In [ ]:

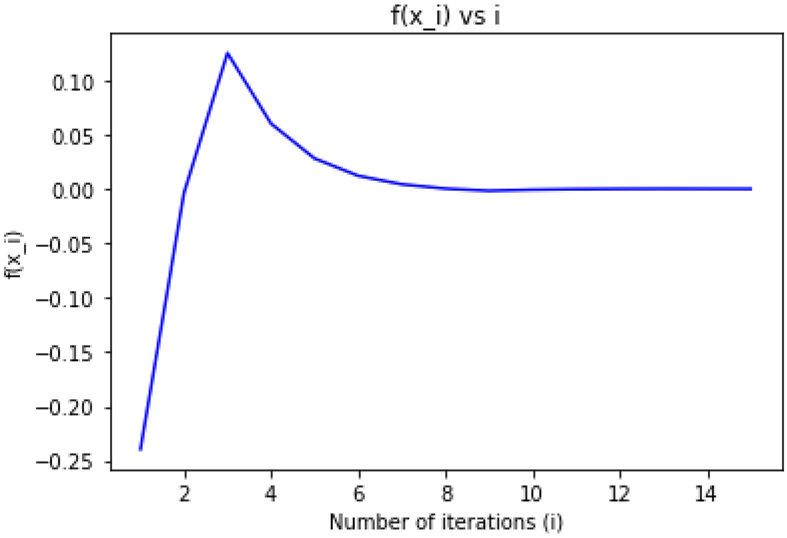
```python
import math
import matplotlib.pyplot as graph

def f1(x):  # Question 1
    return math.log(x/2) - math.sin(5*x/2)

def f2(x):   # Question 2
    return -(x + math.cos(x))
```

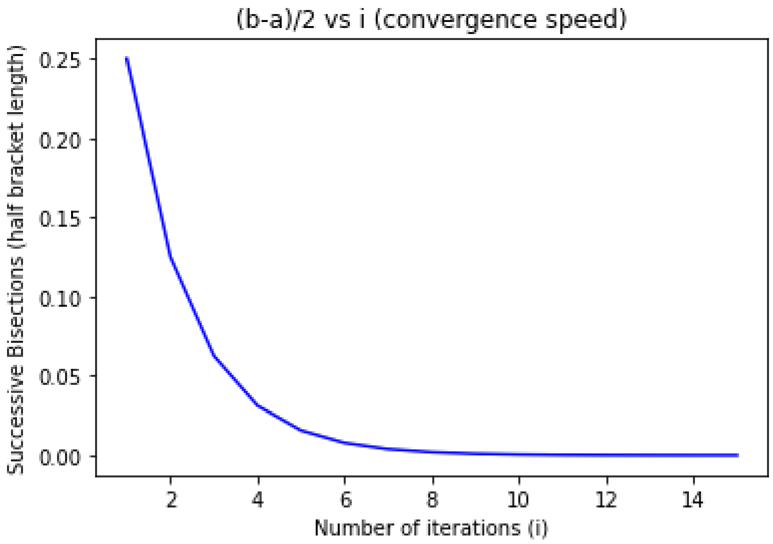
In [ ]:

```python
def Bisection(a,b,e,f):

    itr = []; fx = []; err = []  # storing the no. of iterations, corresponding valu

    if f(a)*f(b) < 0:     # necessary condition of bracketing
        c=(a+b)/2             # Bisection of interval

        switch = True
        i=0

        print("No. of iterations (i)        "+"Successive Bisections (half bracket l

        while switch == True and i<15:

            print("        ", i,"                   ",abs(b-a)/2,"\n")

            if f(a)*f(c) < 0:     # Shortening the bracket
                b = c             # The condtions help determine
            elif f(c)*f(b) < 0:   # if c needs to be shifted towards
                a = c             # a or b, i.e. closer to the root

            c = (a+b)/2

            if abs(b-a)/2 < e:   # tolerance check
                switch = False
            i = i+1

            itr.append(i); fx.append(f(c)); err.append(abs(b-a)/2)

        print (str(c) + " is the root.")
        graph.plot(itr,fx,'-b', label = 'Root finding steps of Bisection Method.')
        graph.title("f(x_i) vs i")
        graph.xlabel("Number of iterations (i)")
        graph.ylabel("f(x_i)")
        graph.show()
        graph.plot(itr,err,'-b', label = 'Steps to convergence in Bisection Method')
        graph.title("(b-a)/2 vs i (convergence speed)")
        graph.xlabel("Number of iterations (i)")
        graph.ylabel("Successive Bisections (half bracket length)")
        graph.show()





    elif f(a)*f(b) == 0:
        if f(a) == 0:
            print(str(a)+" is the root.")
        elif f(b) == 0:
            print(str(b)+" is the root.")

    elif f(a)*f(b) > 0:
        print("Choose interval carefully.")
```

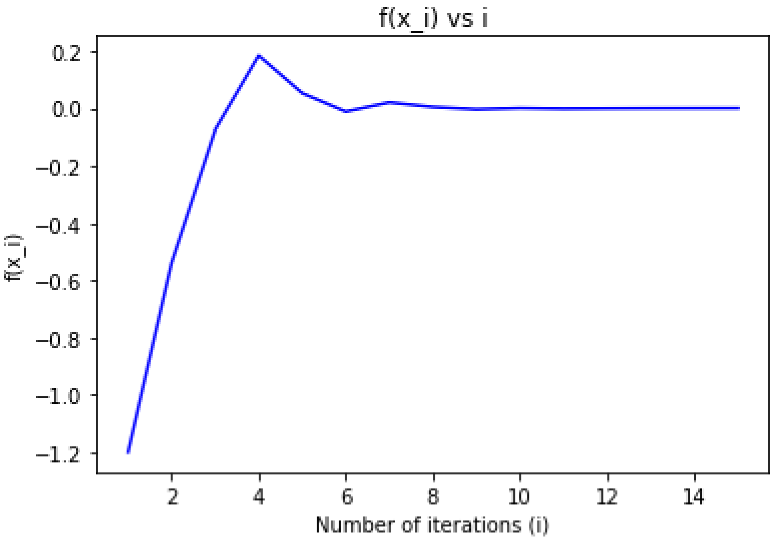
```
Bisection(2,3,0.000000000001,f1)
Bisection(-3.45,1.45,0.0000001,f2)
```

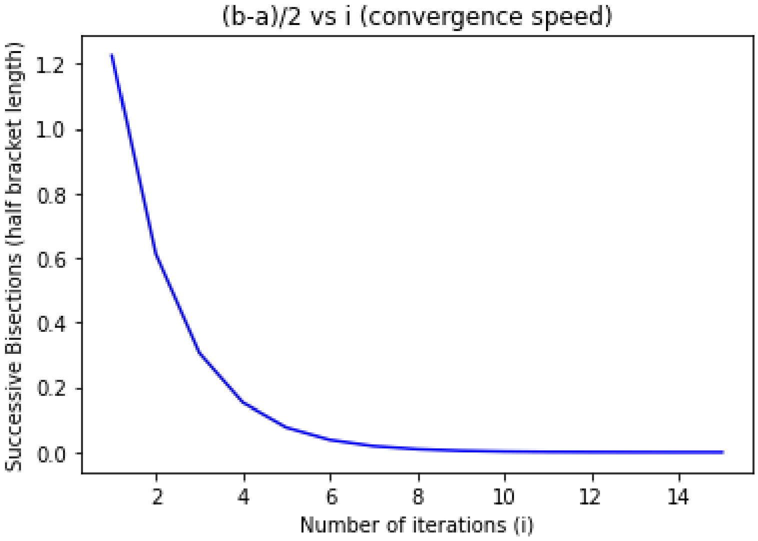
```
No. of iterations (i)        Successive Bisections (half bracket length)
         0                     0.5

         1                     0.25

         2                     0.125

         3                     0.0625

         4                     0.03125

         5                     0.015625

         6                     0.0078125

         7                     0.00390625

         8                     0.001953125

         9                     0.0009765625

         10                     0.00048828125

         11                     0.000244140625

         12                     0.0001220703125

         13                     6.103515625e-05

         14                     3.0517578125e-05

2.6231536865234375 is the root.
```





```
No. of iterations (i)        Successive Bisections (half bracket length)
         0                     2.45

         1                     1.225

         2                     0.6125

         3                     0.30625

         4                     0.153125

         5                     0.07656250000000003

         6                     0.03828124999999999

         7                     0.019140625000000022

         8                     0.009570312500000011

         9                     0.004785156249999978

         10                     0.002392578124999989

         11                     0.0011962890624999667

         12                     0.0005981445312499556

         13                     0.0002990722656249778

         14                     0.0001495361328124889

-0.7391342163085936 is the root.
```

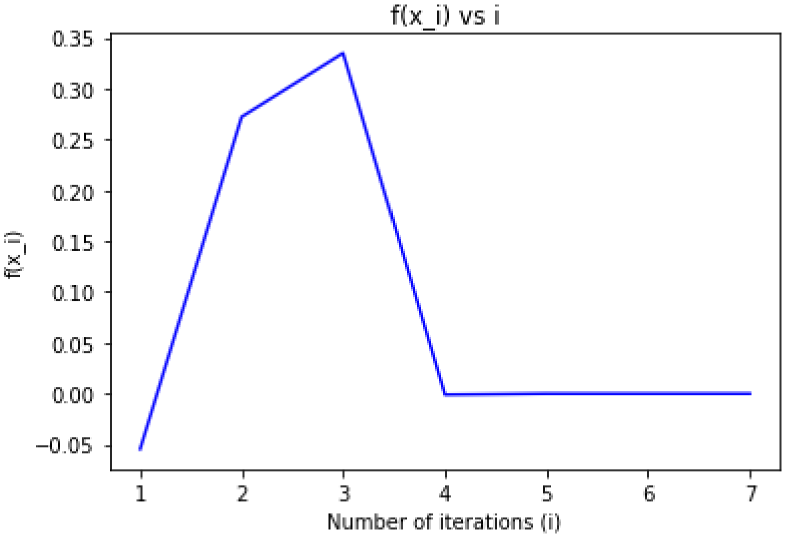


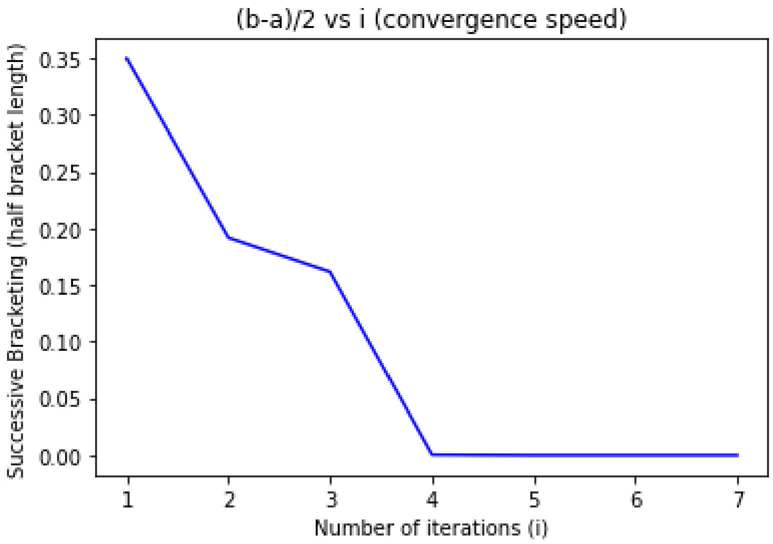


In [ ]:

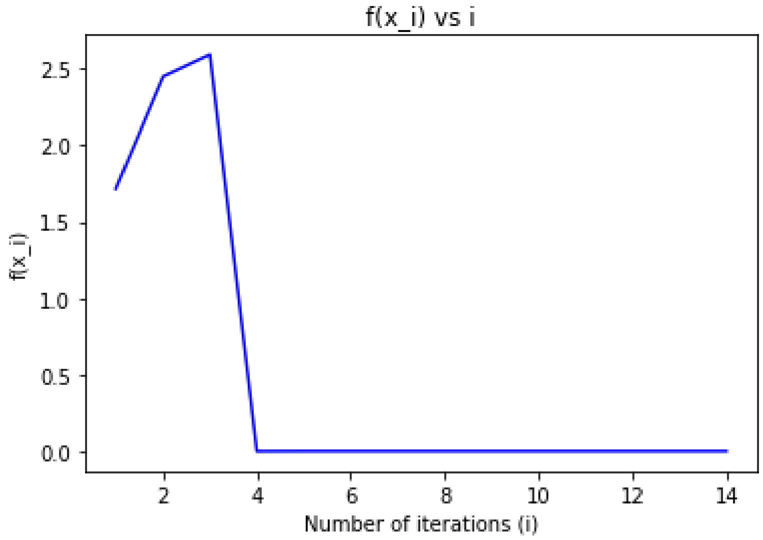
```
def RegulaFalsi(a,b,e,f):

    itr =[]; fx = []; err =[] # storing the no. of iterations, corresponding value o


    if f(a)*f(b) < 0: # necessary condition of bracketing

        print("No. of iterations (i)        "+"Successive Bracketing (half bracket l

        i=0
        while abs(b-a)/2 > e and i<14:

            c = b - (((b-a)*f(b))/(f(b)-f(a)))  # Position through slope of the Line

            print("        ", i,"                       ",abs(b-a)/2,"\n")

            if f(a)*f(c) < 0:  # again, the same process as in Bisection
                b = c
            elif f(c)*f(b) < 0:
                a = c

            c = (a+b)/2
            i = i+1
            itr.append(i); fx.append(f(c)); err.append(abs(b-a)/2)


        print (str(c) + " is the root.")
        graph.plot(itr,fx,'-b', label = 'Root finding steps of Regula Falsi Method.'
        graph.title("f(x_i) vs i")
        graph.xlabel("Number of iterations (i)")
        graph.ylabel("f(x_i)")
        graph.show()
        graph.plot(itr,err,'-b', label = 'Steps to convergence in Regula Falsi Metho
        graph.title("(b-a)/2 vs i (convergence speed)")
        graph.xlabel("Number of iterations (i)")
        graph.ylabel("Successive Bracketing (half bracket length)")
        graph.show()


    elif f(a)*f(b) == 0:
        if f(a) == 0:
            print(str(a)+" is the root.")
        elif f(b) == 0:
            print(str(b)+" is the root.")

    elif f(a)*f(b) > 0:
        print("Choose interval carefully.")

RegulaFalsi(1.6,3,0.000001,f1)
RegulaFalsi(-3.45,1.45,0.0000001,f2)
```

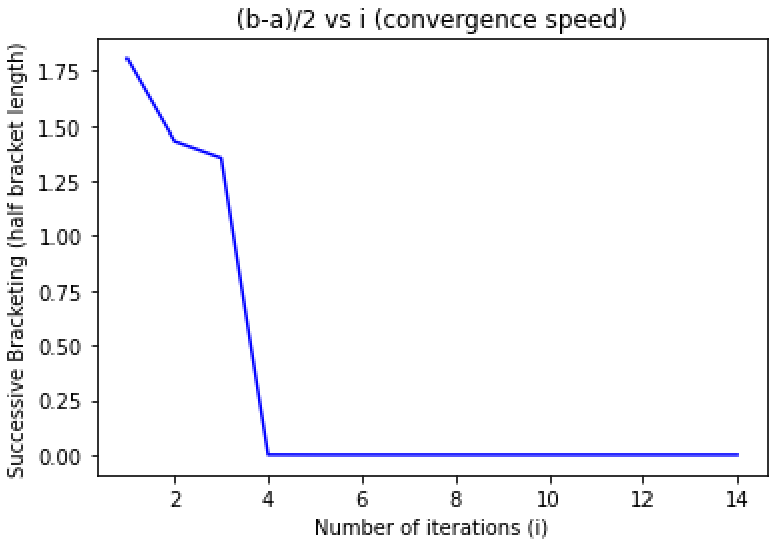
```
No. of iterations (i)        Successive Bracketing (half bracket length)
         0                         0.7

         1                         0.3496309992049955

         2                         0.19174954190501925

         3                         0.16168514194260086

         4                         0.0004883555873411716

         5                         4.382196104879554e-06

         6                         4.380567895312382e-06

2.6231403354363136 is the root.
```





```
No. of iterations (i)        Successive Bracketing (half bracket length)
         0                         2.45

         1                         1.805847141769012

         2                         1.4322006478870304

         3                         1.3558934238142706

         4                         0.00044918886783479933

         5                         1.3200988314754358e-05

         6                         1.319844599872999e-05

         7                         1.3198445983964024e-05

         8                         1.3198445983964024e-05

         9                         1.3198445983964024e-05

         10                         1.3198445983964024e-05

         11                         1.3198445983964024e-05

         12                         1.3198445983964024e-05

         13                         1.3198445983964024e-05

-0.7390983316611446 is the root.
```





In [ ]:

```
def Diff(x,f):        # first derivative of a function
    h = 1/1000
    y = 0.5*(f(x+h) - f(x-h))/h
    return y



def NewtonRaphson(x_o,e,f):

    i = 0 # iterations
    x = 1 # reference marker

    while abs(x-x_o) > e and i < 20:
        x = x_o
        x_o = x_o - (f(x_o)/Diff(x_o,f))  # Formula of the Mean value Theorem

        if f(x_o) == 0:
            print(str(x_o) + " is the root.")
        i += 1
    print("The root is: " + str(x_o))

NewtonRaphson(0,0.00001,f2)
```

```
The root is: -0.7390851332151606
```