Computer Methods in Engineering
Exercise 5

Problem 1

Bracket search methods are described in lecture notes.

a) Implement the bracket search method to locate a minimum of the function $f(x) = x^2 - sin(x)$ on the interval $(a, b) = (0, 1)$. Split the interval (a, b) into four subintervals of length $\Delta x = (b - a)/4.0$ and initially select α and β by $\alpha = a + \Delta x, \beta = b - \Delta x$. Experiment with different values of Δx.

Solution:

Below is a listing of a simple implementation of the bracketing method. The number of iterations required was 10000 before an acceptable solution was found. See also 1.

```
'''Solution problem 1a exercise 5
'''

import numpy as np
import pylab as pl
from math import *

#Function to minimize
def f(x) :
  rval = x*x-np.sin(x)
  return rval

n = 20000;                          # max number of iterations
eps = 1e-12;                      # tolerance

a = 0.0
b = 1.0
nx = 10000
dx = (b-a)/4.0

#Loop over all iterations
for i in range(0,n):

  #Create alpha and beta values
  alpha = a+dx
  beta  = b-dx

  #Update the next interval
  if f(alpha) < f(beta) :
    b = beta
  else :
    a = alpha
```

```
  #Check for convergence
  err = abs(b-a)
  #Update dx
  dx=err/4.0
  if err < eps :
    x0 = (b + a)/2
    print("Estimated value for x0: ", x0)
    print("No of iterations: ", i)
    print("Error: ", err)
    break
  else :
    if(i == n-1):
      print("No convergence!")

xx = np.arange(0,1.0,0.001)
ff = f(xx)
pl.plot(xx,ff)
pl.plot(x0,f(x0),'ro',label='Solution point')
pl.legend()
pl.xlabel('x')
pl.ylabel('f(x)')
pl.savefig('bracket.pdf')
pl.show()
```

b) Implement the golden ration method to locate a minimum of the function $f(x) = x^2 - sin(x)$ on the interval $(a,b) = (0,1)$. Compare the number of iterations used to find a solution with the number of iterations used in the previous point.

Solution:
Below is a listing of an implementation of the Golden ratio method. The number of iterations required to get an acceptable solution was 58. Compared to the bracketing method used in the previous point, the Golden ratio method is faster by a factor of 96/58. See figure 2 for a plot of the solution.

```
'''Solution problem 1b exercise 5
'''

import numpy as np
import matplotlib.pylab as pl
from math import *

def f(x) :
  """
  This function computes f(x)=(x-0.25)**2
  :param x : input float argument
  :return  : (x-0.25)**2
  """

  rval = x*x-np.sin(x)
  return rval

r = (sqrt(5)-1)/2  # inverse golden ratio
n = 100;           # max number of iterations
eps = 1e-12;       # tolerance
```

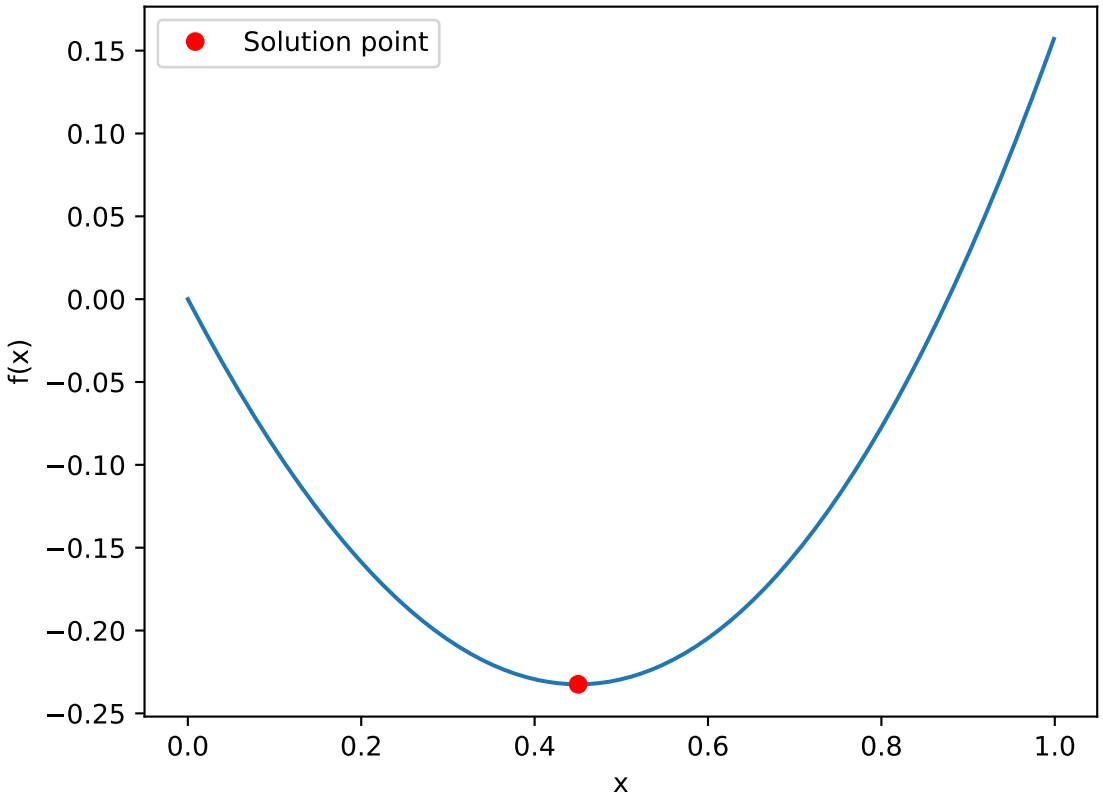

Figure 1: Graph of $f(x) = x^2 - sin(x)$ with minimum point found by simple bracketing.

.

```
a = np.zeros([n])  # start of interval
b = np.zeros([n])  # end of interval
alpha = np.zeros([n])
beta = np.zeros([n])
a[0] = 0;          # lower limit of the interval
b[0] = 1;          # upper limit of the interval
nx=100
x = np.linspace(a[0],b[0],nx);   # intervall x to plot f(x)

for i in range(0,n):
   alpha[i] = a[i]+(1-r)*(b[i]-a[i]);
   beta[i]  = b[i]-(1-r)*(b[i]-a[i]);
   if f(alpha[i]) <= f(beta[i]) :
        a[i+1] = a[i];
        b[i+1] = beta[i];
   else :
        a[i+1] = alpha[i];
        b[i+1] = b[i];

   if abs(b[i]-a[i]) < eps :
        x0 = (a[i]+b[i])/2;
        print('Convergence reached after', i, ' iterations.')
        print('x0 =    ', x0)
        print('f(x0) = ', f(x0))
        break

#Plot f(x) and solution point
dx=0.01
xx = np.arange(0,1.0,dx)
ff = f(xx)
pl.plot(xx,ff)
pl.plot(x0,f(x0),'ro',label='Solution point')
pl.legend()
pl.xlabel('x')
pl.ylabel('f(x)')
pl.savefig('golden.pdf')
pl.show()
```

Problem 2

Consider the one dimensional function

$$f(x) = 2x^3 + 2x^2 - 4x \quad \text{for } x \in [-1, 1]. \tag{1}$$

a) Find the roots of $f(x)$ and check your result using the numpy `roots()` function. *Hint:* The `roots()` function take the polynomial coefficients as a vector input, e.g. for $h(x) = 2x^2 - 1$ you would write: `np.roots([2.0,0.0,-1])`.

Solution:

```
''' Solution problem 2 exercise 5
'''

import numpy as np
import pylab as pl
```

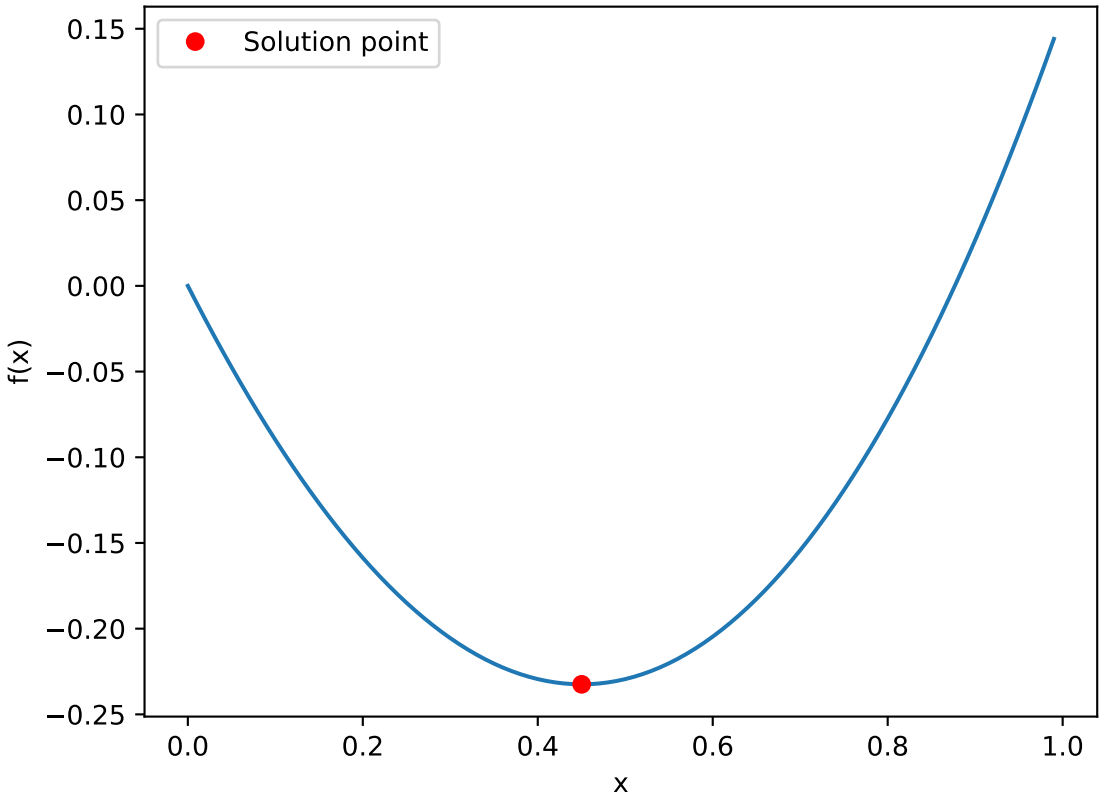

Figure 2: Graph of $f(x) = x^2 - sin(x)$ with minimum point found by the golden ratio method.

.

```
from math import *

#Function to find root
def f(x) :
  rval = 2*x**3+2*x**2-4*x
  return rval

#Find root :
coeff = np.array([2.0,2.0,-4.0,0.0])
r = np.roots(coeff)
print(r)

dx=0.01
xv = np.arange(-3,2.0,dx)
ff = f(xv)
pl.plot(xv,ff)
pl.xlim(-3.0,2.0)
pl.ylim(-3.0,5.0)
pl.plot(r[0],f(r[0]),'ro',label='first root' )
pl.plot(r[1],f(r[1]),'bo',label='second root')
pl.plot(r[2],f(r[2]),'go',label='third root')
pl.plot([-3,2], [0,0])
pl.legend()
pl.xlabel('x')
pl.ylabel('f(x)')
pl.savefig('root.pdf')
pl.show()



def gradf(x):
  return 6*x**2+4*x-4

#Error tolerance
eps = 1e-4

#Maximum number of iterations
maxIt = 100

#Initial point
xx=0.75
itr=0
diff=2*eps

while(itr<(maxIt-1) and diff>eps):
    xxnew=xx-f(xx)/gradf(xx)
    diff=np.abs(xxnew-xx)
    xx=xxnew
    print(itr,xx)
    itr+=1

print('Root: ',xx, '\n Number of iterations: ',itr)

def ddf(x):
  return 12*x+4

#Initial point
xx=-0.25
itr=0
```

```
diff=2*eps

while(itr<(maxIt-1) and diff>eps):
    xxnew=xx-gradf(xx)/ddf(xx)
    diff=np.abs(xxnew-xx)
    xx=xxnew
    print(itr,xx)
    itr+=1

print('Extremum: ',xx, '\n Number of iterations: ',itr)


pl.plot(xv,ff)
pl.xlim(-3.0,2.0)
pl.ylim(-3.0,5.0)
pl.plot(xx,f(xx),'ro',label='extremum' )
pl.plot([-3,2], [0,0])
pl.legend()
pl.xlabel('x')
pl.ylabel('f(x)')
pl.savefig('extremum.pdf')
pl.show()
```

b) Plot $f(x)$ in python using `plot()`. Use an appropriate sampling interval Δx on the x-axis. Give your plot a title and label the axes. Indicate the roots with a red dot and blue dot. Plot also the line $y = 0$

Solution:
See figure 4.

c) Use Netwon's method to find a root of the function f. Suggested starting-points are 0.5 and 0.75. As there are several roots, explain why you get the root you get when using the Netwon method.

Solution:
See the code in the solution of problem 2a.

d) Use Netwon's method to find a maximum or minimum value of the function f. Suggested starting-points are somewhere in the range $[-0.5, 0.0]$. Explain why you get the solution you get when using the Netwon method and your chosen starting-point.

Solution:
See the code in the solution of problem 2a. See figure 4.

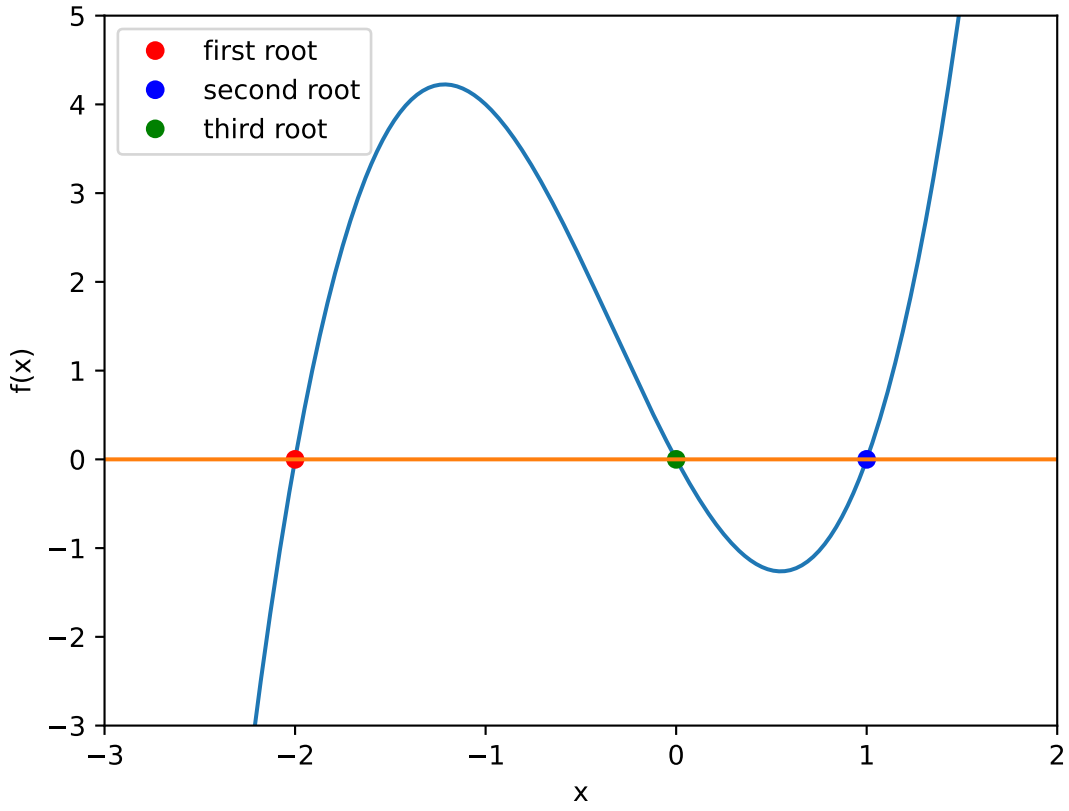

Figure 3: Roots of the function $f(x) = 2x(x - 1.0)(x + 1.0)$

.

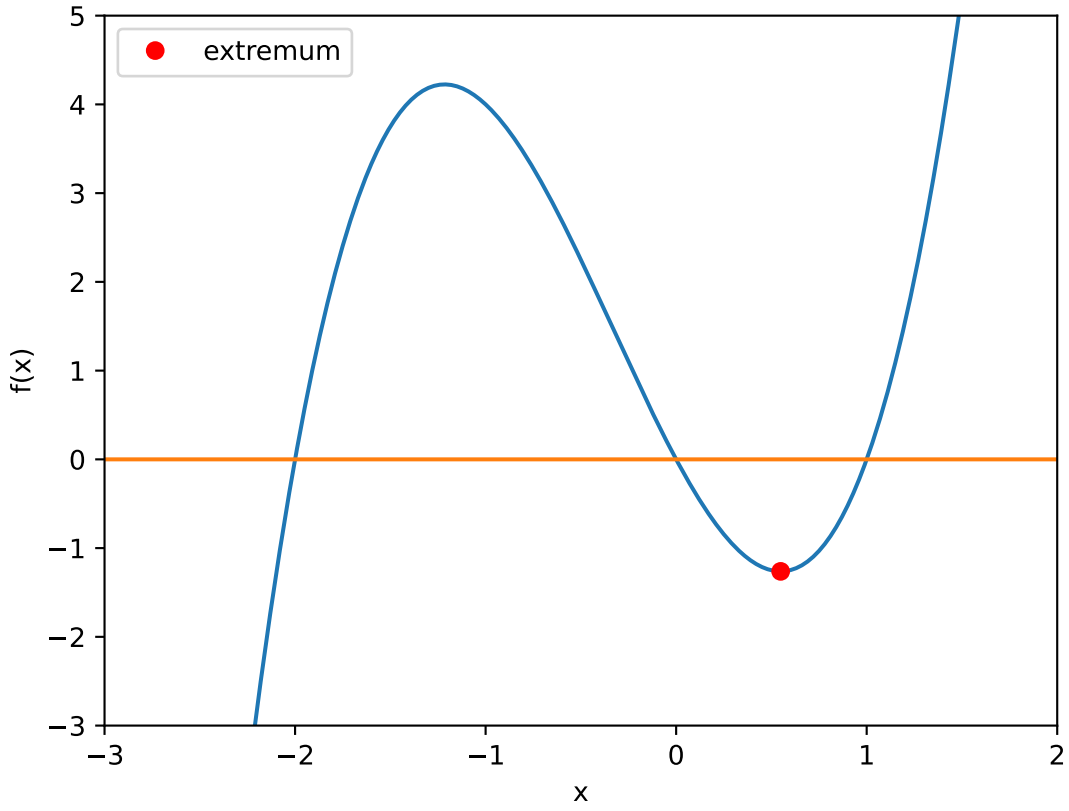

Figure 4: Extremum of the function $f(x)$

.