ENGN2020 – HOMEWORK4

Problem 2

(a) Answer:

Define $f_i = (x - r_i)^{-1}$, where r_i is the ***ith*** root factored out from f.

Then, the general function of the new function g can be write as:

$$g = f * \prod_{i=1}^{n} f_i$$

Note that, for derivative of continuous product of functions:

$$(\prod_{i=1}^{n} f_i)' = \sum_{i=1}^{n} (\prod_{j=1}^{i-1} f_j * f_i{}' * \prod_{k=i+1}^{n} f_k) = \sum_{i=1}^{n} \frac{\prod_{j=1}^{n} f_j}{f_i} * f_i{}'$$

Then g' can be write as:

$$g' = f' * \prod_{i=1}^{n} f_i + f * (\sum_{i=1}^{n} \frac{\prod_{j=1}^{n} f_j}{f_i} * f_i{}')$$

Since $f_i = (x - r_i)^{-1}$, then $f_i{}' = -(x - r_i)^{-2}$, so:

$$\sum_{i=1}^{n} \frac{\prod_{j=1}^{n} f_j}{f_i} * f_i{}' = -\sum_{i=1}^{n} ((\prod_{j=1}^{n} f_j) * f_i)$$

Therefore:

$$g' = f' * \prod_{i=1}^{n} f_i - f * \sum_{i=1}^{n} ((\prod_{j=1}^{n} f_j) * f_i)$$

(b) Answer:

The definition of the "*RootFactor*" class is shown as below:

```
class RootFactor:
    def __init__(self, f, fprime, roots):
        self.f = f
        self.fprime = fprime
        self.roots = roots

    def get_f(self, x):
        #get the length of array nums
        rootNum = len(self.roots)
        #get f(x)
        result = self.f(x)
        #g = f(x) multiplied by the continous product of fi, where fi = 1/(x-root[i])
        for i in range(rootNum):
            result = result*(1./(x-self.roots[i]))
        #return g
        return result

    def get_fprime(self, x):
        #get the length of array nums
        rootNum = len(self.roots)
```

```
# the continous product of fi, where fi = 1/(x-root[i]) is needed several times in the calculation, calculate it first
# continousProduct = 1
for i in range(rootNum):
    continousProduct = continousProduct*(1./(x-self.roots[i]))
#according to the equation in Problem4(a), g` equals two parts summed together
#the first part is f` multiplied by the continous product of fi
part1 =self.fprime(x)*continousProduct
#set the initial value of part2
part2=0
#part2 is the sum of fi*continous product
for i in range(rootNum):
    part2 = part2 -continousProduct*(1.0/(x-self.roots[i]))
#multiplied part2 by f(x)
part2 = part2 * self.f(x)
#g` is the sum of part1 and part2
return part1+part2
```

Problem 3

(a) Answer:

We know that:

$$\begin{cases} f_1(x_1, x_2; \vartheta) = sin3x_1 - x_2 \\ f_2(x_1, x_2; \vartheta) = {x_1}^2 - x_2 - 3x_1 + \vartheta \end{cases}$$

Then, the Jacobian matrix of the system is shown as below:

$$J = \begin{bmatrix} 3cos3x_1 & -1 \\ 2x_1 - 3 & -1 \end{bmatrix}$$

Let $\det(J) = 0$, then $2x_1 - 3 - 3cos3x_1 = 0$,

By using python to solve the equation numerically, three roots can be calculated:

$$\begin{cases} r_1 \approx 0.70880 \\ r_2 \approx 1.59104 \\ r_3 \approx 2.40273 \end{cases}$$

Let $x_1 = r_i, i = 1,2,3$, then θ can be calculated:

$$\begin{cases} \theta_1 \approx 2.4736 \\ \theta_2 \approx 1.2436 \\ \theta_3 \approx 2.2337 \end{cases}$$

(b) Answer:

By using graphical procedure, and try all three possible θ, we can find that when $\theta = 2.4736$, there is only one real root. The figure is shown as below:

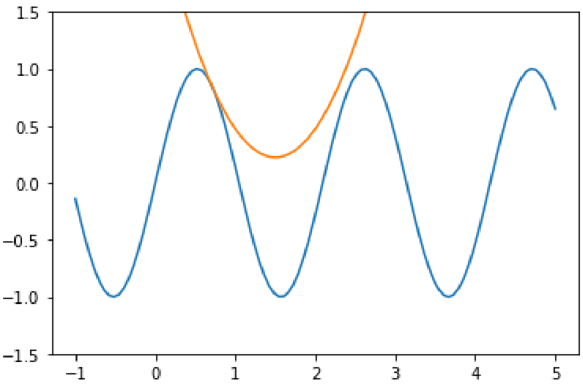

Fig 1. The figure of the system when it has only one real root

(c) Answer:

The maximum number of real roots this system is four. A figure of the system when $\theta = 1.5$ is shown as below:

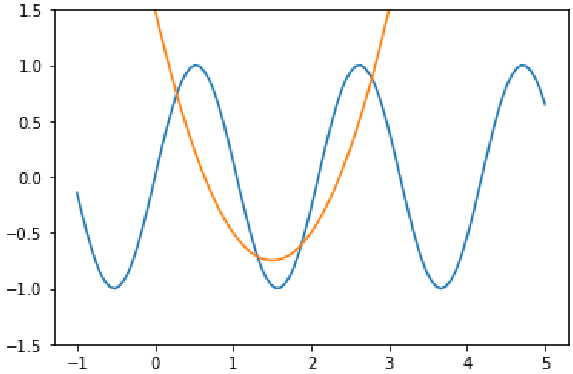

Fig 2. The figure of the system when $\theta = 1.5$ it has four real root

(d) Answer:

When θ is in range of $(1.2436, 2.2337)$, there are four real roots. When $\theta = 1.2436\ or\ \theta = 2.2337$, the system has 3 real roots. The system with $\theta = 1.2436\ and\ \theta = 2.2337$ as bounds is shown as below

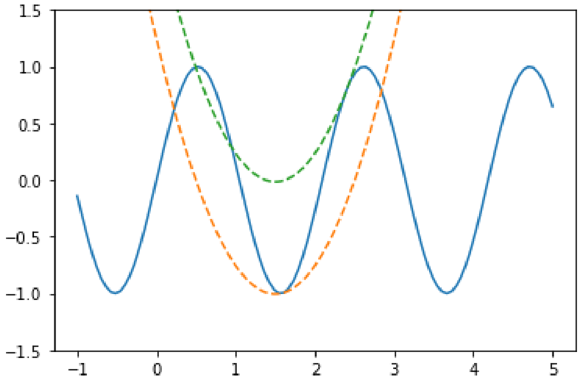

Fig 3. The figure of the system when $\theta = 1.2436\ and\ \theta = 2.233$. When θ is in range of $(1.2436, 2.2337)$, there are four real roots.

Problem 4

(a) Answer:

Below is a contour of given function. The range of x1 and x2 are [-6,6], there are several local minima in the plot, and the one on the top-left corner is the minimum point.

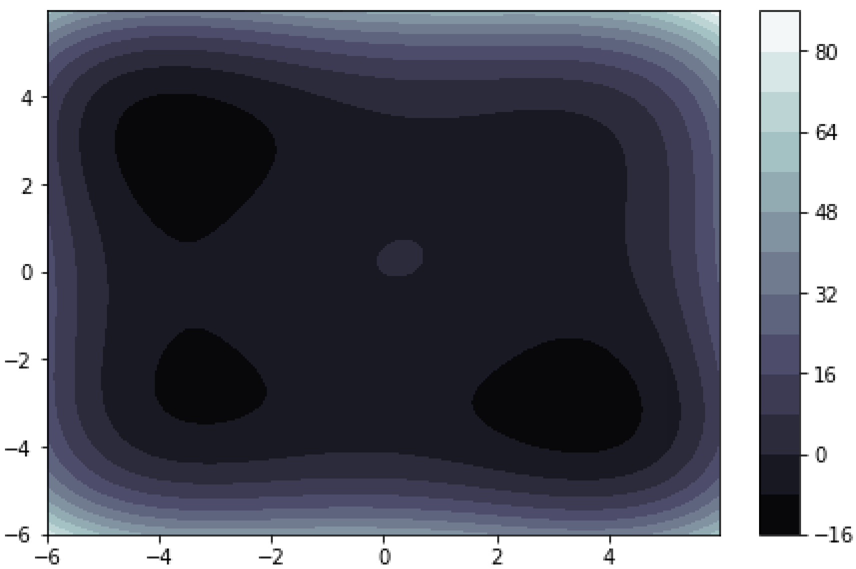

Fig 4. The contour image of the function

(c) Answer:

The initial guesses are *(0,0)*, *(0, 1.02)* and *(-1.04, 0)*. The termination condition is when the standard deviation of the function values of the current three vertices is smaller than 0.1. After several iteration, the loop stops and reaches the minimum point. The trajectories of triangles of each iteration is shown as below:

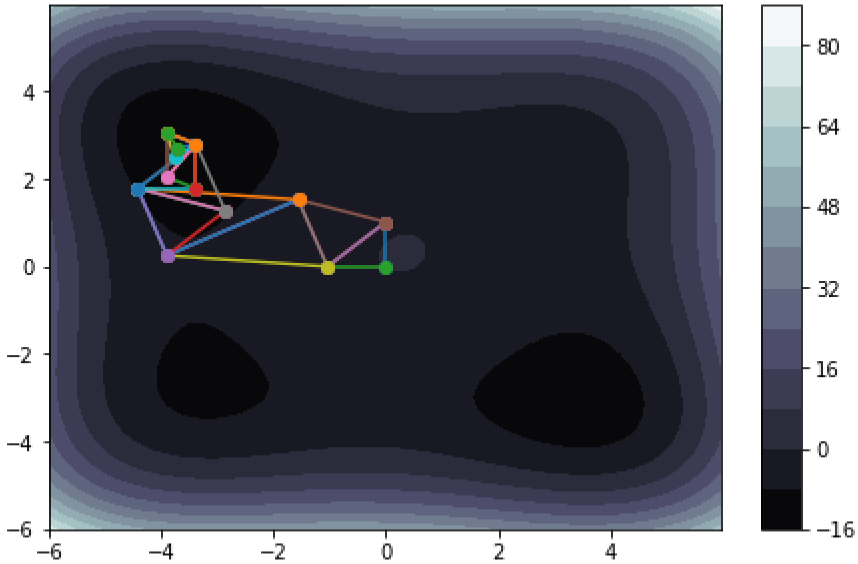

Fig 5. The contour image of the function with triangles of each iteration