- **Author : YASHWANTH**
- **DATE : 6/01/2023**

Problem: NEWTON_RAPHSON

- **Find square-root of 10:**

$$x = \sqrt{10} \tag{1}$$

- **Polynomial equation:**

$$[x^2 - 10 = 0 = f(x)] - -- > (1) \tag{2}$$

In [1]:
```
import numpy as np
import scipy as sp
import numpy.polynomial.polynomial as root
import math as mt
import pandas as pd
import matplotlib.pyplot as plt
```

- **Excat solutions:**
  - using: np.polynomial.polynomial.polyroots(coeff_array)
    - coeff_array = **(c, b, a)** for $ax^2 + bx + c = 0$
  - using math.sqrt() function

In [2]:
```
coeff = (-10,0,1)
excat_root = np.polynomial.polynomial.polyroots(coeff)
print(excat_root, mt.sqrt(10))
```

```
[-3.16227766  3.16227766] 3.1622776601683795
```

Defining function and its derivative:

Newton-Raphson Method:

- For single variable function:

$$x_(n+1) = x_n - \frac{f(x_n)}{f'(x_n)} \tag{3}$$

In [3]:
```
def fun(b):
    return (b**2-10)
def dfun(b):
    return (2*b)
```

Max_Iteration condition:

In [72]:
```
def SolveNS(p,n):
    store = [p]
    error = [np.abs(p-mt.sqrt(10))]
    temp  = 0
    for i in range(0,n,1):
```

```
        a1 = fun(p)
        a2 = dfun(p)
        p = p -(a1/a2)
        store.append(p)
        error.append(np.abs(p-mt.sqrt(10)))
    return (store,error)
```

Max_Error condition:

In [73]:

```
def solveNS(p,e_max):
    store = [p]
    error = [np.abs(p-mt.sqrt(10))]
    temp  = 0
    e=1
    while(e>e_max):
        a1 = fun(p)
        a2 = dfun(p)
        p = p -(a1/a2)
        store.append(p)
        e = np.abs(p-temp)
        temp = p
        error.append(np.abs(p-mt.sqrt(10)))
    return (store,error)
```

Solving:

Max_Iteration condition:

In [173…

```
intial_guess = 2
max_iter     = 5
ss, error = SolveNS(intial_guess,max_iter)
```

In [175…

```
error = np.array(error)
"""
Error_ratio = abs_error(current)/abs_error(previous)

"""
e1_ratio =  (error[1:len(error)]/error[0:(len(error)-1)])
e2_ratio =  (error[1:len(error)]/error[0:(len(error)-1)]**2)
e3_ratio =  (error[1:len(error)]/error[0:(len(error)-1)]**3)
# print(error.size,e_ratio.size)
# solution and error list contains initial_guess values.
print("Initial_guess:",ss[0],"  Initial_abs_error:",error[0])
dict1 = {"x_values":ss[1:],"Abs_errors": error[1:],
         "Error_Ratio^1": e1_ratio, "Error_Ratio^2": e2_ratio, "Error_Ratio^3": e3_ratio
df = pd.DataFrame(dict1,index=(np.arange(1,max_iter+1,1)))


print(df)
print("\nMAX_ITERATION_ERROR:",error[len(error)-1])
print("Converged sol:       ", ss[len(ss)-1])
print("Excat sol:           ", mt.sqrt(10))
```

```
Initial_guess: 2   Initial_abs_error: 1.1622776601683795
   x_values    Abs_errors  Error_Ratio^1  Error_Ratio^2  Error_Ratio^3
1  3.500000  3.377223e-01       0.290569       0.250000       0.215095
2  3.178571  1.629377e-02       0.048246       0.142857       0.423002
3  3.162319  4.176198e-05       0.002563       0.157303       9.654204
4  3.162278  2.757568e-10       0.000007       0.158112    3786.019373
5  3.162278  0.000000e+00       0.000000       0.000000       0.000000

MAX_ITERATION_ERROR: 0.0
```

```
Converged sol:        3.1622776601683795
Excat sol:            3.1622776601683795
```

- **Error_ratio** with $\alpha = 0$, only converged $\implies$ **order of convergence is 2**.
- **Note:**
  - All the ratio will abrubtly go to **zero** if the solution is converging and the **abs_error = 0.** But that does **not mean all ratio are converging**.
  - Observe that other two ratio never showed **signs of convergence!**

Max_Error condition:

In [178...

```
max_error = 1e-9
ss2, error2 = solveNS(intial_guess,max_error)
```

In [179...

```
error2 = np.array(error2)
e3_ratio = (error2[1:len(error2)]/error2[0:(len(error2)-1)])
e4_ratio = (error2[1:len(error2)]/error2[0:(len(error2)-1)]**2)
e5_ratio = (error2[1:len(error2)]/error2[0:(len(error2)-1)]**3)
print("Initial_guess:",ss2[0],"  Initial_abs_error:",error2[0])
dict2 = {"x_values2":ss2[1:],"Abs_errors": error2[1:],
         "Error_Ratio^1": e3_ratio, "Error_Ratio^2": e4_ratio, "Error_Ratio^3": e5_ratio
df2 = pd.DataFrame(dict2)

print(df2)
print("\nMAX_ERROR:",error2[len(error2)-1])
print("Converged sol: ", ss2[len(ss2)-1])
print("Excat sol: ", mt.sqrt(10))
```

```
Initial_guess: 2   Initial_abs_error: 1.1622776601683795
   x_values2    Abs_errors  Error_Ratio^1  Error_Ratio^2  Error_Ratio^3
0   3.500000  3.377223e-01       0.290569       0.250000       0.215095
1   3.178571  1.629377e-02       0.048246       0.142857       0.423002
2   3.162319  4.176198e-05       0.002563       0.157303       9.654204
3   3.162278  2.757568e-10       0.000007       0.158112    3786.019373
4   3.162278  0.000000e+00       0.000000       0.000000       0.000000

MAX_ERROR: 0.0
Converged sol:  3.1622776601683795
Excat sol:  3.1622776601683795
```

Plotting:

In [181...

```
%matplotlib inline
%config InlineBackend.fig_format = 'svg'
```

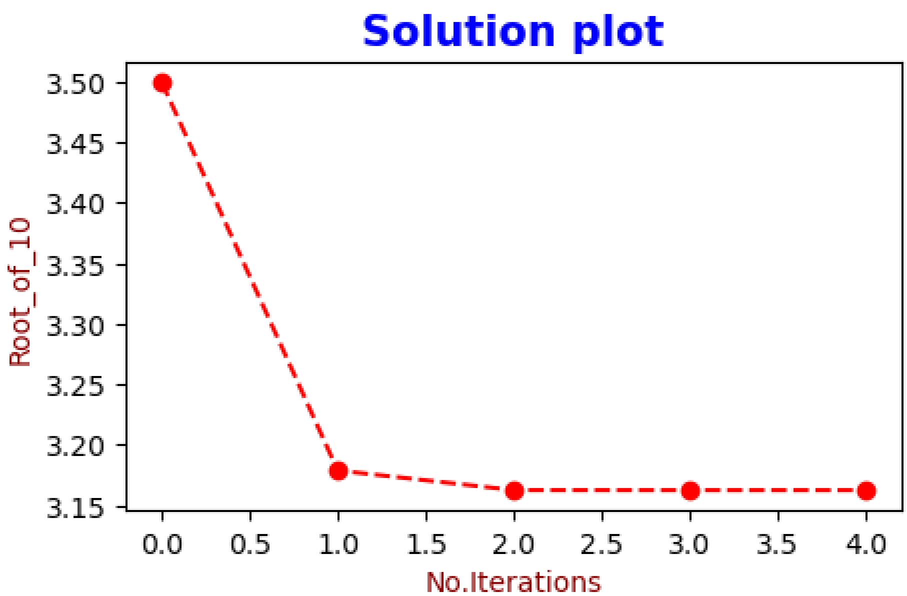
Max_Iteration condition:

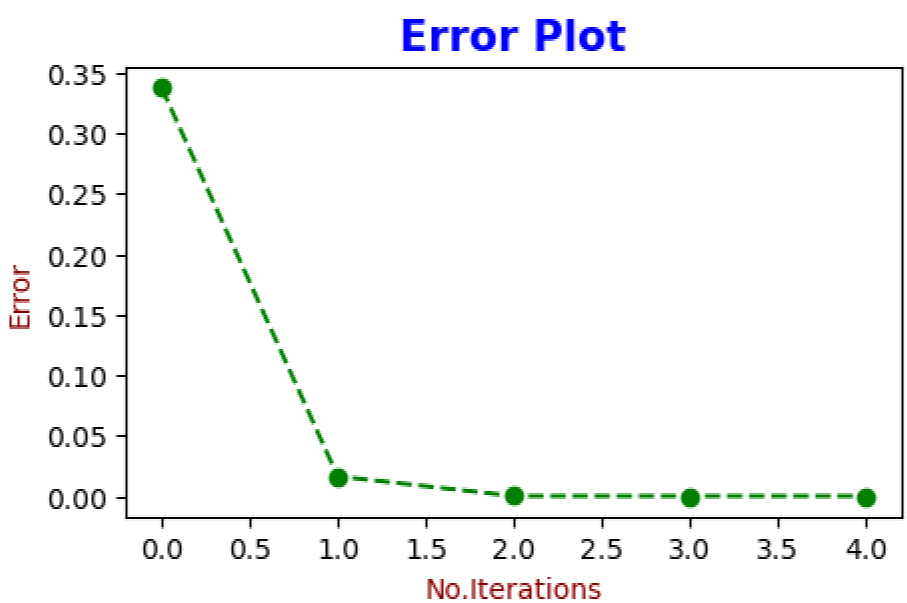
In [187...

```
plt.figure(figsize=(5,7))

# Solution-plot
fig1 = plt.subplot(2,1,1)
plt.plot(np.arange(0,max_iter,1),ss[1:], "r--o")
plt.title("Solution plot", fontweight ="bold", fontsize=15,color="b")
plt.ylabel("Root_of_10", fontsize=10,color="darkred")
plt.xlabel("No.Iterations", fontsize=10,color="darkred")
plt.subplots_adjust(hspace=0.4)

# Error-plot
plt.subplot(2,1,2)
plt.plot(np.arange(0,max_iter,1),error[1:],"g--o")
```

```
plt.title("Error Plot", fontweight ="bold", fontsize=15,color="b")
plt.ylabel("Error", fontsize=10,color="darkred")
plt.xlabel("No.Iterations", fontsize=10,color="darkred")
```

Out[187]: Text(0.5, 0, 'No.Iterations')



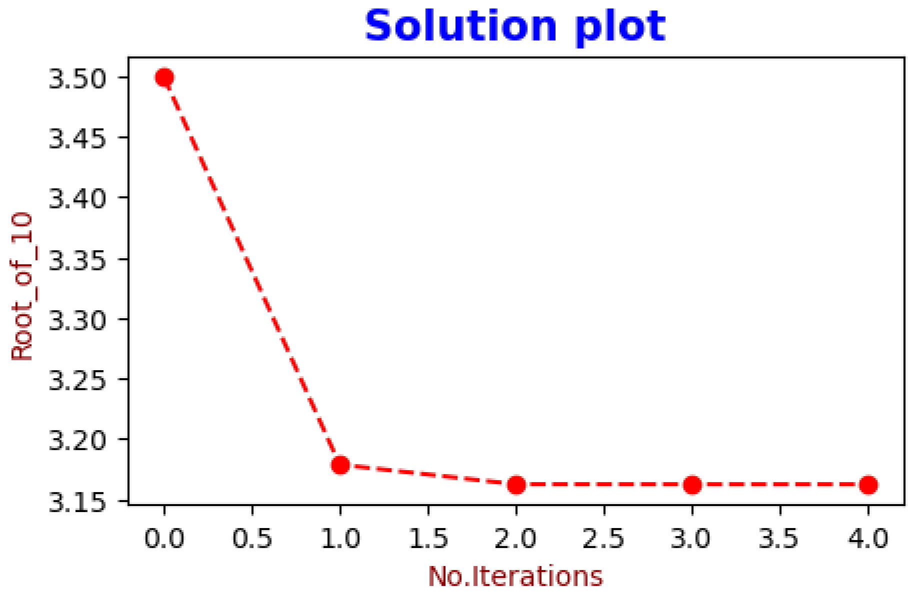


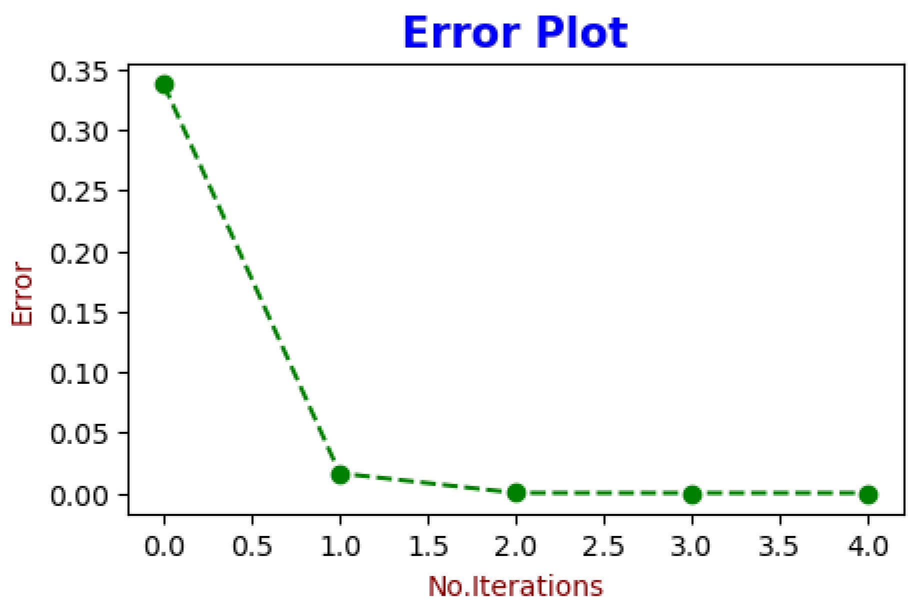
Max_Error condition:

In [186…

```
plt.figure(figsize=(5,7))

# Solution-plot
fig1 = plt.subplot(2,1,1)
plt.plot(np.arange(0,len(ss2)-1,1),ss2[1:], "r--o")
plt.title("Solution plot", fontweight ="bold", fontsize=15,color="b")
plt.ylabel("Root_of_10", fontsize=10,color="darkred")
plt.xlabel("No.Iterations", fontsize=10,color="darkred")
plt.subplots_adjust(hspace=0.4)

# Error-plot
plt.subplot(2,1,2)
plt.plot(np.arange(0,len(error2)-1,1),error2[1:],"g--o")
plt.title("Error Plot", fontweight ="bold", fontsize=15,color="b")
plt.ylabel("Error", fontsize=10,color="darkred")
plt.xlabel("No.Iterations", fontsize=10,color="darkred")
```

```
Out[186]: Text(0.5, 0, 'No.Iterations')
```





Observation:

- Solution is converging like a **second-order** polynomial function.
- **Error_Ratio**: $|e_(n+1)|/|e_(n)^\alpha| = \lambda \neq 0$
    - α : **order of convergence** = 2
    - λ : **assymptotic error constant**
- Find **order of convergence** by first finding different **error ratios**!

```
In [ ]:
```