UNIT-1

NMOUP

Dashrath Nandan

Python: Python is an interpreted, object-oriented, high-level programming language used in various domains, including numerical computation and scientific computing

Features

- Readability and ease-of-maintenance.
- Extensibility with libraries.
- Open Source
- Object Oriented
- Provide GUI Support

Data Types

- Numeric
  - Integer
  - Complex Number
  - Float
- Dictionary
- Boolean
- Set
- Sequence Type
  - Strings
  - List
  - Tuple

Basic Syntax:

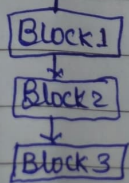
```
X = 5        # Integer      ← Comments
y = 3.14     # Float
name = "Dashrath"  # String
is_true = True     # boolean
```

Indentation:

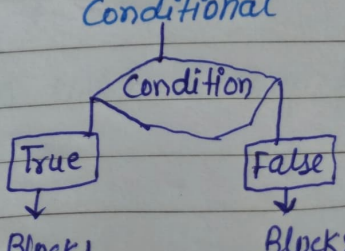
```
if x>0:
    Print("Dashrath")
else :
    print (" Nandan")
```

Control Structures:

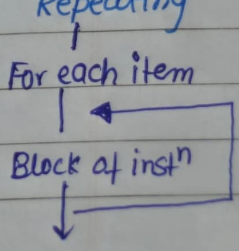
Sequential



Conditional



Repeating



Python Operators are used to perform operations on variables and values.

- Arithmetic Operator: +, -, *, /, %, ** (Exponential).
- Assignment Operator: =, +=, -=, &=, |=, *=, etc
- Comparison " : ==, !=, >, <, >=, <=
- Logical " : and, or, not.
- Bitwise " : &, |, ^, ~, <<, >>

Conditional Structures and Loops:

```
if condition:
    block code1
else:
    block code2
```

```
for i in range(5):
    print(i)
count = 0
while count < 3:
    Print("Counting", count)
    count += 1
```

```
for i in range(10):
    if i == 7:
        break
    if i % 2 == 0
        continue
    print(i):
```

★ Linear algebra, is a branch of mathematics that deals with linear equations and their representation using vectors and matrices.

- Vectors, is a mathematical quantity that have both magnitude & direction.
- Matrix, is a collection of numbers arranged in a rectangular array in rows and columns.

- SciPy is used to work with linear algebra in python.
- It also offers optimization, integration, interpolation and signal processing capabilities.

- NumPy, is the most used library for working with matrices and vectors.

Ex: import numpy as np
Vector = np.array([1,2,3,4,5])
print("Vector:", vector)

Output:
Vector: [1,2,3,4,5]

matrix = np.array([1,2,3], [4,5,6], [7,8,9])
print("Matrix:", matrix)

Output:
Matrix: [[1,2,3]
[4,5,6]
[7,8,9]]

matrix_transpose = matrix.T

★ Functions:

Python functions is a block of statements that returns the specific task. It increases code Readability and Reusability.

1. Defining a Function:
Syntax: def function_name(Parameters):
#func-body
Return Value

Ex:- def add(a,b):
Return a+b

2. Calling Functions: Syntax: 'func_name(arguments)'
Ex: 'result = add(5,3)'

3. Passing Different Types of Arguments:

1) Positional Argument: Args. passed in same order as parameters.
Eg: def greet(name, message):
return f"{message}, {name}!"

print(greet('DN', 'Hello'))
Output: Hello DN!

ii) Keyword Args: Args passed by explicitly starting the param. name.
eg: greet(message='Hi', name='Dashrath')

iii) Default Arg: Parameters with default values.
eg: def greet(name, message='Hello'):
return f"{message}, {name}!"
greet('DN')

iv) Variable-Length Arg: '*args': Non-keyword variable-length arg.
eg: def sum_all(*args):
return sum(args)
sum_all(1,2,3,4) # returns 10

v) . **kwargs: Keyword variable length arguments.
eg: def print_info(**kwargs)
for key, value in kwargs.item():
print(f"{key}:{value}")
print_info(name='Dashrath', age=21, job='Eng.')

Note: *args syntax allows the function to accept any no. of ~~Keywords~~ Positional arguments, which are then treated as a tuple.

→ **kwargs syntax allows the function to accept any no. of keyword arguments, which are then treated as a dictionary.

Scope of Variable: There are two basic scopes of variable in Python -
i) Global Variables: defined outside a function body.
ii) Local Variables: defined inside a function body.

```
Eg: total = 0
    def Sum(arg1, arg2):
        total = arg1 + arg2
    print("Inside the funct" local total:", total)
        return total.
    Sum(10,20)
    print("Outside the func" global total", total)
Output : Inside the Function Local total : 30
         Outside the function global total : 0
```

★ **Data Structures : Lists , Tuples , Dictionaries , Sets .**

	List	Tuple	Set	Dictionary
→	List is a non-homogenous data structure that stores the element in columns of a single or Mult. rows.	Tuple is a non-homogenous ds that stores elements in columns of single or Mult. rows	Set is a non-homogenous D.S but stores the element in a single row.	Non-homogenous DS that stores Key-Value pairs.
→	my_list = [1,2,3,'a']	→ my_tup = (1,2,3,'a',)	→ my_set = {1,2,3,'a'}	→ my_dict = {'key1': 'value1', 'key2': 'v2'}
→	Represented by []	→ ... by ( )	→ ... { }	→ ... by { }
→	List is Mutable.	→ Immutable	→ mutable	→ mutable.
	List is Ordered	→ Ordered.	→ Unordered	→ Ordered.
→	Accessing Elem: my_list[0]	my_tuple [0]		my_dict ['key1']
Eg:	[1,2,3,4,5]	(1,2,3,4,5)	{1,2,3,4,5}	{1:"a", 2: "b"}
→	Allow duplicate elements	Allow	Do not allow duplicate elements	Do not allow.

★ Numerical Methods

Importance of Numerical Methods in Python:

- ↳ Complex problem solving.
- ↳ Approximation of solutions.
- ↳ Computational Effeciency.
- ↳ Optimization and minimization.
- ↳ Data Analysis & Signal Processing.

Applications:

- Root Finding.
- Linear algebra Operations.
- Ordinary Diff. Equation (ODEs) and Partial DE (PDEs).

⇒ Libraries for numerical computing.

1. **NumPy**: It is the fundamental package for numerical computing in Python.

   Key features:

   a) Multidimensional arrays: Effecient D.S. for representing Vectors, matrices,
   b) Mathematical functions: Wide range of math. operations.
   c) Broadcasting: Powerful mechanism for performing operations of different shapes.

2. **SciPy**: It is built on NumPy and provides additional functionality for scientific computing.

   Key features:

   ↳ Integration, interpolation, and optimization tools.
   ↳ Special functions:
   ↳ Signal and image processing functions.
   ↳ Statistical functions.

Eg: pip install numpy scipy

```
import numpy as np
arr = np.array([1,2,3,4,5])
mean_value = np.mean(arr)
print(mean_value)
```

```
from scipy.optimize import minimize
def obj_fun(x):
    return x**2+4*x+4
result= minimize(obj_fun, x0=0)
```

★ Root-Finding Methods:

i) Bisection Method
ii) Newton-Raphson method
iii) Secant method

1. Bisection Method :-

It is a root-finding algorithm used to find a solution to the eqn. $f(x)=0$. It is based on Intermediate Value theorem, which states that for any continuous func. f over an interval $[a,b]$ where $f(a)$ and $f(b)$ have Opposite signs, there exists at least One root in that interval.

✳ Steps of the Bisection Method –

i) Initial Interval Selection: Choose two points a and b, such that $f(a) \cdot f(b) < 0$.
ii) Midpoint Calculation: Calculate $c = \frac{a+b}{2}$. [interval $[a,b]$]
iii) Function Evaluation : Compute $f(c)$.

iv) Sign check:
- If $f(c)=0$, then c is the root.
- if $f(a)\cdot f(c)<0$, the root lies in $[a,c]$, set $b=c$.
- if $f(b)\cdot f(c)<0$, the root lies in $[c,b]$, set $a=c$.

v) Iteration: Repeat the process until the interval $[a,b]$ is sufficiently small (i.e. $|b-a|<$ predetermined tolerance level).

Examples:

Ques: Find roots of the function $f(x)=x^2-4$ in interval $[1,3]$

Sol:- $[a,b]=[1,3]$ $\quad f(x)=x^2-4$

$f(a)=-3$

$f(3)=5 \Rightarrow f(1)\cdot f(3)<0$ $\quad \therefore$ at least one root is present b/w $[a,b]$.

Step 2: $c=\frac{a+b}{2}$, $c=\frac{1+3}{2}=2$.

Step 3: $f(2)=4-4=0$

Step 4: $\because f(c)=0$, $\therefore$ 2 is the root.

Ques) Find the root of the function $f(x)=x^3-x-2$ in the interval $[1,2]$.

Sol:- $f(x)=x^3-x-2$

$f(1)=1-1-2=-2$ $\quad \therefore f(1)\cdot f(2)<0$

$f(2)=8-2-2=4$

Note: YT: [Bisection Method – Dr. Gajendra Purohit]

Question is solved using the same approach

Or, find $f(a)$, $f(b)$, $f(c)$ then use the sign check step.

Iterations	$\ominus$ $f(a)<0$	$\oplus$ $f(b)>0$		
n	a	b	$c=\left(\frac{a+b}{2}\right)$	$f(c)$
1	1	2	$(1+2/2)=1.5$	$(1.5)^3-1.5-2$ $=-0.125<0$
	$\because f(c)<0$,	update $a=c$,	$[c,b]$ will be new interval.	
		($f(a)<0$)		
2	1.5	2	1.75	$1.609>0$
	$\because f(c)>0$,	update $b=c$,	$[a,c]$ will be new interval.	
3	1.5	1.75	1.625	$0.66>0$
4	1.5	1.625	1.562	$0.250>0$
5	1.5	1.562	1.531	$0.057>0$
6	1.5	1.531	1.515	$-0.036<0$
7	1.515	1.531	1.523	$1.857>0$
8	1.515	1.523	1.519	$-0.14<0$
9	1.519	1.523	1.521	$-0.02<0$
10	1.521	1.523	1.522	$0.0036>0$
11	1.521	1.522	1.5215	$0.0007>0$
	1.521	1.5215	1.521	

$\therefore$ Root is 1.521, since it's started repeating or $[1.521, 1.5215]$ interval is sufficiently small.

Ques: find the root of an equation, $f(x)=x^3-x-1$ using bisection method. [PPt – 1.3.3 LMS]

Sol:-

$f(x)=x^3-x-1$

$f(0)=-1$

$f(1)=-1$ ↰

$f(2)=5$ ↲ opposite sign, $\therefore$ the interval is $[1,2]$

	f(1)<0	f(2)>0			
n	a ⊖	b ⊕	$c = \frac{(a+b)}{2}$	f(c)	update
1	1	2	1·5	0·875 >0	b=c
2	1	1·5	1·25	−0·296<0	a=c
3	1·25	1·5	1·375	0·224 >0	b=c
4	1·25	1·375	1·3125	−0·051 <0	a=c
5	1·3125	1·375	1·3437	0·082 >0	b=c
6	1·3125	1·3437	1·3281	0·014 >0	b=c
7	1·3125	1·3281	1·3203	−0·018<0	a=c
8	1·3203	1·3281	1·3242	−0·002<0	a=c
9	1·3242	1·3281	1·3261	0·006>0	b=c
10	1·3242	1·3261	1·3252	0·002>0	b=c
11	1·3242	1·3252	1·3247	−0·00007<0	a=c
12	1·3247	1·3252	1·3249		

Since, c starts repeating

∴ root ≈ 1·324 Ans.

Pseudocode:

```
function bisection(f, a, b, tol)
  if f(a) * f(b) >= 0
    error "f(a) and f(b) must have opp. sgn"
  end if
While (b-a)/2 > tol
  c = (a+b)/2
  if f(c) == 0
    return c
  else if f(c) * f(a) < 0
      b = c
  else
      a = c
  end if
end while

return (a+b)/2
end function.
```

Advantages	Disadvantages
i) Simplicity	i) Slow convergence.
ii) Guranteed Convergence	ii) Initial Interval Requirement.
iii) Robustness	

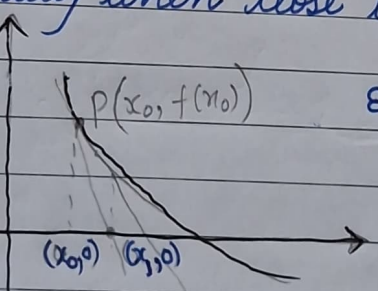
2. Newton-Raphson Method

It provides an iterative procedure to approximate the roots of a real-valued function. It is based on the idea of using the tangent line to the function at a gussed point to find a better approximation of the root.

- This method is particularly useful because of its rapid convergence, especially when close to the roots.

Eqⁿ. of tang. at $P(x_0, f(x_0))$

$\Rightarrow y - f(x_0) = f'(x_0)(x - x_0)$



Eq. of tangent $\Rightarrow$ $y - y_1 = \frac{dy}{dx}(x - x_1)$

$(x_1, 0) \Rightarrow 0 - f(x_0) = f'(x_0)(x_1 - x_0)$

$\Rightarrow x_1 - x_0 = -f(x_0) / f'(x_0) \Rightarrow x_1 = x_0 - \frac{f(x_0)}{f'(x_0)}$

Formula :

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

x_n: Current approximation of root.
x_{n+1}: Next approximation
$f'(x_n)$: derivative of $f(x)$ at x_n.

* Steps of the Newton-Raphson Method

1. **Initial Guess:** Choose an initial approximation x_0 for the root.
2. **Iteration:** Apply the iteration formula to compute next approximation.
3. **Convergence Check:** Repeat the iteration until the diff. between successive approximation is smaller than a predetermind tolerance level.

Ex: finding the roots of $f(x) = x^3 - x - 2$ [LMS :→ Word 1.3.4]

Sol: $f(x) = x^3 - x - 2$, $f'(x) = 3x^2 - 1$

$f(0) = -2$

- $f(1) = -2$ ↔ Opposite sign, means there is at least one root b/w 1 and 2.
  $f(2) = 4$ ↔

⇒ **Initial guess:** $x_0 = 1.5$

⇒ Iteration formula : $x_{n+1} = x_n - \frac{x_n^3 - x_n - 2}{3x_n^2 - 1}$

⇒ Iterations:

Putting $n=0$; $x_0 = 1.5$

$$x_1 = 1.5 - \frac{1.5^3 - 1.5 - 2}{3 \times 1.5^2 - 1} = 1.5 - \frac{-0.125}{5.75} \approx 1.5217$$

$n=1$

- $x_1 = 1.5217$

$$x_2 = 1.5217 - \frac{f(1.5217)}{f'(1.5217)} \approx \underline{1.5214}$$

$n=1$,

- $x_2 = 1.5214$

$$x_3 = 1.5214 - \frac{f(1.5214)}{f'(1.5214)} \approx \underline{1.5214}$$

∴ The root ≈ 1.5214, as it start repeating.

YT: Netwon-Raphson (Q1) – Gajendra Purohit

Ex: find by Newton Raphon method, a root of $x^3-3x-5=0$

Sol: $f(x)= x^3-3x-5$ $\quad f'(x) = 3x^2-3$

$f(0) = -5$

$f(1) = -7$ ← it is closer to zero than (16)

$f(2) = -3$ ✓

$f(3) = 16$ ↵ → opp. sign ⇒ at least 1 root in $[2, 3]$

⇒ Inital guess: $x_0 = 2$

⇒ Iterating formula: $x_{n+1} = x_n - \dfrac{x_n^3-3x_n-5}{3x_n^2-3}$

⇒ Iterations:

- $x_0 = 2$

$(n=0)$ $x_1 = 2 - \dfrac{(2^3-3\times2-5)}{(3\times4-3)} = 2 - \dfrac{(-3)}{9} = 2+\dfrac{3}{9} \approx 2.333$

- $x_1 \approx 2.333$

$(n=1)$ $x_2 = 2.333 - \dfrac{f(2.333)}{f'(2.333)} \approx 2.2805$

- $x_2 = 2.2805$

$(n=2)$ $x_3 = 2.2805 - \dfrac{f(2.2805)}{f'(2.2805)} \approx \boxed{2.2790}$

- $x_3 = 2.2790$

$(n=3)$ $x_4 = 2.2790 - \dfrac{f(2.2790)}{f'(2.2790)} \approx \underline{2.2790}$

$\therefore$ $\boxed{\text{Root} \approx 2.2790}$

Pseudocode:

```
function newtonRaphson(f, df, x0, tol, maxIter)
    x=x0
    for i = 1 to maxIter
        x_new = x - f(x)/df(x)
        if abs(x_new - x) < tol
            return x_new
        end if
        x = x_new
    end for
    error "Method did not converge"
end function.
```

Advantage:
- Rapid Convergence
- Simplicity
- Wide Applicability

Disadvantage:
- Derivative Requirement
- Initial guess sensitivity

3. Secant Method

It is an iterative numerical technique used to find the roots of non-linear equation.
→ This method is useful when the derivative of the function is difficult to compute.

Basic principle :- This method approximates the root of a function $f(x)$ by using two initial approximation , x_0 and x_1. The method iteratively refines these approximation by constructing secant lines between these points & using the x-intercepts of these line as new approximation.

Formula :

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

x_n : Current

x_{n-1} : previous approx.

Steps:

1. **Initial guesses:** Start with two initial approximation x_0 and x_1.

2. **Iterative process:** Apply formula:-

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

3. **Convergence check:** Check if $|x_{n+1} - x_n|$ or $|f(x_{n+1})| < tol$

4. **Repeat:** Repeat steps 2 and 3 until convergence.

* Pseudocode:

```
function secantMethod(f, x0, x1, tol, maxIter)
    for i = 1 to maxIter
        x2 = x1 - f(x1)*(x1-x0)/f(x1) - f(x0)

        if abs(x2-x1) < tol
            return x2
        end if
        x0 = x1
        x1 = x2
    end for
end function
```

Ex: A real root of the equation $x^3 - 5x + 1 = 0$ lies in the interval $(0,1)$. Perform four iterations of the secant method. [YT: Secant Method (Q1) - Gajendra Purohit]

Sol: $f(x) = x^3 - 5x + 1$, $x_{n+1} = x_n - \frac{f(x_n)(x_n - x_{n-1})}{f(x_n) - f(x_{n-1})}$ ← formula

$f(0) = 1$
$f(1) = -3$

$\Rightarrow$ $x_0 = 0$, $x_1 = 1$

$(n=1) \Rightarrow x_2 = x_1 - \frac{f(x_1)(x_1 - x_0)}{f(x_1) - f(x_0)} = 1 - \frac{(-3)(1-0)}{(-3) - (1)} \approx 0.25$

$(n=2)$ $x_1 = 1$, $x_2 = 0.25$

$x_3 = 0.25 - \frac{f(0.25)(0.25 - 1)}{f(0.25) - f(1)} \approx 0.18644$

$(n=3)$ $x_2 = 0.25$, $x_3 = 0.186$

$x_4 = 0.186 - \frac{f(0.186)(0.186 - 0.25)}{f(0.186) - f(0.25)} \approx 0.201$

$(n=4)$ $x_3 = 0.286$, $x_4 = 0.201$

$x_5 = 0.201 - \frac{f(0.201)(0.201 - 0.186)}{f(0.201) - f(0.186)} \approx 0.20081$

$\therefore \boxed{x_5 = 0.20081}$

UNIT: 2

NMOUP

Dashrath Nandan

classmate Date Page

★ Polynomial Interpolation

Interpolation, is a technique or method of estimating unknown values from given set of observations.

➤ Polynomial Interpolation, is the process of finding a polynomial that passes exactly through a given set of datapoints.

For (N+1) datapoints:- $(x_0, y_0), (x_1, y_1) \ldots (x_n, y_n)$, there is a unique polynomial P(x) of degree at most N, that passes through all the datapoints.

i.e. $P(x_i) = y_i \quad \forall \; i \in [0, N]$

- It fits the data exactly.

I. Lagrange Interpolation

Lagrange interpolation, is a polynomial interpolation method that construct a polynomial which passes exactly through a set of known data points.

- It is useful when you have a small number of points and need an exact fit between them.

⇒ Unequal interval:

For, N+1 data points : $(x_0, y_0) \ldots (x_n, y_n)$, L.I.Polynomial, P(x):

$$P(x) = \sum_{i=0}^{n} y_i L_i(x)$$

Lagrange basis Polynomial

$$L_i(x) = \prod_{\substack{0 \le j < n \\ j \neq i}} \frac{x - x_j}{x_i - x_j}$$

Example 1: Use Lagrange's method to find the polynomial that passes through the points (1,2), (2,3), (4,1).

Sol:

P(x) =	X	1^{x_0}	2^{x_1}	4^{x_2}
	y	2_{y_0}	3_{y_1}	1_{y_2}

$N+1=3$

↳ from the formula, $P(x) = \sum_{i=0}^{n=2} y_i \cdot L_i(x)$, Data points = 3

$P(x)$ of dg. 2.

$N=2$

$$\Rightarrow \boxed{P(x) = y_0 \cdot L_0(x) + y_1 \cdot L_1(x) + y_2 \cdot L_2(x)}$$

↳ Using formula, $L_i(x) = \prod_{\substack{0 \le j < n \\ j \ne i}} \frac{x - x_j}{x_i - x_j}$

$$\Rightarrow P(x) = y_0 \cdot \underbrace{\left(\frac{x-x_1}{x_0-x_1}\right)\left(\frac{x-x_2}{x_0-x_2}\right)}_{L_0} + y_1 \cdot \underbrace{\left(\frac{x-x_0}{x_1-x_0}\right)\left(\frac{x-x_2}{x_1-x_2}\right)}_{L_1} + y_2 \cdot \underbrace{\left(\frac{x-x_0}{x_2-x_0}\right)\left(\frac{x-x_1}{x_2-x_1}\right)}_{L_2}$$

$$= y_0 \cdot \left(\frac{x-2}{1-2}\right)\left(\frac{x-4}{1-4}\right) + y_1 \cdot \left(\frac{x-1}{2-1}\right)\left(\frac{x-4}{2-4}\right) + y_2 \left(\frac{x-1}{4-1}\right)\left(\frac{x-2}{4-2}\right)$$

Putting values of y_i's $= 2 \times \frac{(x-2)(x-4)}{3} + 3\frac{(x-1)(x-4)}{(-2)} + 1\frac{(x-1)(x-2)}{6}$

$$P(x) = \frac{2}{3}(x^2-6x+8) + \frac{3}{2}(x^2-5x+4) + \frac{1}{6}(x^2-3x+2)$$

$$= \frac{(4x^2-24x+32) - (9x^2+45x-36) + (x^2-3x+2)}{6}$$

$$= -2x^2+9x-1/3$$

So, the Lagrange interpolating polynomial is

$$y = P(x) = \frac{1}{3}(-2x^2+9x-1)$$

[YT: Grajendra Purohit]

Ques: Find value of y (or $P(x)$) when $x = 10$ using Lagrange's Interpolation.

x	5	6	9	11
y	12	13	14	16

$N+1 = 4$
$\Rightarrow N = 3$

Sol:-

$$y = p(x) = y_0L_0 + y_1L_1 + y_2L_2 + y_3L_3$$

$$= 12 \times \frac{(x-6)(x-9)(x-11)}{(5-6)(5-9)(5-11)} + 13 \times \frac{(x-5)(x-9)(x-11)}{(6-5)(6-9)(6-11)} +$$

$$14 \times \frac{(x-5)(x-6)(x-11)}{(9-5)(9-6)(9-11)} + 16 \times \frac{(x-5)(x-6)(x-9)}{(11-5)(11-6)(11-9)}$$

$$p(x) = \frac{12}{-24}(x-6)(x-9)(x-11) + \frac{13}{15}(x-5)(x-9)(x-11) + \frac{14}{-24}(x-5)(x-6)(x-11)$$

$$+ \frac{16}{60}(x-5)(x-6)(x-9)$$

↳ Simplify the equation to get Lagrange Interpolation poly:

$\Rightarrow$ Put, $x = 10$ in $p(x)$:→

$$y = p(x) = \frac{1}{+2}(4)(1)(+1) + \frac{13}{15_3}(5)(1)(-1) + \frac{7}{+15_3}(5)(4)(+1) + \frac{8}{15_3}(5)(4)(1)$$

$$= 2 - \frac{13}{3} + \frac{35}{3} + \frac{16}{3} = \frac{77-13}{3} = \frac{44}{3} = 14.66$$

$\therefore$ $X = 10$, $y = 14.66$ Ans

Application: Used in Numerical analysis, Computer graphics and data fitting.

Advantages: Simple to understand and implement.
Provide an exact fit for the given points.

II. Newton Interpolation [YT: Grajendra Purohit]

It is a technique to find an interpolating polynomial that fits a set of given points using divided difference.
→ It is widely used when data points are unequally spaced.

for, N data points,

$$P(x) = f(x_0) + (x-x_0)\Delta f(x_0) + (x-x_0)(x-x_1)\Delta^2 f(x_0) + \cdots + (x-x_0)(x-x_1)\cdots(x-x_{n-2})\Delta^{n-1} f(x)$$

Let, $(x_0, y_0), (x_1, y_1), (x_2, y_2), (x_3, y_3)$

X	$Y=f(x)$	$\Delta f(x)$	$\Delta^2 f(x)$	$\Delta^3 f(x)$
x_0	y_0	$\frac{y_1-y_0}{x_1-x_0} \to \Delta f(x_0)$	$\frac{\Delta f(x_1)-\Delta f(x_0)}{x_2-x_0} \to \Delta^2 f(x_0)$	$\frac{\Delta^2 f(x_1)-\Delta^2 f(x_0)}{x_3-x_0}$
x_1	y_1	$\frac{y_2-y_1}{x_2-x_1} \to \Delta f(x_1)$	$\frac{\Delta f(x_2)-\Delta f(x_1)}{x_3-x_1} \to \Delta^2 f(x_1)$	
x_2	y_2	$\frac{y_3-y_2}{x_3-x_2} \to \Delta f(x_2)$		
x_3	y_3			

Advantages: Efficient, Ease of computation.

Ques.) find value of 'y' when $x = 10$ by Newton-Divided Difference formula.

[Gajendra Purohit YT: Ex-2]

$f(x) =$	x	5	6	9	11
	y	12	13	14	16

	x	$f(x)$	$\Delta f(x)$	$\Delta^2 f(x)$	$\Delta^3 f(x)$
x_0	5	12	$\frac{13-12}{6-5} = 1$	$\frac{1/3 - 1}{9-5} = -1/6$	$\frac{2/15 - (-1/6)}{11-5} = \frac{1}{20}$
x_1	6	13	$\frac{14-13}{9-6} = 1/3$	$\frac{1 - 1/3}{11-6} = \frac{2}{15}$	
x_2	9	14	$\frac{16-14}{11-9} = 2/2 = 1$		
x_3	11	16			

$$\Rightarrow p(x) = f(x_0) + (x-x_0)\Delta f(x_0) + (x-x_0)(x-x_1)\Delta^2 f(x_0) + (x-x_0)(x-x_1)(x-x_2)\Delta^3 f(x_0)$$

$$= 12 + (x-5)(1) + (x-5)(x-6)\left(\frac{-1}{6}\right) + (x-5)(x-6)(x-9)\left(\frac{1}{20}\right)$$

$$\boxed{p(x) = 12 + (x-5) - \frac{(x-5)(x-6)}{6} + \frac{(x-5)(x-6)(x-9)}{20}}$$

↳ To get polynomial, simplify it.

⇒ To get value of y at $x = 10$, put $x = 10$ in $P(x)$.

$$P(10) = y_{at(10)} = 12 + (10-5) - \frac{(10-5)(10-6)}{6} + \frac{(10-5)(10-6)(10-1)}{20}$$

$$= 12 + 5 - \frac{10}{3} + 1$$

$$= 18 - \frac{10}{3} = 14.66$$

$\boxed{\begin{array}{l} x = 10 \\ y = 14.66 \end{array}}$ Ans

* Basic Algebra (Not that much Important)

- a linear equation in 'n' variables:

$$a_1x_1 + a_2x_2 + a_3x_3 + \cdots + a_nx_n = b$$

where, $a_1, a_2 \cdots, b$: real number.

a_1 : leading coefficient

x_1 : leading variable.

- a System of m equation in 'n' variables –

$$a_{11}x_1 + a_{12}x_2 + a_{13}x_3 + \cdots + a_{1n}x_n = b_1$$
$$a_{21}x_1 + a_{22}x_2 + a_{23}x_3 + \cdots + a_{2n}x_n = b_2$$
$$\vdots$$
$$a_{m1}x_1 + a_{m2}x_2 + a_{m3}x_3 + \cdots + a_{mn}x_n = b_n$$

Matrix form : $\boxed{Ax = b}$

$$A = \begin{bmatrix} a_{11} & a_{12} & a_{13} \cdots a_{1n} \\ a_{21} & a_{22} & a_{23} \cdots a_{2n} \\ \vdots & & \\ a_{m1} & a_{m2} & a_{m3} \cdots a_{mn} \end{bmatrix} ; \quad x = \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ \vdots \\ x_n \end{bmatrix} ; \quad b = \begin{bmatrix} b_1 \\ b_2 \\ b_3 \\ \vdots \\ b_n \end{bmatrix}$$

⇓ Coefficent Matrix

- Homogeneous system of linear equation : A system of linear equation is said to be homogenous if all the constant terms is zero.

$$a_{11}x_1 + a_{12}x_2 + \cdots + a_{1n}x_n = 0$$
$$a_{21}x_1 + a_{22}x_2 + \cdots + a_{2n}x_n = 0$$
$$\vdots$$
$$a_{m1}x_1 + a_{m2}x_2 + \cdots + a_{mn}x_n = 0$$

★ Polynomial Curve Fitting

It is a process of constructing a curve that has the best fit to a series of data points.
Unlike polynomial interpolation, which requires the polynomial to pass exactly through all given data points, polynomial curve fittings seeks to find a polynomial that best approximates the data in a Least Square Sense.

⇒ Most common method for curve fitting :-

Least Square Fitting: This method minimizes the sum of the squares of the residuals (the diff. between observed and fitted values).

- For a polynomial fit, we aim to find the coefficients $a_0, a_1, a_2, \ldots a_n$ such that -

$$y = a_0 + a_1 x + a_2 x^2 + \cdots + a_n x^n$$

minimizes the residual sum of squares:

$$S = \sum_{i=1}^{n} \left(y_i - (a_0 + a_1 x_i + a_2 x_i^2 + \cdots + a_n x_i^n) \right)^2$$

✱ **Linear Least Square**: The model function is assumed to be a straight line of the form :- $y = mx + b$

$$m = \frac{n\Sigma xy - \Sigma x \Sigma y}{n\Sigma x^2 - (\Sigma x)^2} \quad ; \quad b = \frac{\Sigma y - m\Sigma x}{n}$$

n = No. of data points

Ques:- Write a linear equation that 'best fits' the data-

X	1	2	3	4	5
Y	2	3	5	4	5

$N = 5$

Sol:-

X	Y	XY	X^2
1	2	2	1
2	3	6	4
3	5	15	9
4	4	16	16
5	5	25	25
$\Sigma X = 15$	$\Sigma Y = 19$	$\Sigma XY = 64$	$\Sigma X^2 = 55$

$$\Rightarrow m = \frac{n\Sigma xy - \Sigma x \Sigma y}{n\Sigma x^2 - (\Sigma x)^2}$$

$$= \frac{5 \times 64 - 15 \times 19}{5 \times 55 - 255}$$

$$= \frac{7}{10} = 0.7$$

$$\Rightarrow b = \frac{\Sigma y - m\Sigma x}{n} = \frac{19 - 0.7 \times 15}{5} = 1.7$$

$\therefore$ $\boxed{y = 0.7x + 1.7}$

Types ⇒

Linear Fitting

- It involves, fitting a straight line to the data points.
- Eqⁿ: $y = a_0 + a_1 x + a_2 x_2 + \cdots$
- Simple and easy to Compute.
- Solved using Least Square method.
- Always converge to a unique solution.
- Result in a straight line or a hyperplane.

Non-Linear Fitting

- It involves fitting more complex model or non-linear funⁿ.
- Eq: $y = ae^{bx}$
- More complex.
- Solved using iterative method like gradient descent.
- May have multipe local maxima.
- Curved or more complex shapes.

Application: Data analysis, Finance, Machine Learing,

Advantages: Handles noisy data, Flexible.

★ Numerical Integration

It is a fundamental technique in numerical analysis to approximate the value of a definite integral when an exact solution is difficult to find.

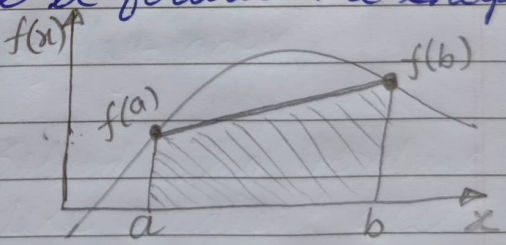
Application: Physics, Engineering, economics, etc.

Methods used for numerical integration:

i) Trapezoidal rule.
ii) Simpson's 1/3 rule.
iii) Simpson's 3/8 rule.
iv) Gaussian Quadrature.

1) Trapezoidal Rule: It is a numerical method used to approximate the value of a definite integral, by approximating the area under the curve using trapezoid instead of rectangles.

It is more accurate because it follows the shape of the curve more closely.



Formula:-

$$\int_a^b F(x)dx = h\left[\frac{Y_0+Y_n}{2} + Y_1+Y_2+\cdots\cdots+Y_{n-1}\right]$$

$$h = \frac{b-a}{n}$$

where, n = no. of segments / no. of interval (choose any number)

$$a = x_0$$
$$x_1 = x_0 + h$$
$$x_2 = x_1 + h$$
$$\vdots$$
$$x_n = x_{n-1} + h = b$$

Example: Evaluate $\int_0^1 \frac{dx}{1+x^2}$ using Trapezoidal rule. [YT: Grajendra Purohit]

Sol:- $a=0,\ b=1$

let, $n=6$, $\Rightarrow h = \frac{b-a}{n} = \frac{1-0}{6} = \frac{1}{6}$, $y = \frac{1}{1+x^2}$

$x_0 = 0\ (a)$	$Y_0 = \frac{1}{1+0} = 1$
$x_1 = 0+\frac{1}{6} = \frac{1}{6}$	$Y_1 = \frac{1}{1+x_1^2} = \frac{36}{37}$
$x_2 = 2/6$	$Y_2 = \frac{1}{1+(2/6)^2} = 0.9$
$x_3 = 3/6$	$Y_3 = 0.8$
$x_4 = 4/6$	$Y_4 = 9/13$
$x_5 = 5/6$	$Y_5 = 36/31$
$x_6 = 1\ (b)$	$Y_6 = \frac{1}{2} = 0.5$

Now, using trapezoidal formula -

$$\int_a^b f(x)dx = h\left[\frac{Y_0+Y_N}{2} + Y_1+Y_2+Y_3+\cdots+Y_{N-1}\right]$$

$$= h\left[\frac{Y_0+Y_6}{2} + Y_1+Y_2+Y_3+Y_4+Y_5\right]$$

$$= \frac{1}{6}\left[\frac{3}{4} + \frac{36}{37} + \frac{9}{10} + \frac{4}{5} + \frac{9}{13} + \frac{36}{31}\right]$$

$$\boxed{\int_a^b f(x)dx = 0.7833}$$ Ans.

2.) **Simpson's $\frac{1}{3}$ Rule:** It divides the interval into an even no. of sub-intervals and uses parabolic segments to approximate the area under the curve.

- More accurate than trapezoidal method for smooth functions.

Formula:

$$\int_a^b F(x)\,dx = \frac{h}{3}\left[(Y_0+Y_n) + 4\underbrace{(Y_1+Y_3+Y_5+\cdots)}_{\text{odd}} + 2\underbrace{(Y_2+Y_4+Y_6+\cdots)}_{\text{even}}\right]$$

N = must be even.

3.) **Simpson's 3/8 Rule**: This method is useful when number of intervals is not divisible by 2 and divisible by 3. i.e. Subintervals is odd or multiple of 3.

- Less accurate than Simpson's 1/3 Rule.

Formula:

$$\int_a^b F(x)\,dx = \frac{3h}{8}\left[(Y_0+Y_N) + 3(Y_1+Y_2+Y_4+\cdots) + 2\underbrace{(Y_3+Y_6+\cdots)}_{\text{multiple of 3}}\right]$$

Imp: If we have to solve a question by all three methods then, let value of $n = 6$.

Example: Evaluate $\int_0^1 \frac{dx}{1+x^2}$ using Simpson's 1/3 and 3/8 rule.

Let, $n=6$; $\Rightarrow h = \frac{b-a}{6} = \frac{1-0}{6} = \frac{1}{6}$; $y = \frac{1}{1+x^2}$

YT: Gajendra Purohit

$x_0 = 0$
$x_1 = 1/6\ (x_0+h)$
$x_2 = 1/3\ (x_1+h)$
$x_3 = 1/2\ (x_2+h)$
$x_4 = 2/3\ (x_3+h)$
$x_5 = 5/6\ (x_4+h)$
$x_6 = 1 = b$

$Y_0 = \frac{1}{1+0^2} = 1$
$Y_1 = 1/1+(Y_6)^2 = 36/37$
$Y_2 = 9/10$
$y_3 = 4/5$
$y_4 = 9/13$
$y_5 = 36/31$
$y_6 = 1/2 = y_N$

⇒ Using Simpson's 1/3 formula :-

$$\int_a^b f(x)dx = \frac{h}{3}\left[(y_0+y_N) + 4(y_1+y_3+\cdots) + 2(y_2+y_4+\cdots)\right]$$

$$= \frac{h}{3}\left[(y_0+y_6) + 4(y_1+y_3+y_5) + 2(y_2+y_4)\right]$$

$$= \frac{1}{18}\left[\frac{3}{2} + 4(2.3631) + 2(1.5923)\right]$$

$$= 0.785$$

⇒ Using Simpson's 3/8 rule formula :-

$$\int_a^b f(x)dx = \frac{3h}{8}\left[(y_0+y_N) + 3(y_1+y_2+y_4) + 2(y_3+y_6+\cdots)\right]$$

$$= \frac{3h}{8}\left[(y_0+y_6) + 3(y_1+y_2+y_4) + 2(y_3)\right]$$

$$= \frac{1}{16}\left[3/2 + 3(3.1554) + 2(0.8)\right]$$

$$= 0.7853$$

4) **Gaussian Quadrature :** This method selects the specific points (Gauss points) and their associated weights to maximize accuracy.

- Gives accurate results with fewer calculation.

Steps :→ 1) Change interval $\int_a^b f(x)dx$ [a,b] into [-1,1] using

$$x = \frac{b-a}{2}u + \frac{b+a}{2}$$

2) Now, substitute the value of x and dx in $f(x)dx$ to get $\int_{-1}^{1} g(u)du$, where $g(u)du$ is the update function.

3.) Now, apply the formula:- for given 'n':

$$\int_{-1}^{1} g(u)\,du = w_1 g(u_1) + w_2 g(u_2) + \cdots + w_n g(u_n)$$

4) Take the values of w_i and u_i for given 'n' from below table:→

n	w_i	u_i
1	$w_1 = 2$	$u_1 = 0$
2	$w_1 = w_2 = 1$	$u_1 = -1/\sqrt{3}$, $u_2 = 1/\sqrt{3}$
3	$w_1 = 5/9$, $w_2 = 8/9$, $w_3 = 5/9$	$u_1 = -\sqrt{3/5}$, $u_2 = 0$, $u_3 = \sqrt{3/5}$
4	$w_1 = w_4 = 0.34785$ $w_2 = w_3 = 0.65215$	$u_1 = -u_4 = -0.86114$ $u_2 = -u_3 = 0.33998$

Que) $f(x) = \int_0^1 x^2\,dx$, apply Gaussian Quadrature for $n = 2$.

Sol:- Change the interval $[a,b] = [0,1]$ to $[-1,1]$ ⇒

$$x = \frac{b-a}{2} \times u + \frac{b+a}{2} = \frac{1-0}{2} \times u + \frac{1+0}{2} = \frac{u}{2} + \frac{1}{2}$$

$$\therefore \boxed{x = \left(\frac{u+1}{2}\right)}, \quad \frac{d(x)}{du} = \frac{d}{du}\left(\frac{u+1}{2}\right)$$

$$\Rightarrow 1 = \frac{1}{2}\frac{du}{dx} \Rightarrow dx = \frac{du}{2}$$

Now, Substitute x and dx in $f(x)dx$ to get $g(u)\cdot du$

$$\int_{-1}^{1} g(u)\,du = \int_{-1}^{1} \left(\frac{u+1}{2}\right)^2 \cdot \frac{du}{2} = \frac{1}{8}\int_{-1}^{1} (u+1)^2 \cdot du$$

⇒ applying formula for $n=2$.

$$\int_{-1}^{1} g(u)\cdot du = W_1 g(u_1) + W_2 g(u_2)$$

$$= 1 \times \frac{1}{8}\left(\frac{-1}{\sqrt{3}}+1\right)^2 + 1 \times \frac{1}{8}\left(\frac{1}{\sqrt{3}}+1\right)^2$$

$$= \frac{1}{8}\left[1+\frac{1}{3}+1+1/3\right] = \frac{1}{8}\left[2+\frac{2}{3}\right] = \frac{1}{4}\left[4/3\right] = 1/3$$

$$\frac{1}{3} = \boxed{0.333}$$

General Formula ; Apply in step 3 after Transformation :→

One-Point ($n=1$) ⇒ $\int_{-1}^{1} g(u)\,du = 2\,g(0)$

Two-point ($n=2$) ⇒ $\int_{-1}^{1} g(u)\,du = g\left(\frac{-1}{\sqrt{3}}\right) + g\left(\frac{1}{\sqrt{3}}\right)$

Three-point ($n=3$) ⇒ $\int_{-1}^{1} g(u)\,du = \frac{1}{9}\left[5\,g\left(-\sqrt{3/5}\right) + 8\,g(0) + 5\,g\left(\sqrt{\frac{3}{5}}\right)\right.$

★ Ordinary Differential Equations (ODE):

It is an equation involving a function of one independent variable and its derivative. The goal is to determine the unknown function that satisfies the given relationship between the function itself and its derivatives.

General Form: $F(t, y, \frac{dy}{dt}, \frac{d^2y}{dt^2}, \cdots, \frac{d^ny}{dt^n}) = 0$

→ Order of an ODE:- Highest derivative present in the eqⁿ.

→ Degree of an ODE:- Power of highest derivative.

✱ Linear ODE	Non-Linear ODE
i) The funcⁿ & it's derivative appears to the power of 1 & are not multiplied together.	→ The funcⁿ & its derivative appears with power other than 1 and are multiplied together.
ii) The degree of $f(x)$ and $f'(x)$ is 1.	→ can be higher than 1.
iii) Superposition of solutions is valid.	→ does not hold.
Applicⁿ: Electrical circuits, heat conduction.	→ Fluid dynamics, population models.

✱ Types of ODE:

i) 1st Order ODEs: These involves only 1st derivatives of funcⁿ.

$$\frac{dy}{dx} = f(x, y) \quad : \text{General Form}$$

ii) 2nd Order ODEs: involves upto second derivatives:

$$\frac{d^2y}{dx^2} + P(x)\frac{dy}{dx} = g(x)$$

iii) 3rd order ODEs: involves derivatives of order 3 or higher:

$$y^n = f(x, y, y', y'', \cdots y^{n-1}), \quad y' = \frac{dy}{dx}$$

DE

Partial DE

→ If any differential equⁿ. contains more than 1 independent variable then its is known as PDE.

Eg: $\frac{\partial u}{\partial x} + \frac{\partial u}{\partial y} = 0$

Ordinary DE

★ Initial & Boundary value problems in ODEs:

i) IVP: Initial Value Problem, involves solving an ODE with condition specified at a single point; typically at the begining of interval (initial point.).

Eg: $\frac{dy}{dx} = f(x,y)$; $y(x_0) = y_0$

ii) BVP: It involves solving an ODE with condition specified at the boundaries (usually more than two points) of the interval.

Eg: $\frac{d^2y}{dx^2} = f(x,y)$; $y(a) = A$, $y(b) = B$

✳ Methods to Solve IVP and BVP:

i) Analytics Method: These involves finding a closed-form solution to ODEs. These include techniques
↳ Seperation of variable, IF Method, Series Solution.

ii) Numerical Method: When analytics sol^{n}s are difficult or impossible to find, numerical method provide approximate solution →
- Euler Method
- Runge-Kutta
- Finite difference Method.

★ Euler's Method

It is a simple, 1st order numerical procedure for solving ODEs.

- It provides a straight forward approach to approximate the solution of IVP (Initial value Point).
- Euler's method is Runga-Kutta method of First-order.

Iterative Formula:- $Y_{n+1} = Y_n + h \cdot f(x_n, y_n)$ (h: step size; $f(x_n, y_n)$: value of $\left(\frac{dy}{dx}\right)_{x_n, y_n}$)

$$h = \frac{\text{Final value} - \text{Initial value}}{\text{step}}$$

Note:→ if step is not given, take **step = 5**.

Ques:→ Solve $\frac{dy}{dx} = x+y$ with boundary cond^n $Y=1$ at $x=0$.
Find approx value of y for $x=0.1$. [YT: Gajendra Purohit]

Sol:- let, step = 5 $\Rightarrow h = \frac{0.1-0}{5} = 0.02$

$f(x,y) = x+y$, $Y_{n+1} = Y_n + h(x_n + y_n)$

$X_0 = 0$	at $x=0, y=1$
$X_1 = 0+h = 0.02$	$\therefore Y_0 = 1$
(X_1+h) $X_2 = 0.04$	put $n=0 \Rightarrow$
(X_2+h) $X_3 = 0.06$	$Y_1 = Y_0 + h(x_0+Y_0) = 1+0.02(0+1) = 1.02$
(X_3+h) $X_4 = 0.08$	put $n=1 \Rightarrow$
$X_5 = 0.1$	$Y_2 = 1.02 + 0.02(0.02+1.02) = 1.0408$
	put $n=2 \Rightarrow$
	$Y_3 = Y_2 + h(x_2+Y_2) = 1.0624$

put $n=3 \Rightarrow$

$y_4 = y_3 + h(x_3+y_3) = 1.0848$

put $n=4 \Rightarrow$ $\boxed{y_5 = 1.1081}$

* Euler's Modified Method (Runge-Kutta Method of Second Order.)

$$y^*_{n+1} = y_n + h \cdot f(x_n+y_n)$$

$$y_{n+1} = y_n + \frac{h}{2}\left[f(x_n, y_n) + f(x_{n+1}, y^*_{n+1})\right]$$

* Advantages	Disadvantages
• Simplicity	• Low accuracy.
• Low Cost	• Step size dependency.
• Minimum Computation.	• Not adaptive.

★ Runge-Kutte Method (More Accurate)

These are family of iterative methods that provide higher accuracy than Euler's method.

- Most commonly used is fourth-order Runge-kutta method (RK4).
- Used for numerically solving ODEs.
- Used for both linear & Non-linear ODEs.

Algorithm:

i) Initial Value: Start with $f(x_0) = y_0$.

ii) Slope Evaluation: K: increment factor w.r.t. y-axis.

$K_1 = h \cdot f(x_n, y_n)$

$K_2 = h \cdot f\left(x_n + \frac{h}{2}, y_n + \frac{K_1}{2}\right)$

$K_3 = h \cdot f\left(x_n + \frac{h}{2}, y_n + \frac{K_2}{2}\right)$

$K_4 = h \cdot f(x_n + h, y_n + K_3)$

iii) Combine the slopes:

$$K = \frac{1}{6}[K_1 + 2K_2 + 2K_3 + K_4]$$

The final value of 'y' at next step $x_{n+1} = x_n + h$ is:

$$y_{n+1} = y_n + K$$

Ques: Given, $\frac{dy}{dx} = x + y^2$, $y(0) = 1$. find $y(0.2)$ when $h = 0.1$ using RK of order 4. [YT: Gajendra Purohit]

Sol:- $f(x,y) = x + y^2$

$X_0 = 0$	$Y_0 = 1$
$X_1 = 0.1$	$Y_1 = 1.1165$
$X_2 = 0.2$	

put $n = 0$:

$K_1 = h\,f(x_0, y_0) = h(x_0 + y_0^2) = 0.1\,(0+1) = 0.1$

$K_2 = h\,f\left(x_0 + \frac{h}{2}, y_0 + \frac{K_1}{2}\right) = h\,f\left[(x_0 + h/2) + \left(y_0 + \frac{K_1}{2}\right)^2\right] = 0.11525$

$K_3 = h\,f\left(x_0 + \frac{h}{2}, y_0 + \frac{K_2}{2}\right) = h\,f\left[(x_0 + h/2) + \left(y_0 + \frac{K_2}{2}\right)^2\right] = 0.1169$

$K_4 = 0.1347$

$\Rightarrow K = \frac{1}{6}[0.1 + 2(0.11525) + 2(0.1169) + 0.1347]$

$= 0.1165$

$$Y_1 = Y_0 + K = 1 + 0.1165 = 1.1165$$

∴ Continue for $n = 1$, find K then Y_2..

UNIT: 3

NMOUP

Dashrath Nandan.

classmate Date Page

★ Optimization

Optimization involves finding the best possible solution to a problem from a set of feasible options to achieve the desired objectives.

* Key components of Optimization :

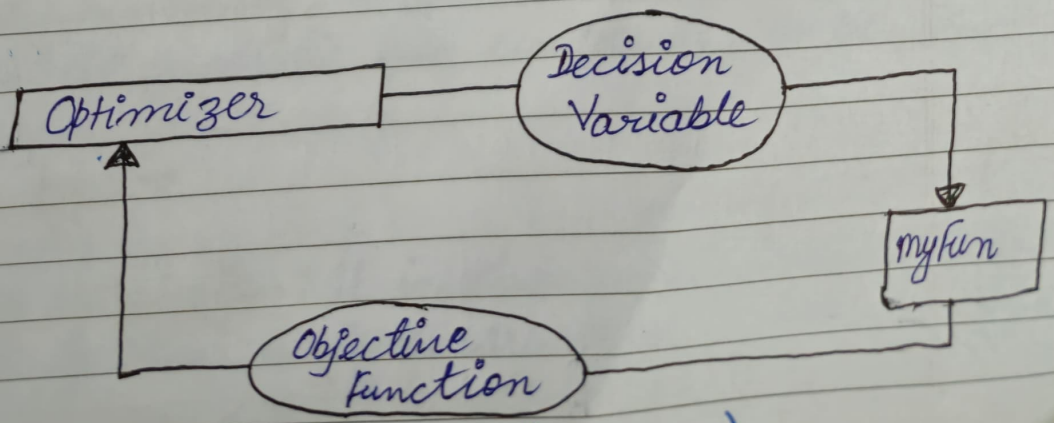
i) **Objective Function** :- It is the function that needs to be optimized or either maximized or Minimized

ii) **Decision Variable** :- These are the variables that can be adjusted to influence the optimization process.

iii) **Constraints** :- Constraints are restrictions or limitations that must be satisfied during the optimization process.

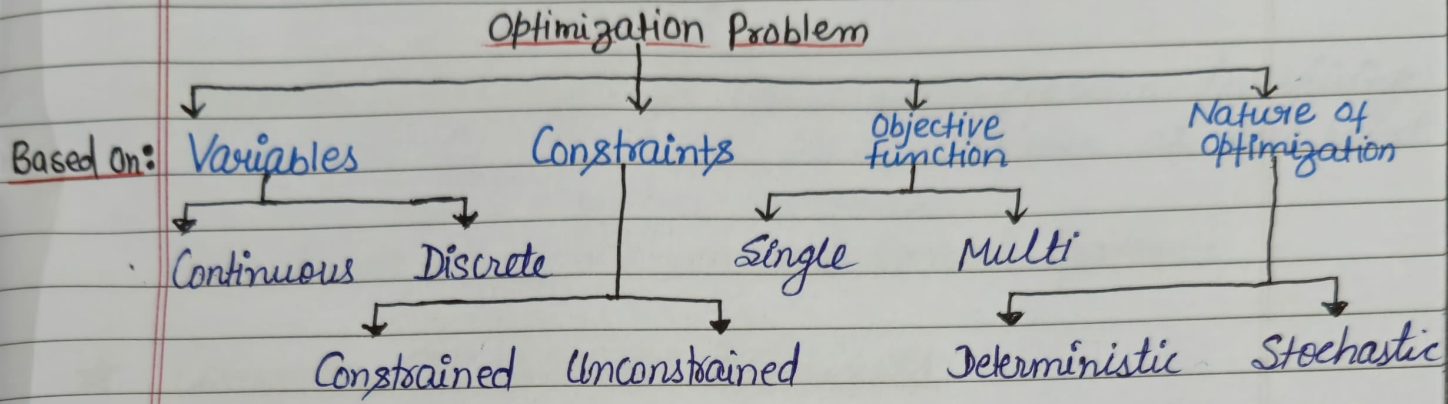
* Applications :-

i) Engineering - Designing efficient system and structures.
ii) Data Science - Tuning Machine learning Models.
iii) Economics & Finance - Protfolio optimization, cost minimization.
iv) Healthcare - Optimizing treatment plans.



Result = Optimizer (myfun, x_0)

★ Classification of Optimization Problems:



* Type of Optimization Problem based on both Objective & Constraints:

1) Linear Optimization (Programming): In this, both the objective function and the constraints are linear.
- Obj. func^n can be expressed as combination of decision variables
- Applied in - Resource allocation, production planning, etc.
- Ex: Minimizing the cost of material subject to production constraints.

2) Non linear Optimization: It deals with problem where objective function or at least one constraint is non-linear.
- Engineering design, Economic modeling.

Ex: Maximizing profit with non-linear cost function.

3) Quadratic Programming: A special case of NLP, where Obj. func^n is quadratic but constraints can be linear.

Other types →

- Integer Optimization: It involves problems where some or all decision variables are restricted to integer values.

Ex: Scheduling, capital budgeting, facility location.

- **Mixed-Integer Programming (MIP):-** It involves problems that includes both integer & continuous decision variable.

- **Dynamic Optimization:** It deals with problem involving time-dependent processes, where decision at one point in time affect future outcomes.
  Ex: Economics, Finance and Control System.

★ Common Optimization Techniques

1) Gradient Descent.
2) Genetic Algorithms.
3) Simulated Annealing.
4) Particle Swarm Optimization.
5) Simplex Method: It is a popular algorithm for solving Linear Programming (LP) problems.

★ Optimization in Python

Python is a versatile and powerful programming language widely used for optimization due to its rich ecosystem of libraries and tools.

* Key Libraries for Optimization in Python :-

i) SciPy: It is a fundamental lib. for scientific computing in python. It includes modules for optimization, integration, interpolation, etc.

ii) NumPy: It is essential for numerical computing, involves support for arrays, matrices, etc.

iii) Pandas: Powerful lib. for data manipulation & analysis.

iv) Scikit-Optimize: It a lib. for Sequential model-based optimization.

v) TensorFlow and PyTorch: These deep learning lib. include optimization modules for training ML models.

Example: Non-linear Optimization with Scipy →

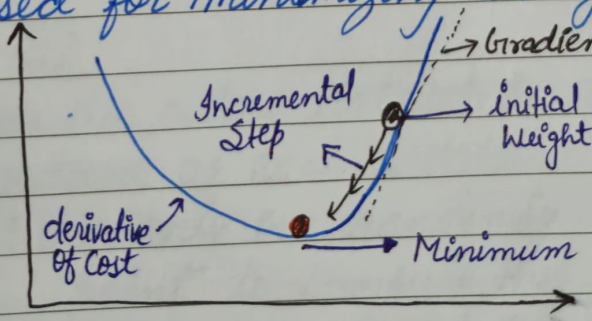
```
from scipy.optimize import minimize
def objective(x):
    return x[0]**2 + x[1]**2
x0 = [1, 1] // Initial Guess
result = minimize(objective, x0) // Solve the problem
print(f"Optimal value: {result.fun}")
print(f"Optimal solution: {result.x}")
```

★ **Unconstrained Optimization: Gradient-based Methods**

Unconstrained optimization focuses on finding the optimal solution to a problem without any restriction or constraints on the variables.

✱ **Gradient Descent**

Gradient Descent is an iterative optimization algorithm commonly used for minimizing an objective function.



Formula: $x_{K+1} = x_K - \alpha \cdot \nabla f(x_k)$ α: Learning Rate

* Key concept of Gradient Descent :

i) Objective function :- The function $f(x)$ that we aim to minimize.

ii) Gradient :- The gradient of $f(x)$, denoted as $\nabla f(x)$, is a vector of partial derivatives w.r.t. all variables. It point to the direction of the steepest increase of function.

iii) Learning rate (α) :- It is a positive scalar that determines the step size in each iteration.

iv) Iteration : The process of updating the variables by moving in direction opposite to the gradient to find the min^{n} of func^{n}.

* Algorithm :

1. Initialize : Start with an initial guess x_0 for the variable.
2. Compute Gradient : Calculate the gradient $\nabla f(x_k)$ at the current point x_k.
3. Update : Update the variables :

$$x_{k+1} = x_k - \alpha \nabla f(x_k)$$

4. Iterate : Repeat the process until convergence i.e. until the changes in x $(x_{k+1} - x_k)$ becomes negligible.

* Variants of GD :

1) Batch GD : Uses the entire dataset to compute the gradient at each iteration.

2) Stochastic GD : Uses only one randomly selected data point to compute the gradient.

3) Mini-batch GD : A compromise between batch and stochastic GD.

* **Example:** $f(x) = (x-2)^2$, $x_0 = 0$, $\alpha = 0.1$, iteration = 100

Code

```
import numpy as np
def f(x):
    return (x-2)**2
def grad_f(x):
    return 2*(x-2)
def gradient_descent(x0, α, itr):
    X = x0
    for _ in range(itr):
        gradient = grad_f(x)
        X = X - α * gradient
    return X

x0 = 0
α = 0.1
itr = 100
Optimal_x = gradient_descent(x0, α, itr)
print(f"Optimal x: {Optimal_x}")
```

- $f(x) = (x-2)^2$
- $f'(x) = 2(x-2)$ [gradient]

➡ Initial guess, $x_0 = 0$, $\alpha = 0.1$

Using formula –

$$x_{k+1} = x_k - \alpha \cdot \nabla f(x_k)$$

$\Rightarrow x_1 = 0 - (0.1)(2\times0 - 4)$
$= 0.4$

Next Step:
$x_2 = 0.4 - (0.1)(2\times0.4 - 4)$
$= 0.72$

Next itr^n:
$x_3 = 0.72 - (0.1)(2\times0.72 - 4)$
$= 0.976$

Next itr^n:
$x_4 = 0.976 - (0.1)(2\times0.976 - 4)$
$= 1.180$

repeate until $(x_{k+1} - x_k)$ becomes negligible.

* **Advantages:**
Efficiency , Simplicity , Scalability.

* **Challenges:**
- Choosing the Learning rate.
- Local minima : For non-convex func^n, gradient descent can get stuck in local minima.
- Gradient Vanishing / Exploding.

★ Constrained Optimization

It involves finding the best solution to a problem within a set of restrictions or constraints.

- These constraints can represent real-world limitations such as resource availability, time, budget, etc.

✱ Key Concepts:

i) Objective Function
ii) Constraints
iii) Feasible Region: Set of all points that satisfy the constraints
iv) Decision Variable

✱ Type of Constraints:

Mathematical formulation:→

A typical constrained optimized problem:

minimize $f(x)$

Subject to $g_i(x) = 0, \quad i = 1, \cdots, m$

$h_j(x) \leq 0, \quad j = 1, \cdots, p$

where,
$f(x)$: Objective func^n
$g_i(x)$: are Equality Constraint
$h_j(x)$: Inequality Constraints
x: is vector of decision Variable

1) **Equality Constraints:** These constraints requires that a certain function equals a specific value.

$$\boxed{g_i(x) = 0} \rightarrow \text{representation}$$

Ex: i) **Resource allocation:** If $x_1, x_2, \cdots, x_n$ represents hours allocated to n tasks, then constraint is:

$$x_1 + x_2 + x_3 + \cdots + x_n = \text{Total hours.}$$

ii) Chemical Mixture.
iii) Circuit design.

2) Inequality Constraints: These constraints requires that certain func^n. is either $<$ or $=$, $>$ or $=$ a specific value.

$$h_i(x) \leq 0 \quad \text{or} \quad h_i(x) \geq 0$$

Ex: • Budget Constraints: If $C_1, C_2, \ldots C_n$ represents the cost of 'n' projects, and B is the budget, then constraint is -

$$C_1 + C_2 + C_3 + \cdots + C_n \leq B$$

• Production limits:

* Methods for Solving Optimization Problems with Constraints

- Lagrange Multipliers.
- Linear Programming (LP).
- Quadratic Programming (QP).

* Application:

• Economics, Engineering, ML, Operation Research.

★ Lagrange Multipliers

Lagrange multipliers are mathematical tool used in optimization to find the maximum or minimum of a function $F(x,y,z)$ subject to a constraint of the form $G(x,y,z)=0$.

* Algorithm:

To optimize a function $f(x,y,z)$ subject to a constraint $g(x,y,z,\ldots)=0$

1. **Formulate the Lagrangian:** Write Lagrangian function by combining the original func^n. and constraint using a Lagrange multiplier, λ.

$$\boxed{L(x,y,\lambda) = f(x,y) - \lambda g(x,y)}$$

2. **Compute the Partial Derivatives:** Find the partial derivative of lagrangian with respect to each variable and λ.

$$\frac{\partial L}{\partial x} = 0 \;,\; \frac{\partial L}{\partial y} = 0 \;,\; \frac{\partial L}{\partial \lambda} = 0$$

3. **Solve the system of Equations:** Solve the resulting system of equation to find the values of x, y and λ.

4. **Verify the Solution:** Check the sol^n to ensure they satisfy the original constraint, and determine if they are maxima, minima or sadde points.

Example: Maximize the func^n $f(x,y) = x+y$ subject to constraint $g(x,y) = x^2+y^2-1 = 0$

Sol:- **Step 1:** Create the Lagrange function.

$$L(x,y,\lambda) = (x+y) - \lambda(x^2+y^2-1)$$

Step 2: Find partial derivative w.r.t. x, y and λ.

$$\frac{\partial L}{\partial x} = 1 - 2\lambda x = 0 \Rightarrow \lambda = \frac{1}{2x} \quad \text{---- (I)}$$

$$\frac{\partial L}{\partial y} = 0 \Rightarrow 1 - 2\lambda y = 0 \Rightarrow \lambda = \frac{1}{2y} \quad \text{---- (II)}$$

$$\frac{\partial L}{\partial \lambda} = 0 \Rightarrow 0 - (x^2+y^2-1) = 0 \Rightarrow x^2+y^2 = 1 \quad \text{---- (III)}$$

Step3: Solve the system of equations:

from ① & ② ⇒ $x = y$.

now, from ③ ⇒ $x^2 + x^2 = 1 \Rightarrow 2x^2 = 1 \Rightarrow x = \pm\sqrt{\frac{1}{2}}$

$$\therefore \boxed{x = y = \pm\frac{1}{\sqrt{2}}}$$

Step4: Find the maximum function.

$$f(x,y) = x + y$$

- At $\left(\frac{1}{\sqrt{2}}, \frac{1}{\sqrt{2}}\right)$, $f(x,y) = \sqrt{2}$
- At $\left(\frac{-1}{\sqrt{2}}, \frac{-1}{\sqrt{2}}\right)$, $f(x,y) = -\sqrt{2}$
- At $\left(\frac{1}{\sqrt{2}}, \frac{-1}{\sqrt{2}}\right)$ or $\left(\frac{-1}{\sqrt{2}}, \frac{1}{\sqrt{2}}\right)$, $f(x,y) = 0$

⇒ Maxima of $f(x,y) = \sqrt{2}$
Minima of $f(x,y) = -\sqrt{2}$

* **Application:** Economics, Engineering, ML, Physics.

★ Linear Programming (LP)

LP is a mathematical technique used for optimizing a linear objective function, subject to a set of linear inequality and equality constraints.

- Used in economics, business, engineering, etc.

Maximize or Minimize $\rightarrow Z = C_1x_1 + C_2x_2 + \dots + C_nx_n$

Components of LP

Objective Function
$Z = c_1 x_1 + \cdots + c_n x_n$

Constraints
$a_{11} x_1 + \cdots + a_{1n} x_n \leq b_1$
$a_{m1} x_1 + \cdots + a_{mn} x_n \leq b_m$

Non-negative Restrict?
$x_i \geq 0 \; \forall \; i$

* Steps to Solve LP:

1. Mark the decision variable in the Problem.
2. Build the objective fun: of the problem & check it needed to minimized or maximized.
3. Write down all the constraints of LP.
4. Ensure non-negative restrictions of decision variable.
5. Now solve LP, using either Simplex or graphical method.

* Methods to Solve LP (Geeks for Geeks)

i) **Simplex Method:** An iterative algorithm that moves from one vertex to adjacent vertex of the feasible region until an optimum solution is achieved.

ii) **Graphical Method:** Suitable for problems with two decision variables. It involves plotting the constraints on a graph, identifying the feasible region, & finding optimal sol^n.

Ex: Find the maximal and minimal value of $Z = 6x + 9y$, when constraint conditions are:

- $2x + 3y \leq 12$
- $x + y \leq 5$
- $x, y \geq 0$

Sol^n:

Step 1: Convert the inequation into normal Equation:

$2x + 3y = 12$, $x + y = 5$, $x = 0, y = 0$

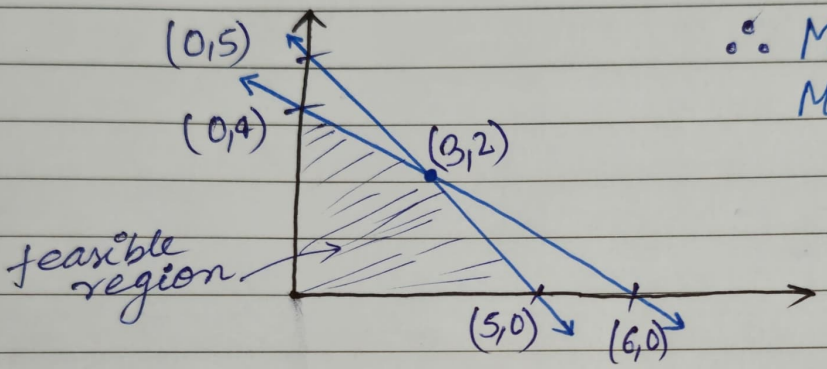
Step 2: Find x-intercept ($y = 0$) and y-intercept ($x = 0$)

Step3: Find the optimal point, where 2 lines intersecting each other. i.e. At (3,2).

Step4: For $x \geq 0$, $y \geq 0$, we find that both inequ^n. are followed

Step5: Find Z for each point and maxima & minima.

Coordinates	(0,5)	(5,0)	(0,4)	(6,0)	(3,2)
$Z = 6x + 9y$	45	30	36	36	36



∴ Maxima at (0,5)
Minima at (5,0)

Advantages

- Provides precise, optimal sol^n.
- Efficiently handle large and complex problems.

Limitations:

- Not suitable for non-linear relationships.

★ **Quadratic Programming (QP)**

It is an optimization technique used to find the minimum or maximum of a quadratic objective function subject to linear constraints.

Components of QP

- Quadratic objective function: $f(x) = \frac{1}{2} x^T Q x + c^T x$
- Linear Constraints: $Ax \leq b$, $Ex = d$
  - A & E are matrices of coefficient
- Non-negativity Constraints: $x \geq 0$, $l \leq x \leq u$
  - l: Lower Bound
  - u: Upper Bound

Quadratic objective function :-

Minimize / Maximize , $f(x) = \frac{1}{2} x^T Q x + c^T x$	Q: Covariance matrix c: Vector of coefficient x: Vector of decision Variable

* Methods to Solve QP:

i) Gradient-Based Methods.

ii) Interior-Point Methods: These works within the feasible region & approach the optimal sol? iteratively.

iii) Active-Set Methods: These identify & work with a subset of constraints that are active at optimal sol?.

iv) Conjugate Gradient Method: These are iterative methods particularly effective for solving large system of linear Eq?.

★ Metaheuristic Optimization Algorithm

These are a subset of optimization algorithms designed to find optimal or near-optimal solutions to complex problems within a reasonable timeframe.

* Key Concepts:

1. Optimization Problem:
   Objective func?: The Function that needs to be minimized or Maxim?
   Constraints:
   Search Space: The domain of possible solutions.

2. Metaheuristic: designed to explore search space efficiently and are commonly used when traditional optimization techniques are impractical.

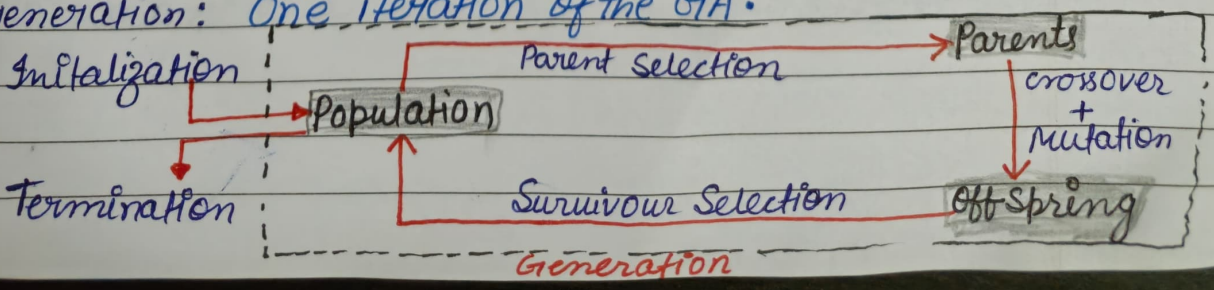
* Metaheuristic Algorithms:

1. Genetic Algorithm (GA)
2. Simulated Annealing (SA).
3. Particle Swarm Optimization (PSO).
4. Ant Colony Optimization (ACO) :- It pheromone trails to guide the search towards promising areas of the search space.
5. Tabu Search (TS):- Uses memory structures that describe the visited solutions. Uses a tabu list to avoid revisiting recently explored solutions.
6. Differential Evolution (DE):
   - Operates on population and uses differential mutation.
   - Efficient for continous optimization problems.
7. Harmony Search (HS):
   Inspired by the musical improvisation process.

1. Genetic Algorithms

Inspired by the principles of natural selection & genetics. used to find approximate solutions to complex problems.

- Population: A set of potential sol.ⁿ to the optimization problem.
- Chromosome: Represent a solution to the problem.
- Gene: A component of a chromosome.
- Fitness Function: Measure two or more solutions.
- Crossover: Combines parts of two parent's chromosome.
- Mutation: Randomly alters gene in chromosome.
- Generation: One iteration of the GA.



* Algorithm :

1. Initialization :- Generate an initial population of individuals.
2. Evaluation :- Compute the fitness of each individual in populat^{n}.
3. Selection :- Select Individual based on their fitness .
4. Crossover :- Apply crossover to the selected parents to produce offspring.
5. Mutation :- Apply mutation to offspring with a low probability.
6. Replacement :- Form a new population by replacing old popul^{n}.
7. Termination :- Stop the algorithm after population is conuerged to a satisfactory solution.

* Key operators :

1. Selection Operators :- Roulette Wheel Selection , Tournament Selection , Rank-Based Selection .

2. Crossover Operators :- Single-point or Multi-point Crossover , Uniform Crossover .

3. Mutation Operators :- Bit-Flip Mutation , Swap Mutation Gaussian Mutation .

* Advantages :

Robustness , flexible , can deal with both discrete and continuous variables.

* Disadvantages :

- May require extensive computational resources.
- No gurantee of finding the global optimum.

2. Simulated Annealing

SA is a probabilistic optimization technique inspired by the annealing process in metallurgy, where materials are heated and then slowly cooled to remove defects and achieve a more stable structure.

Annealing Process: Heating and Cooling.
Metropolis Criterion: A probabilistic rule to decide whether to accept a new solution.

* Algorithm:

1. **Initialization:**
   Start with an initial sol^n. Set initial temp. (high) and cooling schedule.

2. **Iteration:**
   - Generate a neighbouring solution.
   - Calculate the change in objective function value (ΔE)
   - Apply Metropolis criterion to decide to accept or not new sol^n.
   - Update the current sol^n. if new solution is accepted.
   - Reduce the temp. according to cooling schedule.

3. **Termination:**
   Stop the algorithm when the system is sufficiently "Frozen".

- Acceptance Probability, $P(\Delta E, T) = e^{-\left(\frac{\Delta E}{T}\right)}$,
  ΔE: Change in objective func^n. value, T: Current Temp.

* Advantage

- Simple
- Flexible

Disadvantage

- Performance depend on cooling schedule.
- Slow.
- No gurantee of finding global optimum

* **Pseudo Code:**

```
Set x = x0 ;   // Generate the initial solution.
Set T = Tmax ;   // Starting temp.
repeat
    repeat
        Generate a random neighbour x'
            ΔE = f(x') - f(x)
            If ΔE ≤ 0 then
                x = x' ;   // Accept neighbour soln.
            else
                Accept x' with probability e^(-ΔE/T) ;
                x = x' ;
            end if
    until (Equilibrium Condition)
    T = g(T) ;
until (Stoping Criteria Statisfied)
return x ;   // Best found soln.
```

3. Particle Swarm Optimization

- PSO is a population-based optimization algorithm inspired by social behavior of birds and fishes.
- A swarm can be defined as structured collection of interacting organisms.
- Used to find approximate solutions to complex optimization problem by simulating social dynamics of a swarm particles.

* Key Concepts:

- Particle :- Represent a potential solution to optimiz? problem.
- Swarm :- A collection of particles that moves through search space.
- Fitness Function :-
- Global Best (gBest) : The best position found by any particle in the entire swarm.
- Personal Best (pBest) : The best position found by a specific particle over its history.

* Algorithm:

1. **Initialization:**
   - Initialize a swarm of particles with random position & velocity.
   - Evaluate the fitness of each particle.
   - Identify initial pBest for each particle & gBest for swarm.

2. **Iteration:**
   Update velocity of each particle based on vel., its dist from pBest and gBest.
   Update the position of each particle using new velocity.
   Evaluate the fitness of each particle at its new position.
   Update pBest for particle and gBest for swarm if better pos. are found.

3. **Termination:**
   Stop after a fixed iterations or swarm has converged to a satisfactory sol^n.

* Key Equations:

a) $V_i(t+1) = \omega \cdot V_i(t) + c_1 \cdot r_1 \cdot (pBest - x_i(t)) + c_2 \cdot r_2 \cdot (gBest - x_i(t))$

Personal Influence

Social Influence

Velocity Update

$V_i(t)$ = Velocity of particle i at itr^n t

w: Inertia weight. r_1, r_2: Random number between 0 & 1.

C_1, C_2: Acceleration coefficient, control the influce of pBest & gBest.

pBest, gBest.

$x_i(t)$: Position of particle i at itr^n. t.

b.) position update:

$$x_i(t+1) = x_i(t) + V_i(t+1)$$

$x_i(t+1)$: New position of particle i.

$x_i(t)$: Current position.

$V_i(t+1)$: Updated velocity of particle i.

* Advantages:
- Simple, can handle non-linear optimiz^n. problem.
- Require few parameters to adjust.

* Disadvantages:
- No proper tuning.
- Require blancing btw^n exploration & exploitation.