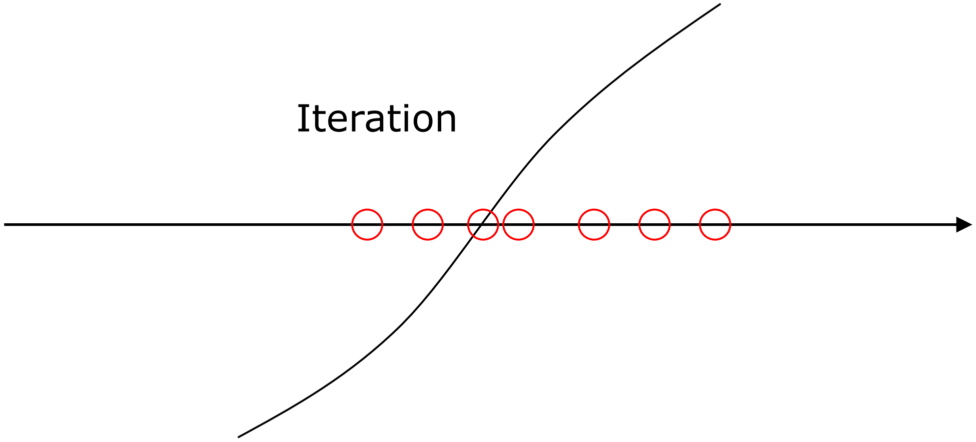
Numerical Method
Solve Equation(or Finding Roots)

- Solve Equation
  - Equation: f(x,y,s,t) = 0



- How to solve it by Numerical Method?

- Iteratively, find a solution by a Computer



Why Numerical Methods are Required?

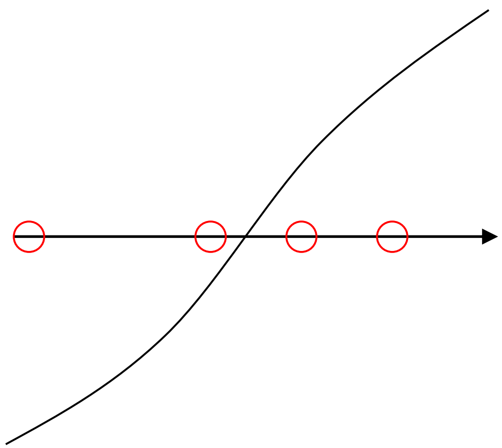
- 1. Equations are Complex
  - Remind Robot Kinematics or Dynamics are very complex
  - Generally, we CANNOT solve it by analytical methods

$$\begin{pmatrix} m_1 l_1^2 + m_2 l_1^2 + m_2 l_2^2 + 2 m_2 l_1 l_2 c_2 & m_2 l_2^2 + m_2 l_1 l_2 c_2 \\ m_2 l_2^2 + m_2 l_1 l_2 c_2 & m_2 l_2^2 \end{pmatrix} \begin{pmatrix} \ddot{\theta}_1 \\ \ddot{\theta}_2 \end{pmatrix} + \begin{pmatrix} -m_2 l_1 l_2 (2\dot{\theta}_1 + \dot{\theta}_2) s_2 \dot{\theta}_2 \\ m_2 l_1 l_2 \dot{\theta}_1^2 s_2 \end{pmatrix} + g \begin{pmatrix} m_1 l_1 c_1 + m_2 (l_2 c_{12} + l_1 c_1) \\ m_2 l_2 c_{12} \end{pmatrix} = 0$$

- 2. We learn Convergence by Iterative Methods
  - <u>Iteration: In Each turn, **a method moves to the solution**</u>
  - Convergence or Divergence Problem
  - **Dynamic Programming**: Learning, Control, Numerical Methods, etc.

Numerical Method: 1. Bisection Method

- See test1.m



$f(x)$

Algorithm

1. Start with two points

Assume

$$x_L < x_s < x_R$$

$$f(x_L)f(x_R) < 0$$

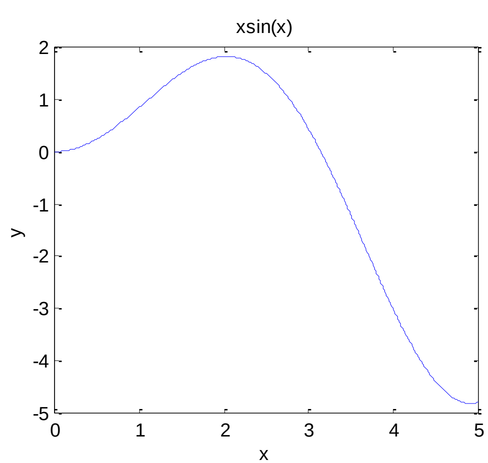
2. Find mid point

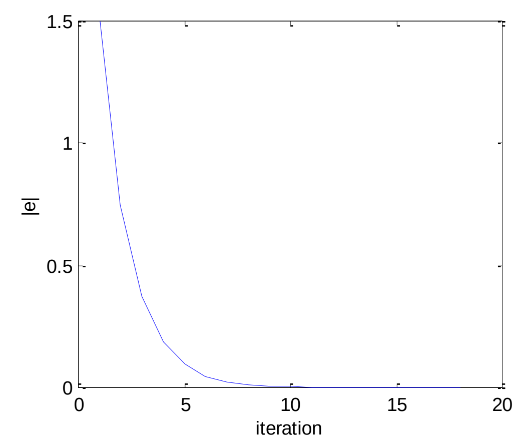
$$x_m = \frac{x_L + x_R}{2}$$

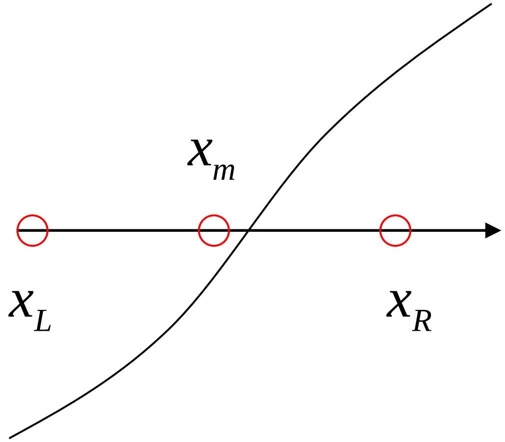
$$f(x_m)f(x_L) < 0 : x_m \rightarrow x_R{}'$$

$$f(x_m)f(x_R) < 0 : x_m \rightarrow x_L{}'$$

$$x \sin x = 0$$



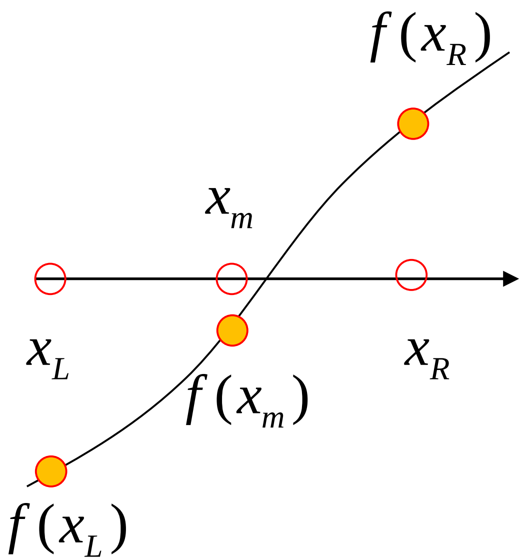




1st turn

$$x_m = \frac{x_L + x_R}{2}$$

Question: Xm is left or Right?



Solution, X is always between
$X_L < X < X_R$

$$\therefore f(x_L)f(x_R) < 0$$

Thus, Xm will be left or right

$$\text{if } f(x_L)f(x_m) < 0, \text{ new } x_R = x_m$$

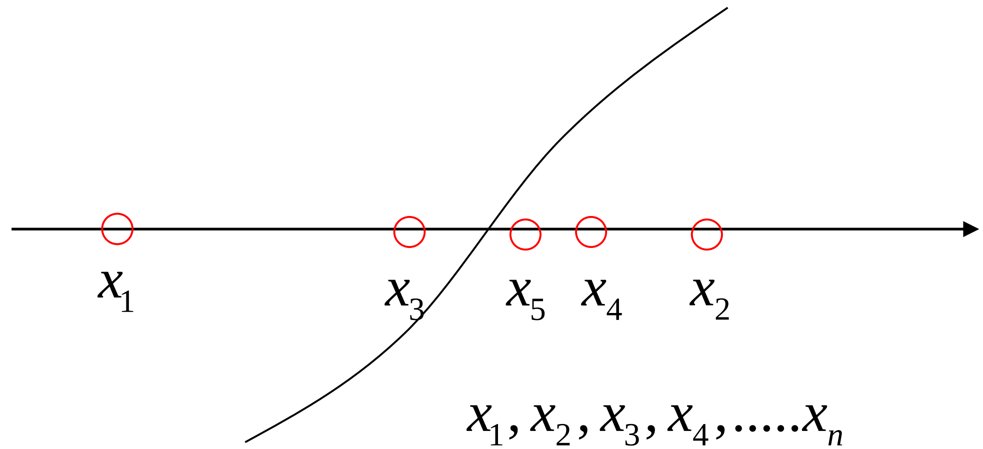
$$\text{if } f(x_R)f(x_m) < 0, \text{ new } x_L = x_m$$

$$x_L \leftarrow x_m$$

Convergence

- Modeling it as you have Learned in other classes

$$x_m = \frac{x_L + x_R}{2}$$



$$x_1, x_2, x_3, x_4, \ldots\ldots x_n$$

$$\rightarrow |\mathrm{e}| \square |x_L - x_R|$$

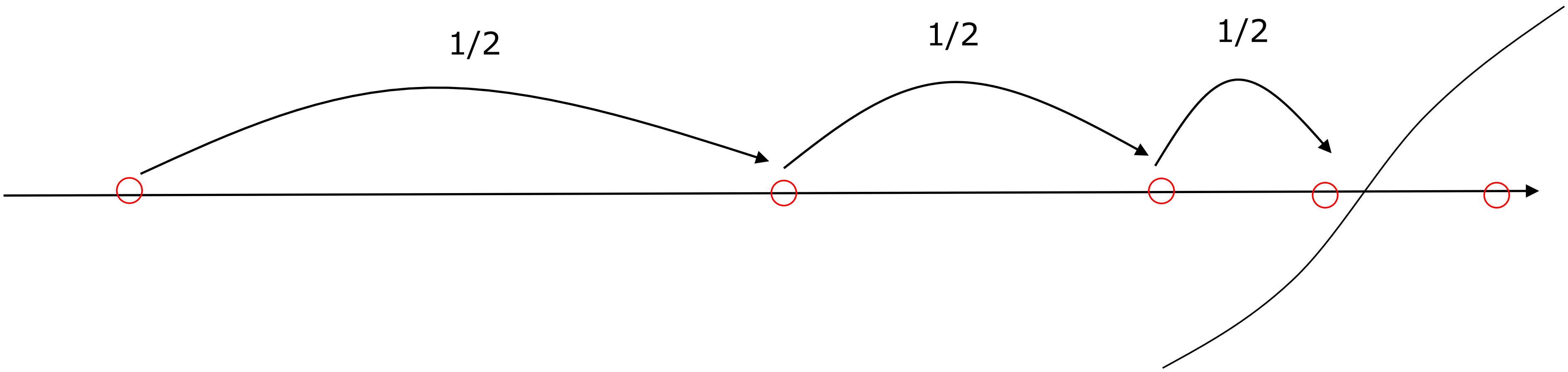
$$|\mathrm{e}_1| = |x_2 - x_1|$$
$$|\mathrm{e}_2| = |x_3 - x_2|$$
$$\ldots$$
$$\therefore |x_3 - x_2| \le \frac{|x_2 - x_1|}{2}$$

$$|e_n| = |x_{n+1} - x_n| \le \frac{|x_n - x_{n-1}|}{2} \le \frac{|x_{n-1} - x_{n-2}|}{4} \ldots \le \frac{|x_2 - x_1|}{2^{n-1}}$$

$$\lim_{n\to\infty} |e_n| \le \lim_{n\to\infty} \frac{|x_2 - x_1|}{2^{n-1}} = |x_2 - x_1| \lim_{n\to\infty} \frac{1}{2^{n-1}} = 0$$

$$\therefore \lim_{n\to\infty} |e_n| \le 0$$

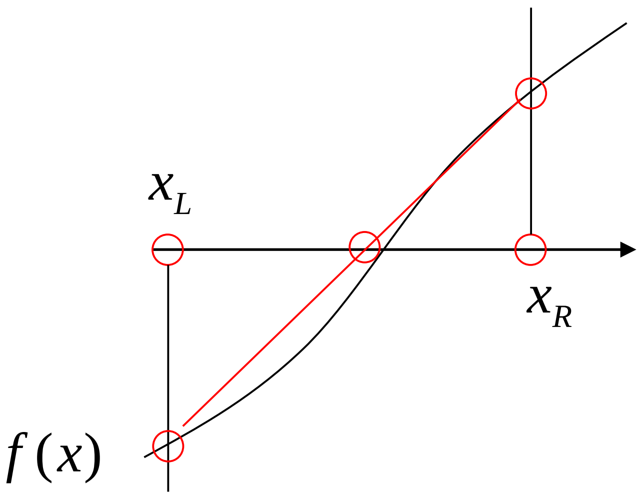
Bisection Method is Too Slow



- If the initial X_L or X_R is too far from a solution, ½ is NOT so Fast!

- How can we speed up?
  - Ratio of a Function is better than 1/2 .

Numerical Method: 2.Secant Method



Algorithm

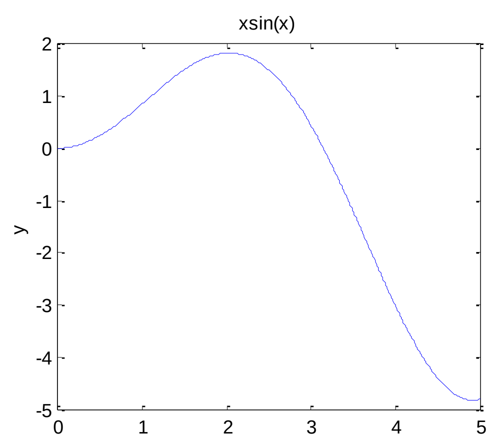
1. Start with two points

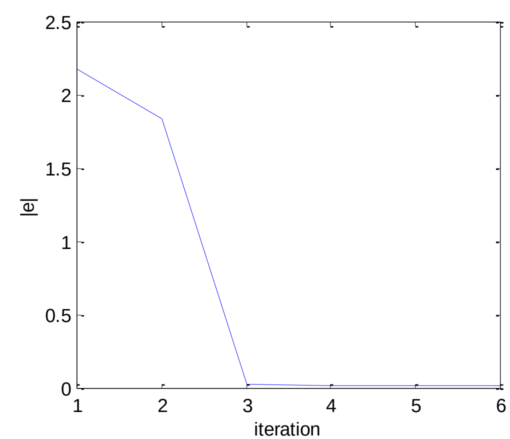
$$y = \frac{f(x_R) - f(x_L)}{x_R - x_L}(x - x_L) + f(x_L) = 0$$

2. Find mid point

$$\frac{f(x_R) - f(x_L)}{x_R - x_L}(x_m - x_L) + f(x_L) = 0$$

- See test2.m



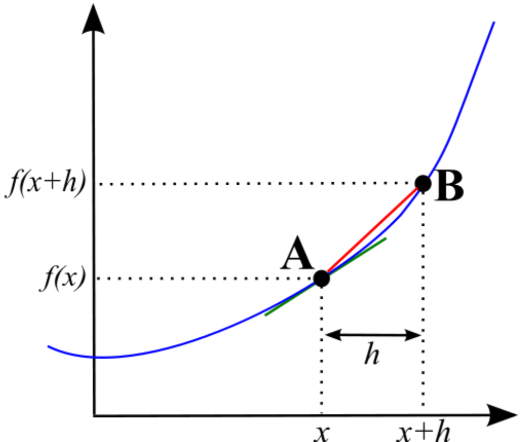


Why Secant Method is Faster than Bisection Method?

- Focus on a new Mid point.

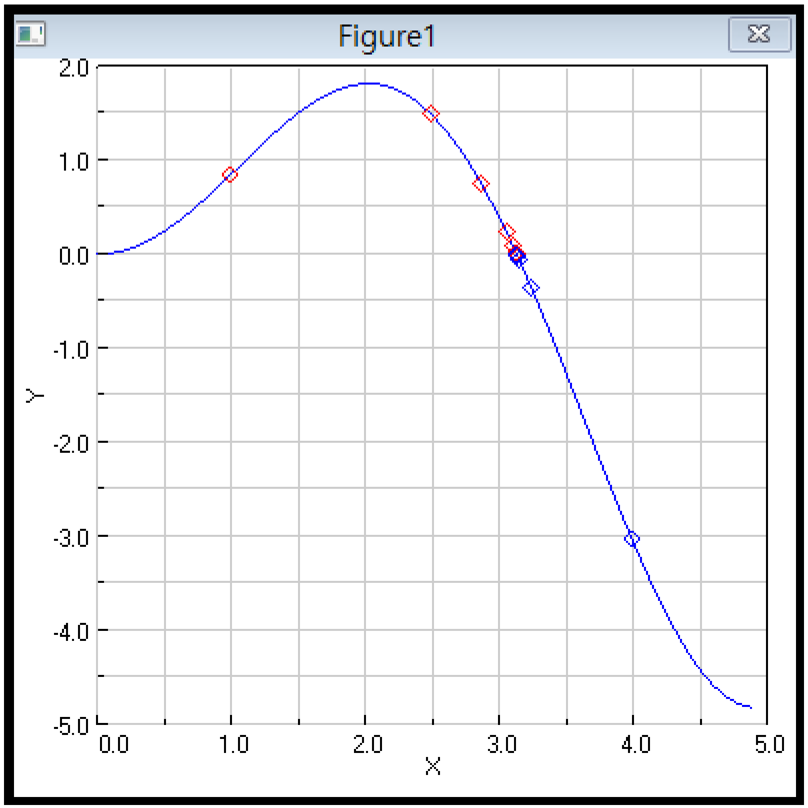
$$y = \boxed{\frac{f(x_R) - f(x_L)}{x_R - x_L}}(x - x_L) + f(x_L) = 0$$

- Function Ratio is good for faster convergence.
- What does the Function Ratio remind us of?

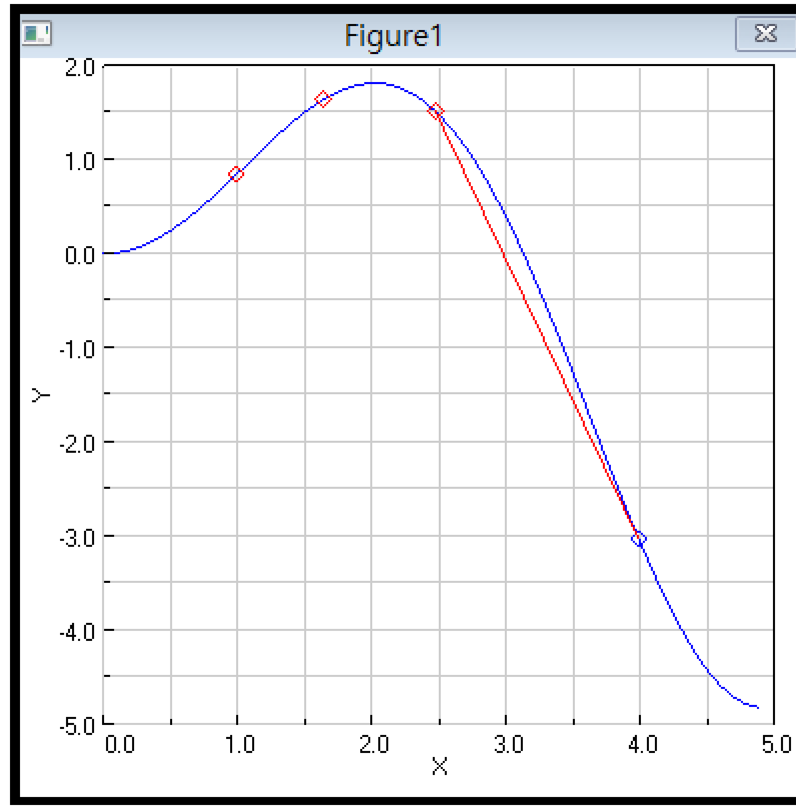


Differentiation!

Example) l2bisect and l2secant ex/ml/l2bisect and ex/ml/l2secant



import l2bisect
l2bisect.test(1,4)



import l2secant
l2secant.test(1,4)

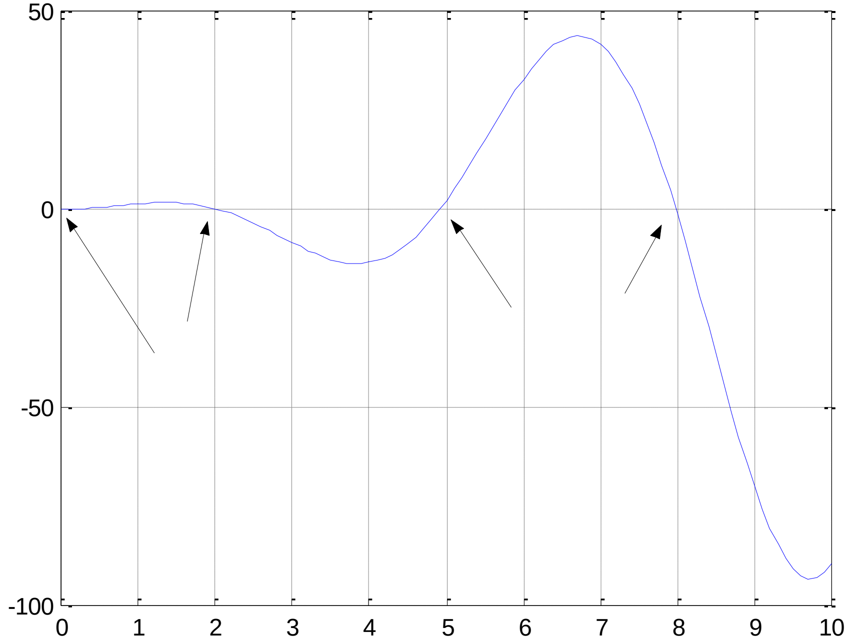
ex/ml/l2bisect and ex/ml/l2secant

```
def test(xl,xr):
    eold    = 0;
    drawfunc()

    for i in range(0,100):
        fl  = func(xl)
        fr  = func(xr)

        if (fl*fr>=0):
            print("Wrong initial guess.");
            return

        # New mid
        xm  = (xl+xr)/2;
        fm  = func(xm)

        # print data
        loop.io.print(i,":",xl,xr,"xm=",xm)

        # draw xl and xr
        graph(2)
        plot(xl,fl,"rd")
        graph(3)
        plot(xr,fr,"bd")

        # find New xl or xr
        if (fl*fm<0):   # xl and xm as new guesses
            xr  = xm
        elif (fr*fm<0): # xm and xr as new guesses
            xl  = xm

        error   = abs(xl-xr)

        # stop or not
        if (abs(error-eold)<1e-5):
            break;

        eold    = error
        loop.pause()

    print("XL=",xl,"XR=",xr, "Error",error)
```

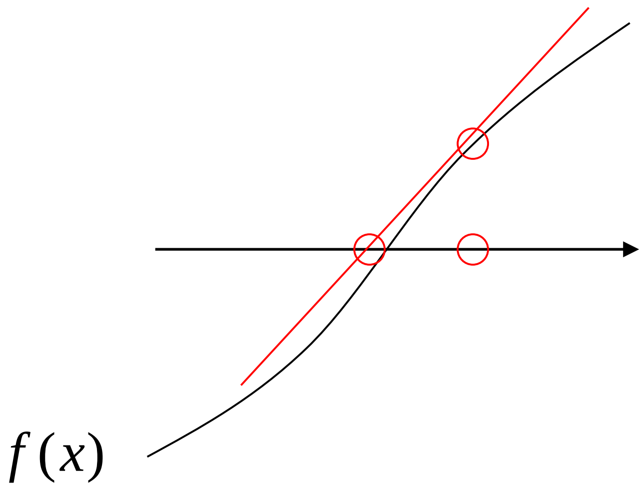
```
def test(xl,xr):
    clear()
    eold    = 0;
    drawfunc()

    for i in range(0,100):
        loop.pause()
        fl  = func(xl)
        fr  = func(xr)

        if (fl*fr>=0):
            print("Wrong initial guess.");
            return

        # New mid with Secant Method
        r   = (fr-fl)/(xr-xl)
        xm  = xl-fl/r
        fm  = func(xm)

        # print data
        loop.io.print(i,":",xl,xr,"xm=",xm)

        # draw xl,xr and line
        graph(2)
        plot(xl,fl,"rd")
        graph(3)
        plot(xr,fr,"bd")
        drawline(xl,fl, xr,fr)

        # find New xl or xr
        if (fl*fm<0):   # xl and xm as new guesses
            xr  = xm
        elif (fr*fm<0): # xm and xr as new guesses
            xl  = xm

        error   = abs(xl-xr)

        # stop or not
        if (abs(error-eold)<1e-5):
            break;
```

HW1) Find the Solution X with Bisection and Secant Methods

- Given Equation x*sin(x)+x^2*cos(x)=0
- Condition 0<x<10
- Find all solution, x within the range [0,10]



Numerical Method:
3. Newton-Raphson (NR) Method



Algorithm

1. Start with one point(a GUESS value)

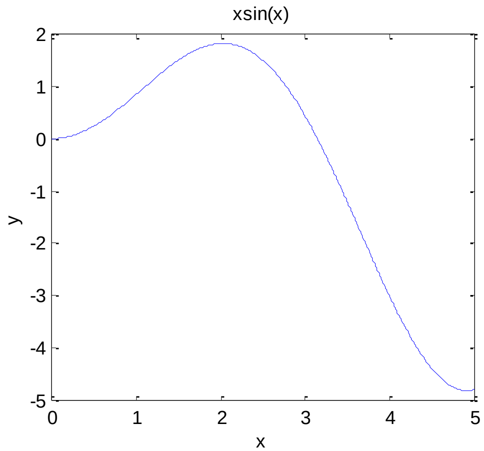
$$y = f'(x_n)(x - x_n) + f(x_n) = 0$$

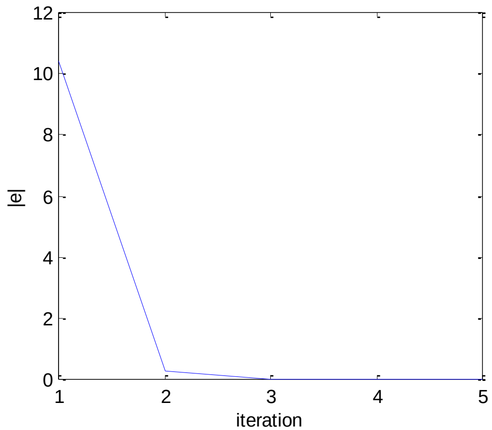
2. Find a new point

$$f'(x_n)(x_{n+1} - x_n) + f(x_n) = 0$$

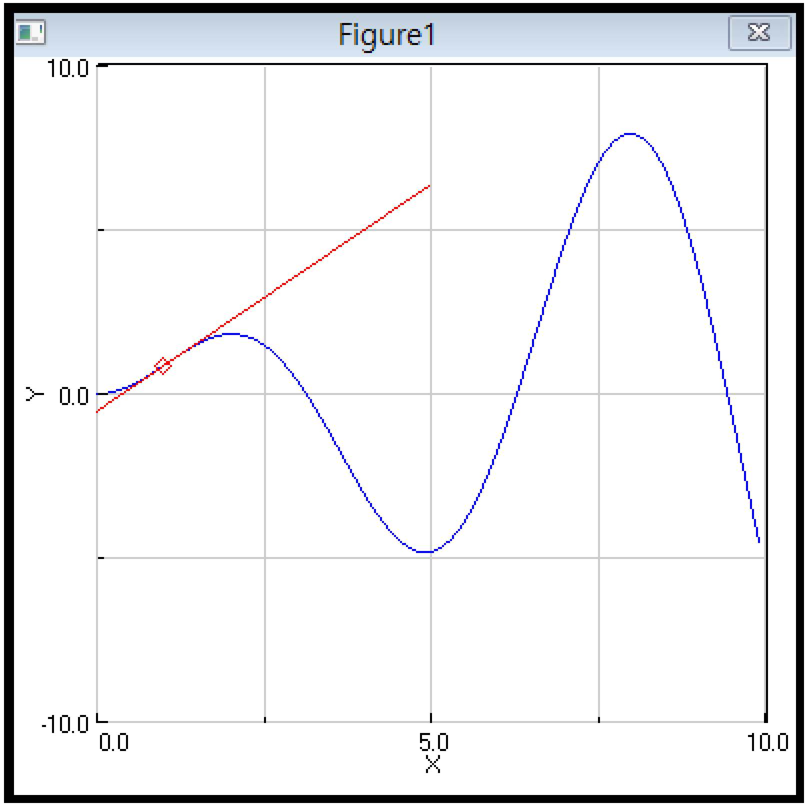
$$x_{n+1} = -\frac{f(x_n)}{f'(x_n)} + x_n$$

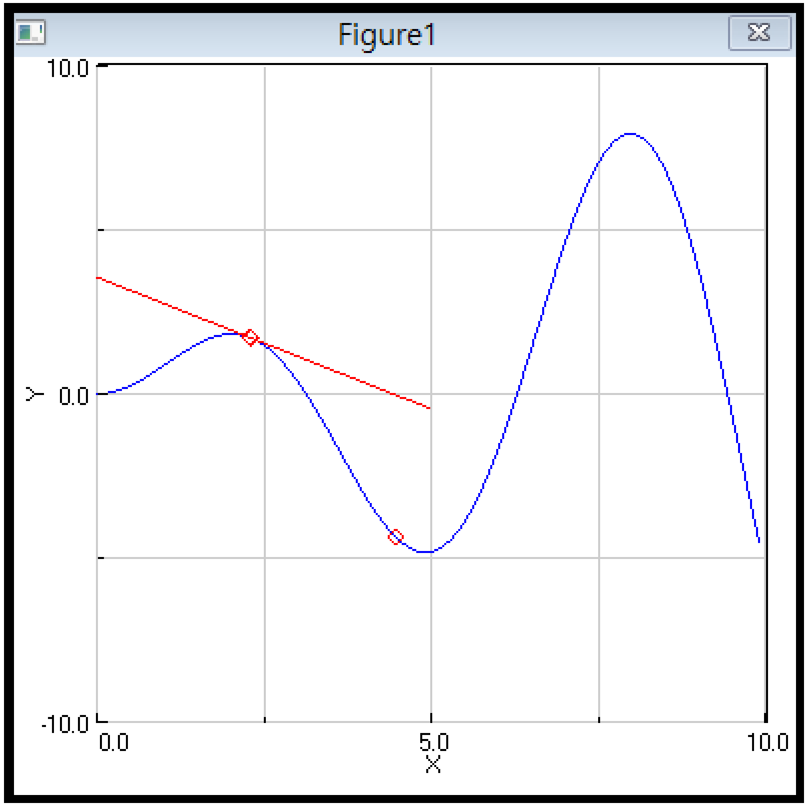
- See test3.m





Example) Newton-Raphson ex/ml/l2nr





import l2nr
l2nr.test(0.1)

import l2nr
l2nr.test(2.3)

Comparison Three Cases

Bisection

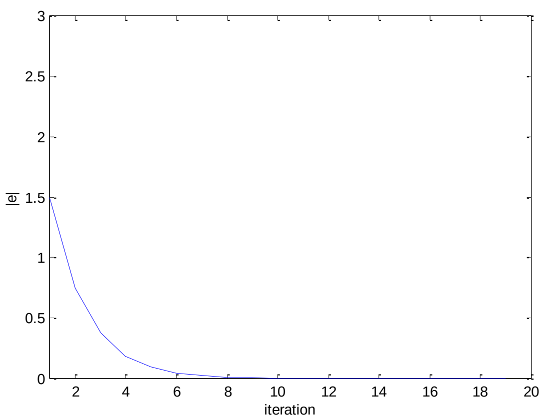
$$x_m = \frac{x_L + x_R}{2}$$

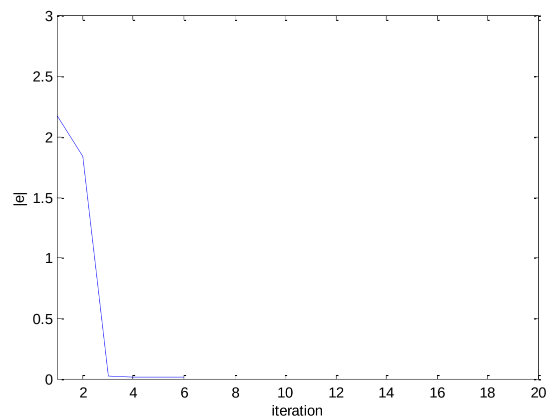
Secant

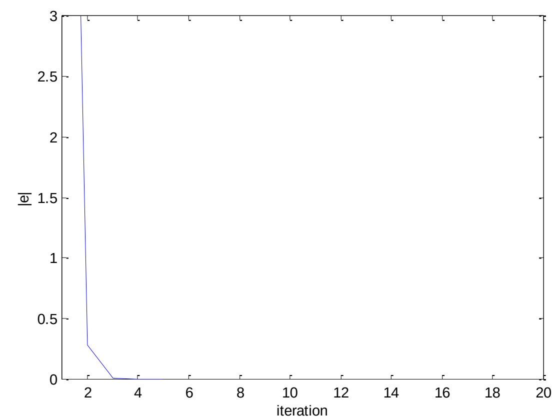
$$x_m = -\frac{f(x_L)}{\left(\dfrac{f(x_R) - f(x_L)}{x_R - x_L}\right)} + x_L$$

Newton-Raphson

$$x_{n+1} = -\frac{f(x_n)}{f'(x_n)} + x_n$$







Exponentially Converged

Good for convergence:
-Stable
-Continuous and smooth convergence but Slow

Using Ratio = Linearization

Many Nonlinear problems are approximated for linearity.

Using Ratio = Linearized method with Differentiation

- Fast
- But, initial guess is important for stability

Solution Analysis

- Bisection Method
  - Is it Bad?
  - Exponentially Converged → Stable $|e_n| \propto 2^{-k}$
- Secant Method(Faster)

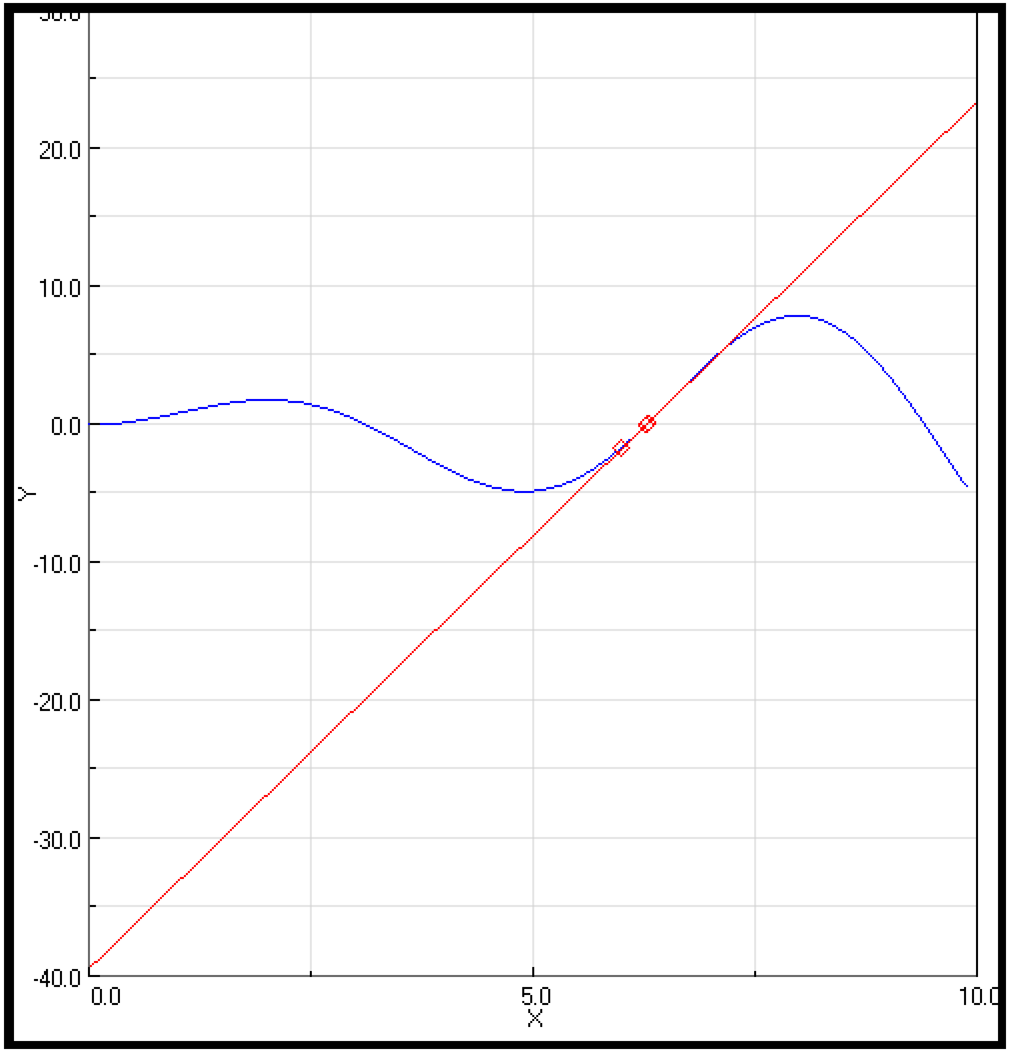
$$x_m = -\frac{f(x_L)}{\left(\frac{f(x_R)-f(x_L)}{x_R-x_L}\right)} + x_L \qquad if \ \frac{f(x_R)-f(x_L)}{x_R-x_L} = 0, \ \ it \ fails.$$

- NR Method(much Faster)

$$x_{n+1} = -\frac{f(x_n)}{f'(x_n)} + x_n \qquad if \ f'(x_n) = 0, \ \ it \ fails.$$

  - Sometimes, it becomes very unstable.

HW2)N-R method
How we get solution x= 6.28318?



- Hint: Change your initial guess

3

Numerical Methods for Multiple Equations

f(x,y)=0 and g(x,y)=0

$$\begin{pmatrix} m_1 l_1^2 + m_2 l_1^2 + m_2 l_2^2 + 2 m_2 l_1 l_2 c_2 & m_2 l_2^2 + m_2 l_1 l_2 c_2 \\ m_2 l_2^2 + m_2 l_1 l_2 c_2 & m_2 l_2^2 \end{pmatrix} \begin{pmatrix} \ddot{\theta}_1 \\ \ddot{\theta}_2 \end{pmatrix} + \begin{pmatrix} -m_2 l_1 l_2 (2\dot{\theta}_1 + \dot{\theta}_2) s_2 \dot{\theta}_2 \\ m_2 l_1 l_2 \dot{\theta}_1^2 s_2 \end{pmatrix} + g \begin{pmatrix} m_1 l_1 c_1 + m_2 (l_2 c_{12} + l_1 c_1) \\ m_2 l_2 c_{12} \end{pmatrix} = 0$$

$$f(\ddot{\theta}_1, \ddot{\theta}_2, \dot{\theta}_1, \dot{\theta}_2, \theta_1, \theta_2) = 0$$

$$\mathrm{g}(\ddot{\theta}_1, \ddot{\theta}_2, \dot{\theta}_1, \dot{\theta}_2, \theta_1, \theta_2) = 0$$

Multi Dimension Equation

- 1 Dim. Problem
  - Solve x with f(x)=0

$$x_{n+1} = -\frac{f(x_n)}{f'(x_n)} + x_n \text{ , Iteration} \qquad \therefore x_n \to x_s$$

- 2 Dim. Problem
  - Solve vector x

$$f(x,y) = 0$$
$$g(x,y) = 0$$

Ex) $f(x,y) = x^2 + y^2 - 3 = 0$

$g(x,y) = x + y - 1 = 0$

$$\hat{X}_{n+1} = \begin{pmatrix} x_{n+1} \\ y_{n+1} \end{pmatrix}, \quad F = \begin{pmatrix} f(x,y) \\ g(x,y) \end{pmatrix}$$

Matrix and Vector

$$\hat{X}_{n+1} = -\frac{F(x_n)}{F'(x_n)} + \hat{X}_n \text{ , Iteration} \qquad \hat{X}_n \to X_s$$

Multi Dim. NR uses Matrix

$$\hat{X}_{n+1} = \begin{pmatrix} x_{n+1} \\ y_{n+1} \end{pmatrix}, \quad F = \begin{pmatrix} f(x, y) \\ g(x, y) \end{pmatrix}$$

$$\hat{X}_{n+1} = -\frac{F(x_n)}{F'(x_n)} + \hat{X}_n$$

Matrix has NO DIVISION.

$$y = f'(x_n)(x - x_n) + f(x_n) = 0$$

$$f'(x_n)(x_{n+1} - x_n) + f(x_n) = 0$$

$$\therefore x_{n+1} = -\frac{f(x_n)}{f'(x_n)} + x_n$$

$$\hat{F}'(\hat{X}_{n+1} - \hat{X}_n) + \hat{F} = 0$$

$$\hat{F}'(\hat{X}_{n+1} - \hat{X}_n) = -\hat{F}$$

$$\hat{X}_{n+1} - \hat{X}_n = -\left(\hat{F}'\right)^{-1} \hat{F}$$

$$\therefore \hat{X}_{n+1} = \hat{X}_n - \left(\hat{F}'\right)^{-1} \hat{F}$$

$$= \hat{X}_n - J^{-1}\hat{F}$$

$J = \text{Jacobian Matrix}$

Remind that Multi Dim. NR requires,
- Matrix calculation
- Differentiation → Jacobian Matrix
- Division → Inverse matrix

Another Derivation: NR Method for Non-Linear Equations

- Non linear Equations, f(x,y)=0 , g(x,y)=0
  - Solve x,y
- Define F(x,y)

$$F(x,y)=\begin{pmatrix} f(x,y) \\ g(x,y) \end{pmatrix}$$

- Taylor series(or Differentiation)

$$F_i(\hat{x}+\hat{h}) \approx F_i(\hat{x})+\sum_j \frac{\partial F_i}{\partial x_j} h_j + = F_i(\hat{x})+J_i h$$

$$\therefore F(\hat{x}+h)=F(\hat{x})+J\hat{h}$$

- Remind F(x,y)=0

$$F(\hat{x}+h)=0=F(\hat{x})+J\hat{h}$$

$$\therefore \hat{h}=-J^{-1}F \qquad \rightarrow \hat{x}_{k+1}=\hat{x}_k+\hat{h}=\hat{x}_k-J^{-1}F$$

Example) nlnr.m

```
%NR example
x=3;
y=1;

J=[];

for i=1:1000
    f = x^2+y^2-10;
    g =x*y-5;
    J(1,1)   = 2*x;
    J(1,2)   = 2*y;
    J(2,1)   = y;
    J(2,2)   =x;

    Ji = inv(J);
    F = [f;g];
    h = [-Ji*F];

    X=[x;y];
    X=X+h;

    x=X(1);
    y=X(2);

     [x y]
     [f g];
    if (f^2+g^2<1e-7)
       break;
    end
    pause

end
```

- X0=3, y0=1

$$f = x^2 + y^2 - 10$$

$$g = xy - 5$$

$$J = \frac{\partial(f,g)}{\partial(x,y)} = \begin{bmatrix} \frac{\partial f}{\partial x} & \frac{\partial f}{\partial y} \\ \frac{\partial g}{\partial x} & \frac{\partial g}{\partial y} \end{bmatrix} = \begin{bmatrix} 2x & 2y \\ y & x \end{bmatrix}$$

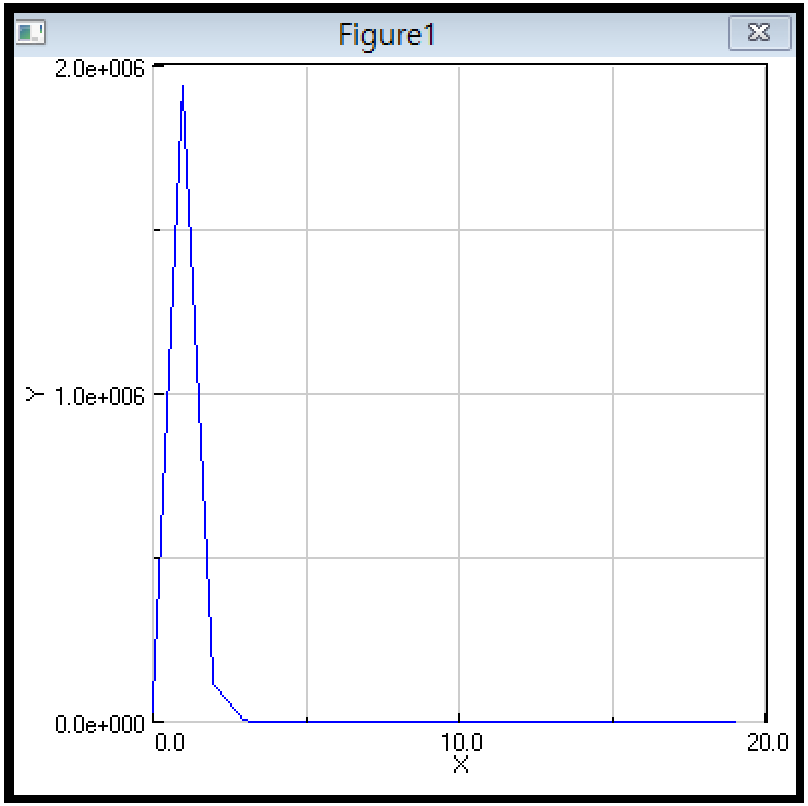
$$\hat{h} = -J^{-1}\hat{F} = -J^{-1}\begin{bmatrix} f \\ g \end{bmatrix}$$

$$\hat{X} = \begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} x \\ y \end{bmatrix} + \hat{h} = \begin{bmatrix} x \\ y \end{bmatrix} - J^{-1}\begin{bmatrix} f \\ g \end{bmatrix}$$

Non-Linear Newtown-Raphson Method

- See l2nlnr

```
l2nlnr.test(0.1,0.1)
124.50048857879665
1937546.4477837086
119174.55138009507
6988.417253141524
339.8059228162356
8.228910930054738
0.02144254383316363
2.2397752679586273e-07
2.4751775419627393e-17
3.944304526105059e-30
3.944304526105059e-30
3.944304526105059e-30
3.944304526105059e-30
3.944304526105059e-30
3.944304526105059e-30
3.944304526105059e-30
3.944304526105059e-30
3.944304526105059e-30
3.944304526105059e-30
3.944304526105059e-30
result 2.23606797749979
2.23606797749979
```



Error becomes 3.9 e-30

HW 3

2DOF SCARA Robot Inverse Kinematics Solver with Nonlinear NR Method ex/ml/l2nrscara

- Forward Kinematics(See Robotics Lecture4, pp.20)

$$x = l_1 \cos(\theta_1) + l_2 \cos(\theta_1 + \theta_2)$$
$$y = l_1 \sin(\theta_1) + l_2 \sin(\theta_1 + \theta_2)$$

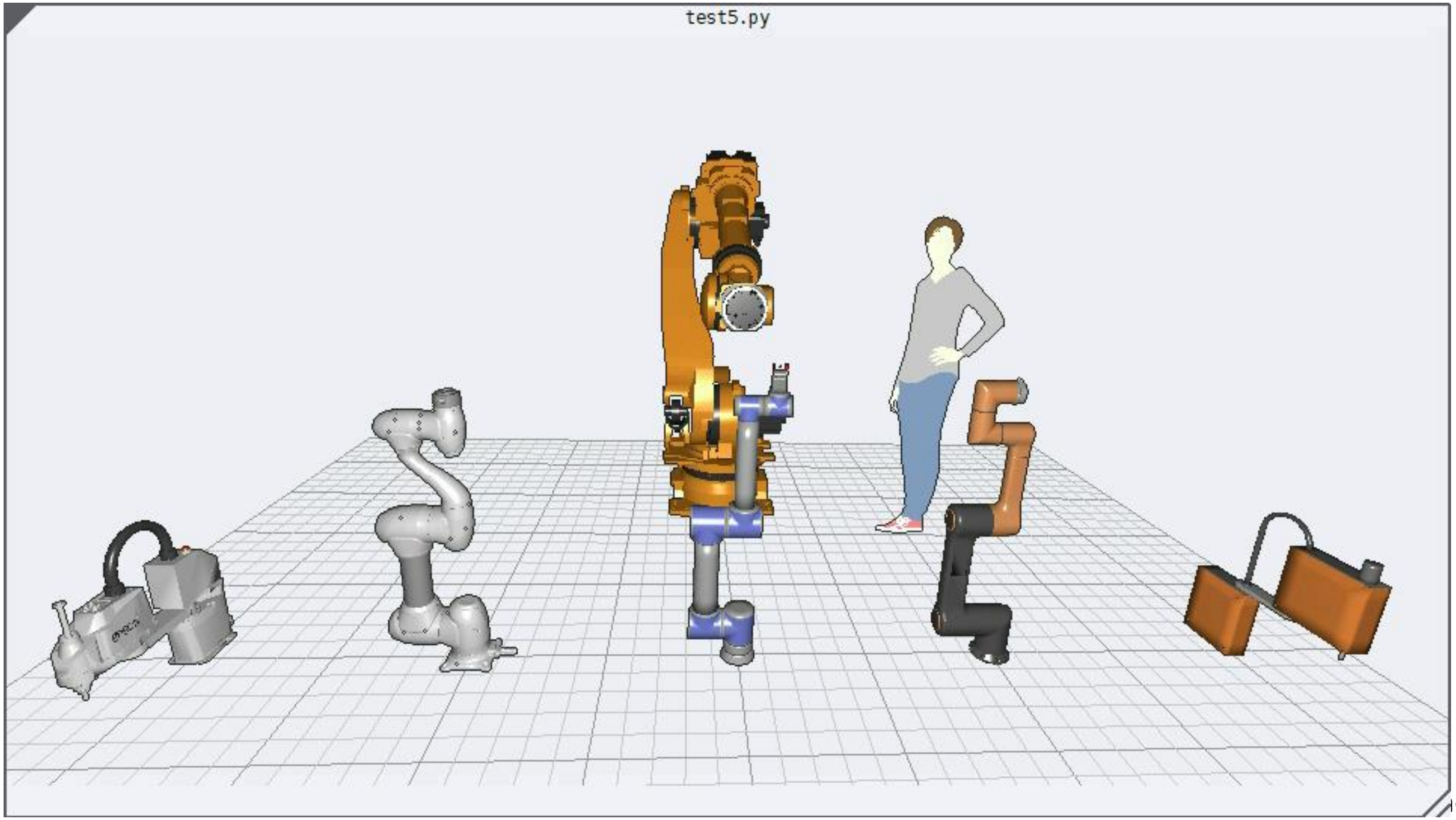
L1 = L2= 5

- When x,y are given,
  - X = 8.1603
  - Y= 5.7139

- Question: What are q1, q2 ?
- Solve it with Nonlinear Newton Raphson Method

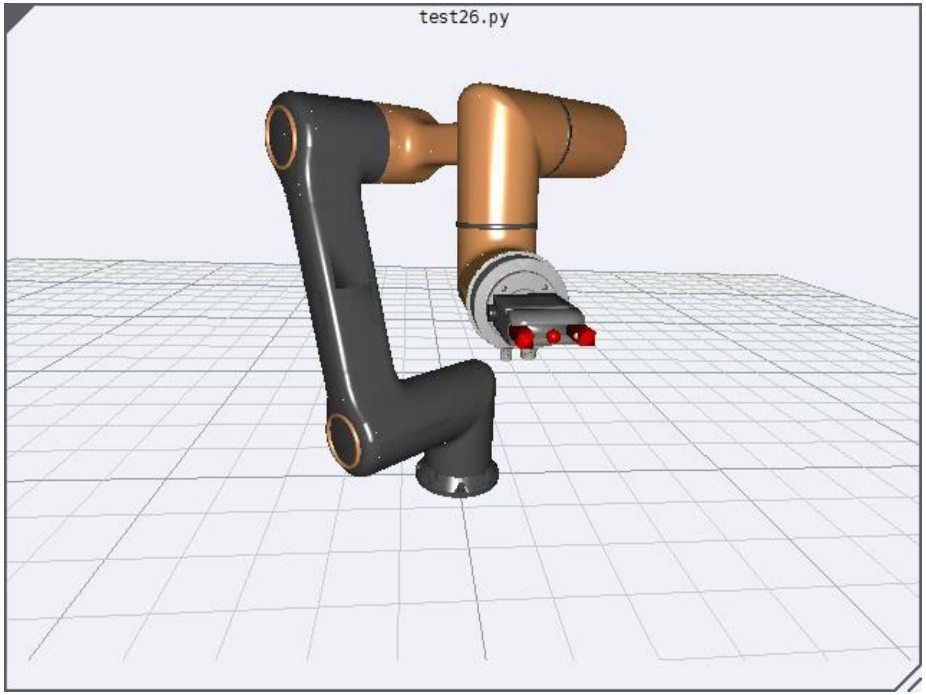
Inverse Kinematics in Robotics

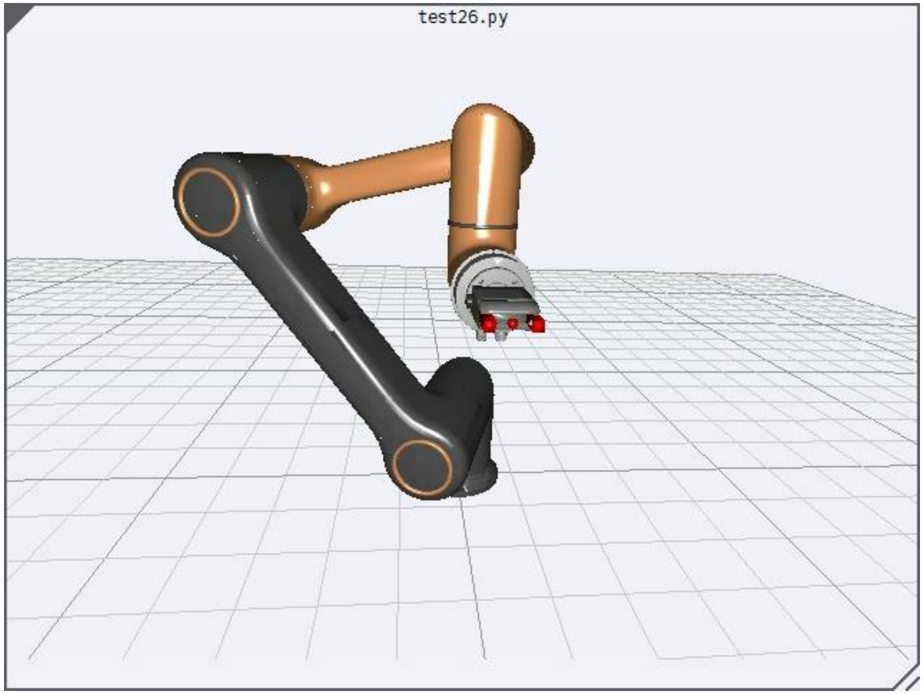
- In recent years, Most robots solve IK with Nonlinear Newtown-Raphson Method

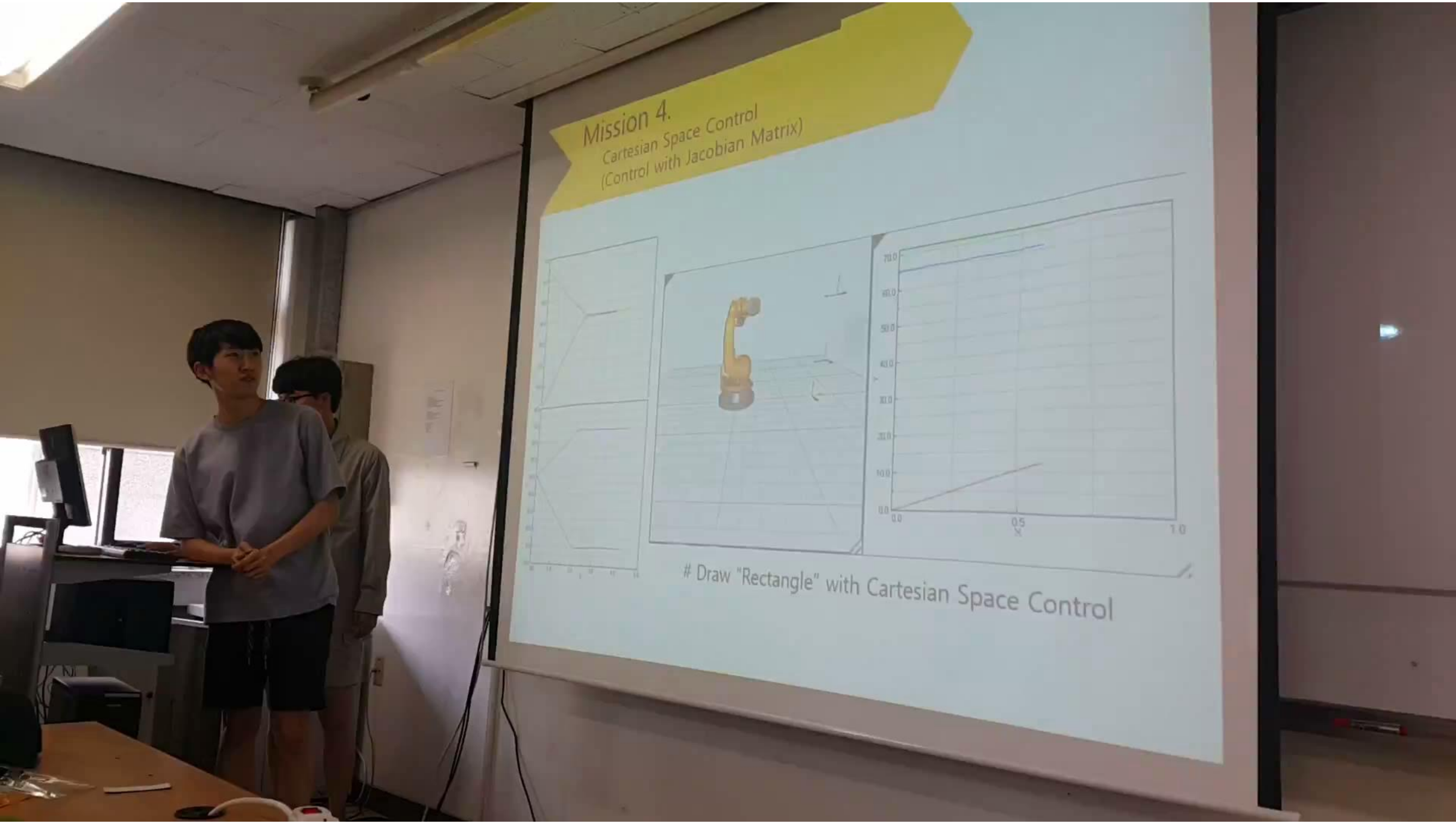


Many Cooperative Robots do NOT have Analytical Inverse Kinematics

Nonlinear Newton-Raphson Method (or Inverse Jacobian Method)






Mission 4.
Cartesian Space Control
(Control with Jacobian Matrix)
Draw "Rectangle" with Cartesian Space Control