In [18]:

```python
%run my_functions_library.ipynb                                    #running my library here
import matplotlib.pyplot as plt                                    #importing essential functions
import numpy as np
```

Question 1

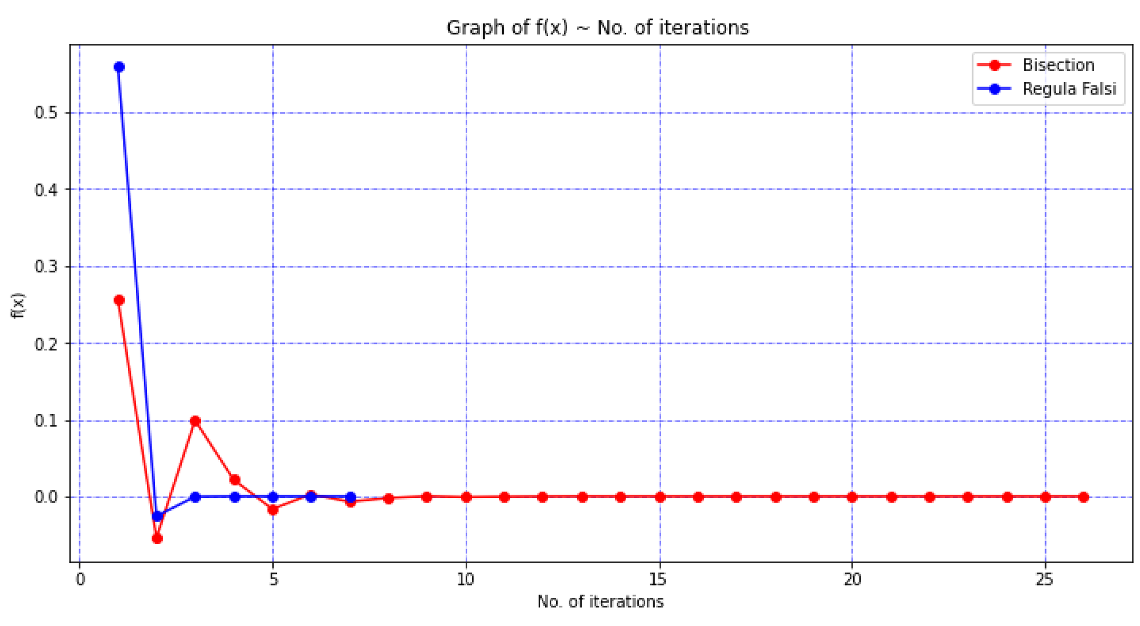
In [19]:

```python
def func_1(x):                                                         #defining the function to return the result of
    return math.log(x/2)-math.sin(5*x/2)                               #mathematical equation

eps=10**-8                                                             #defining till the power of -8 for accuracy

p=1.6
q=2.4
a,b=bracketing(p,q,func_1)                                             #calling bracketing function

print("\n BISECTION METHOD \n")                                        #starting Bisection method
root=bisection(a,b,func_1)
if p==a and q==b:
    print("The root of the given function in the interval (" + str(p) + "," + str(q) + ") is "+str(root))
else:
    print("The root does not fall in the given range (" + str(p) + "," + str(q)+")")
    print("Changing the interval to (" + str(a) + "," + str(b)+")")
    print("The root of the given function in the interval (" + str(a) + "," + str(b) + ") is "+str(root))

print("\n REGULA FALSI METHOD \n")                                     #starting Regula Falsi method
root=regula_falsi(a,b,func_1)
if p==a and q==b:
    print("The root of the given function in the interval (" + str(p) + "," + str(q) + ") is "+str(root))
else:
    print("The root does not fall in the given range (" + str(p) + "," + str(q)+")")
    print("Changing the interval to (" + str(a) + "," + str(b)+")")
    print("The root of the given function in the interval (" + str(a) + "," + str(b) + ") is "+str(root))
```

```
 BISECTION METHOD 

The root does not fall in the given range (1.6,2.4)
Changing the interval to (1.6,2.8)
The root of the given function in the interval (1.6,2.8) is 2.6231403321027753

 REGULA FALSI METHOD 

The root does not fall in the given range (1.6,2.4)
Changing the interval to (1.6,2.8)
The root of the given function in the interval (1.6,2.8) is 2.6231403354374474
```

In [20]:

```python
plt.figure(figsize=(12,6))

p=1.6
q=2.4

a,b=bracketing(p,q,func_1)
x_bis, y_bis, z_bis = bisection_for_plotting(a,b,func_1)
x_rf, y_rf, z_rf = regula_falsi_for_plotting(a,b,func_1)

plt.plot(x_bis, y_bis, 'r-o', label='Bisection')
plt.plot(x_rf, y_rf, 'b-o', label='Regula Falsi')

plt.grid(color='b', ls = '-.', lw = 0.5)
plt.xlabel('No. of iterations')
plt.ylabel('f(x)')
plt.title('Graph of f(x) ~ No. of iterations ')
plt.legend()
plt.show()

                                                        #printing the data for different methods below

print("\nBISECTION METHOD")
print ("{:<20} {:<20} {:<20}".format('No. of iterations', 'f(x)', 'Root convergence'))
for i in range(len(x_bis)):
    print ("{:<20} {:<20} {:<20}".format(Round(x_bis[i],7), Round(y_bis[i],7), Round(z_bis[i],7)))

print("\n\nREGULA FALSI METHOD")
print ("{:<20} {:<20} {:<20}".format('No. of iterations', 'f(x)', 'Root convergence'))
for i in range(len(x_rf)):
    print ("{:<20} {:<20} {:<20}".format(Round(x_rf[i],7), Round(y_rf[i],7), Round(z_rf[i],7)))
```




```
BISECTION METHOD
No. of iterations    f(x)                 Root convergence
1.0                  0.2563228            2.5
2.0                  -0.0537849           2.65
3.0                  0.0989994            2.575
4.0                  0.0216321            2.6125
5.0                  -0.0163703           2.63125
6.0                  0.0025636            2.621875
7.0                  -0.0069209           2.6265625
8.0                  -0.002183            2.6242188
9.0                  0.0001893            2.6230469
10.0                 -0.0009971           2.6236328
11.0                 -0.000404            2.6233398
12.0                 -0.0001074           2.6231934
13.0                 4.09e-05             2.6231201
14.0                 -3.32e-05            2.6231567
15.0                 3.9e-06              2.6231384
16.0                 -1.47e-05            2.6231476
17.0                 -5.4e-06             2.623143
18.0                 -8e-07               2.6231407
19.0                 1.5e-06              2.6231396
20.0                 4e-07                2.6231401
21.0                 -2e-07               2.6231404
22.0                 1e-07                2.6231403
23.0                 -0.0                 2.6231404
24.0                 0.0                  2.6231403
25.0                 -0.0                 2.6231403
26.0                 0.0                  2.6231403


REGULA FALSI METHOD
No. of iterations    f(x)                 Root convergence
1.0                  0.5587359            2.3497199
2.0                  -0.0256251           2.6358588
3.0                  -0.0003458           2.6233111
4.0                  -3.1e-06             2.6231419
5.0                  -0.0                 2.6231403
6.0                  -0.0                 2.6231403
7.0                  -0.0                 2.6231403
```

Question 2

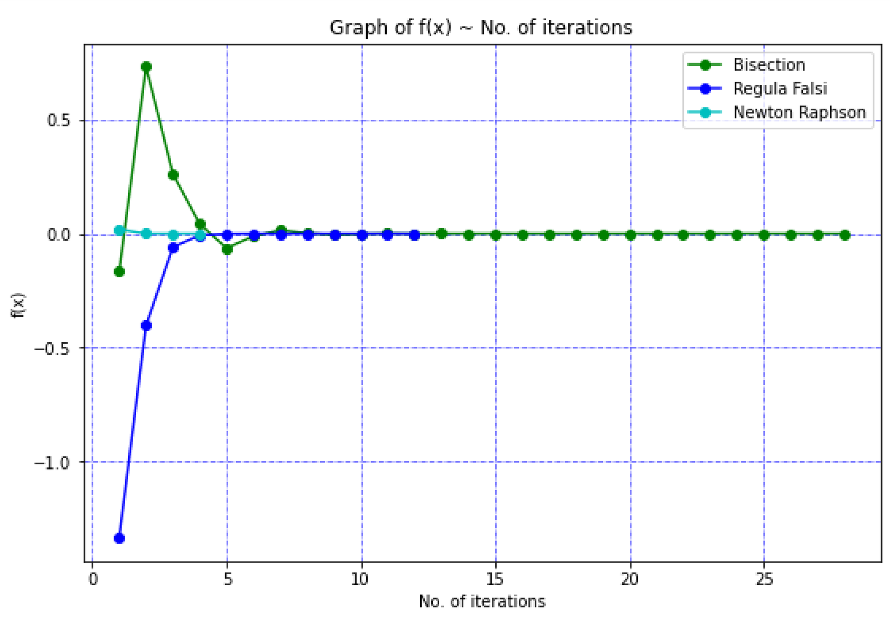
In [21]:

```python
def func_2(x):                                                   #defining a function to return the result of the
    return -1*math.cos(x)-x                                      #mathematical expression

eps=10**-8                                                       #defining till the power of -8 for accuracy

p=1.6
q=2.4
a,b=bracketing(p,q,func_2)                                       #calling bracketing function

print("\n BISECTION METHOD \n")                                  #starting Bisection method
root=bisection(a,b,func_2)
if p==a and q==b:
    print("The root of the given function in the interval (" + str(p) + "," + str(q) + ") is "+str(root))
else:
    print("The root does not fall in the given range (" + str(p) + "," + str(q)+")")
    print("Changing the interval to (" + str(a) + "," + str(b)+")")
    print("The root of the given function in the interval (" + str(a) + "," + str(b) + ") is "+str(root))

print("\n REGULA FALSI METHOD \n")                               #starting Regula Falsi method
root=regula_falsi(a,b,func_2)
if p==a and q==b:
    print("The root of the given function in the interval (" + str(p) + "," + str(q) + ") is "+str(root))
else:
    print("The root does not fall in the given range (" + str(p) + "," + str(q)+")")
    print("Changing the interval to (" + str(a) + "," + str(b)+")")
    print("The root of the given function in the interval (" + str(a) + "," + str(b) + ") is "+str(root))

print("\n NEWTON RAPHSON METHOD \n")                            #starting Newton Raphson method
x=0
root=newton_raphson(x,func_2)
print("The nearest root of the given function for the given value of x = " + str(x) + " is = "+str(root))
```

```
 BISECTION METHOD 

The root does not fall in the given range (1.6,2.4)
Changing the interval to (-1.649999999999999,2.4)
The root of the given function in the interval (-1.649999999999999,2.4) is -0.7390851384960108

 REGULA FALSI METHOD 

The root does not fall in the given range (1.6,2.4)
Changing the interval to (-1.649999999999999,2.4)
The root of the given function in the interval (-1.649999999999999,2.4) is -0.7390851330483599

 NEWTON RAPHSON METHOD 

The nearest root of the given function for the given value of x = 0 is = -0.7390851332151607
```

In [22]:

```python
plt.figure(figsize=(9,6))

p=1.6
q=2.4
guess=0
a,b=bracketing(p,q,f2)

x_bis, y_bis, z_bis = bisection_for_plotting(a,b,f2)
x_rf, y_rf, z_rf = regula_falsi_for_plotting(a,b,f2)
x_nr, y_nr, z_nr = newton_raphson_for_plotting(guess,f2)

plt.plot(x_bis, y_bis, 'g-o', label='Bisection')
plt.plot(x_rf, y_rf, 'b-o', label='Regula Falsi')
plt.plot(x_nr, y_nr, 'c-o', label='Newton Raphson')

plt.grid(color='b', ls = '-.', lw = 0.5)
plt.xlabel('No. of iterations')
plt.ylabel('f(x)')
plt.title('Graph of f(x) ~ No. of iterations')
plt.legend()
plt.show()

                                                        #printing the data for different methods below

print("\nBISECTION METHOD")
print ("{:<20} {:<20} {:<20}".format('No. of iterations', 'f(x)', 'Root convergence'))
for i in range(len(x_bis)):
    print ("{:<20} {:<20} {:<20}".format(Round(x_bis[i],7), Round(y_bis[i],7), Round(z_bis[i],7)))

print("\n\nREGULA FALSI METHOD")
print ("{:<20} {:<20} {:<20}".format('No. of iterations', 'f(x)', 'Root convergence'))
for i in range(len(x_rf)):
    print ("{:<20} {:<20} {:<20}".format(Round(x_rf[i],7), Round(y_rf[i],7), Round(z_rf[i],7)))

print("\n\nNEWTON RAPHSON METHOD")
print ("{:<20} {:<20} {:<20}".format('No. of iterations', 'f(x)', 'Root convergence'))
for i in range(len(x_nr)):
    print ("{:<20} {:<20} {:<20}".format(Round(x_nr[i],7), Round(y_nr[i],7), Round(z_nr[i],7)))
```




```
BISECTION METHOD
No. of iterations    f(x)                 Root convergence
1.0                  -0.1660862           -0.6375
2.0                  0.7295658            -1.14375
3.0                  0.2616988            -0.890625
4.0                  0.0420312            -0.7640625
5.0                  -0.0635574           -0.7007812
6.0                  -0.0111353           -0.7324219
7.0                  0.0153563            -0.7482422
8.0                  0.0020874            -0.740332
9.0                  -0.0045297           -0.736377
10.0                 -0.0012226           -0.7383545
11.0                 0.000432             -0.7393433
12.0                 -0.0003954           -0.7388489
13.0                 1.83e-05             -0.7390961
14.0                 -0.0001885           -0.7389725
15.0                 -8.51e-05            -0.7390343
16.0                 -3.34e-05            -0.7390652
17.0                 -7.6e-06             -0.7390806
18.0                 5.4e-06              -0.7390883
19.0                 -1.1e-06             -0.7390845
20.0                 2.1e-06              -0.7390864
21.0                 5e-07                -0.7390854
22.0                 -3e-07               -0.739085
23.0                 1e-07                -0.7390852
24.0                 -1e-07               -0.7390851
25.0                 0.0                  -0.7390851
26.0                 -0.0                 -0.7390851
27.0                 -0.0                 -0.7390851
28.0                 0.0                  -0.7390851


REGULA FALSI METHOD
No. of iterations    f(x)                 Root convergence
1.0                  -1.3299444           0.4147119
2.0                  -0.4027049           -0.4829322
3.0                  -0.059259            -0.7033929
4.0                  -0.0072328           -0.7347593
5.0                  -0.0008591           -0.7385718
6.0                  -0.0001017           -0.7390244
7.0                  -1.2e-05             -0.7390779
8.0                  -1.4e-06             -0.7390843
9.0                  -2e-07               -0.739085
10.0                 -0.0                 -0.7390851
11.0                 -0.0                 -0.7390851
12.0                 -0.0                 -0.7390851


NEWTON RAPHSON METHOD
No. of iterations    f(x)                 Root convergence
1.0                  0.0189231            -0.7503639
2.0                  4.65e-05             -0.7391129
3.0                  0.0                  -0.7390851
4.0                  0                    -0.7390851
```

Question 3

In [23]:

```python
coefficient=[1,0,-5,0,4]

n=len(coefficient)
guess = 1.5

print("Solutions of the polynomial equation are:")
for i in range(n-1):
    coeff, root = laguerre(poly_function, first_deriv_poly, second_deriv_poly, coefficient, guess)
    print(Round(root,5))
```

```
Solutions of the polynomial equation are:
1.0
1.0
1.0
1.0
```