# Lecture_9

March 9, 2023

# 1 Roots of Algebraic and Transcendental Functions

## 1.1 1. Bisection Method

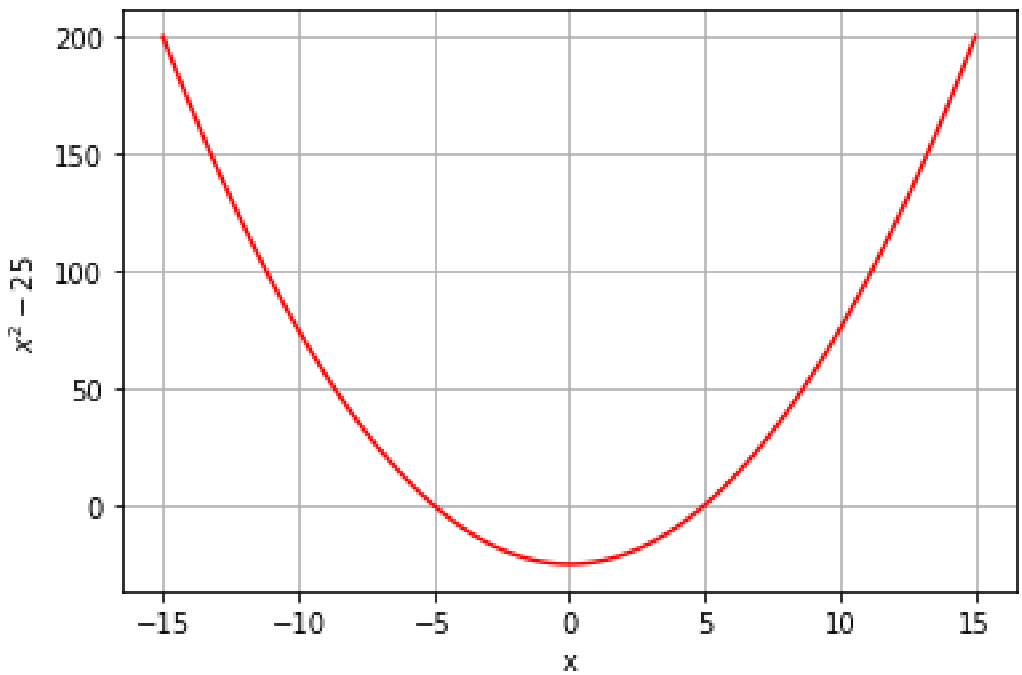
[1]:
```
tol = 10e-4

def bisect(f, a, b):

    fa = f(a)
    if fa == 0: return a

    fb = f(b)
    if fb == 0: return b

    if fa * fb > 0.0 : raise ValueError("Root is not bracketed")


    while abs(b-a) >= tol:

        r = 0.5 * (a+b)

        fr = f(r)

        if fr == 0 : return r

        if fa * fr < 0.0:
            b = r
            fb = fr

        else:
            a = r
            fa = fr

    return 0.5*(a+b)
```

[2]:
```
def f(x):
    return x**2 - 25
```

```python
a, b = eval(input("lower limit, upper limit \n"))

# the eval function evaluates the "String" like a python expression and returns␣
↪the result as an integer

print("Root is",bisect(f,a,b))
```

```
lower limit, upper limit
0,9
Root is 5.000152587890625
```

[3]:
```python
import numpy as np
import matplotlib.pyplot as plt

x = np.linspace(-15,15,100)
plt.plot(x,f(x), color = 'r')
plt.ylabel(r"$x^2 - 25$")
plt.xlabel("x")
plt.grid()
```



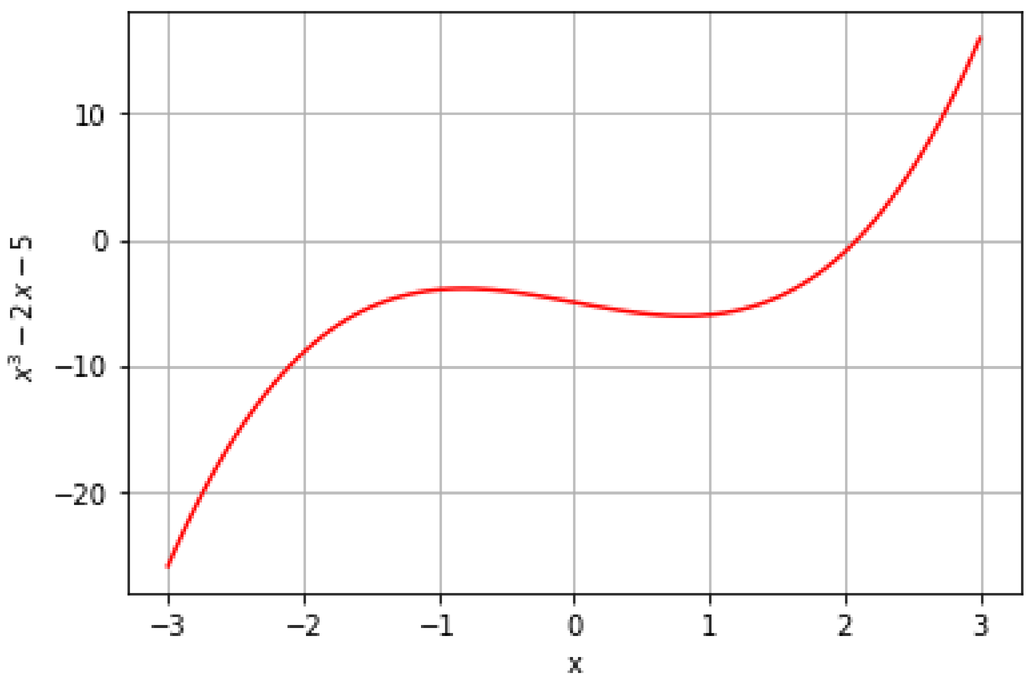
### 1.2 Secant Method

[36]:
```
tol = 10e-4

def secant(f, x0, x1):

    f0 = f(x0)
    if f0 == 0: return x0

    f1 = f(x1)
    if f1 == 0: return x1

    while abs(f(x1)) >= tol:

        x2 = x1 - f(x1)*(x1 - x0)/ (f(x1) - f(x0))
        x0, x1 = x1, x2

    return x1
```

[37]:
```
def f(x):
    return x**3 - 2*x - 5

x0, x1 = eval(input("x0, x1 \t"))

print("Root is", secant(f,x0,x1))
```

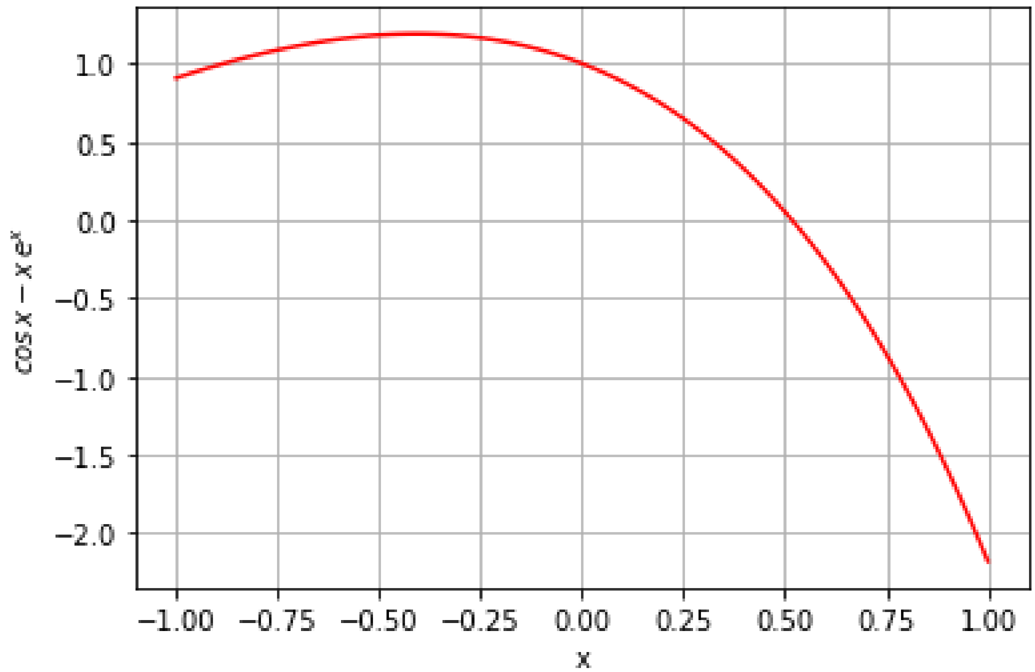
```
x0, x1  0,9
Root is 2.0944755847107026
```

[54]:
```
import numpy as np
import matplotlib.pyplot as plt

x = np.linspace(-3,3,100)
plt.plot(x,f(x), color = 'r')
plt.ylabel(r"$ x^3 - 2 \, x - 5$")
plt.xlabel("x")
plt.grid()
```



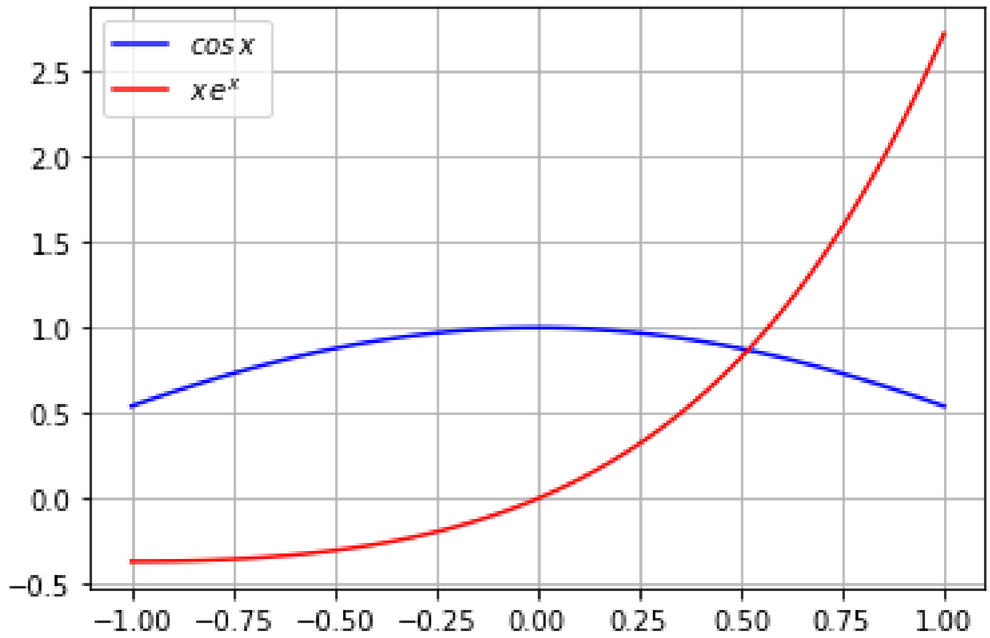

**Q : Modify the above program for regula - falsi method. Then compare the result for the function $ cos , x = x , e^x $.**

[53]:
```
import numpy as np
import matplotlib.pyplot as plt

def g(x):
    return np.cos(x) -  x * np.exp(x)

x = np.linspace(-1,1,100)
plt.plot(x, g(x), color = 'r')
plt.ylabel(r"$cos \, x - x \, e^x$")
plt.xlabel("x")
plt.grid()
```




[50]:
```
import numpy as np
import matplotlib.pyplot as plt

def g(x):
    return np.cos(x)

def h(x):
    return x * np.exp(x)

x = np.linspace(-1,1,100)
plt.plot(x, g(x), color = 'b', label = r"$cos \, x$")
plt.plot(x, h(x), color = 'r', label = r"$x \, e^x$")
plt.grid()
plt.legend()
```

[50]:
```
<matplotlib.legend.Legend at 0x17daa4e8b50>
```




### 1.3 Newton Raphson Method

[5]:
```
def f(x):
    return x**2 - 25.0

def fd(x):
    return 2.0*x

n = 1000
x1 = 7
tol = 10e-5

while abs(f(x1)) >= tol:
    fx = f(x1)
    fdx = fd(x1)
    h = - fx/fdx
    x1 = x1 + h

print("The root is", x1)
```

```
The root is 5.000005953745352
```

Now suppose you don't know derivative of the given function. Then you need numerical differentiation methods to use Newton Raphson method.