Q1:

a)

(i) 10/10

(ii) 10/10

(iii) 10/10

(iv) 10/10

(v) 2/5

(vi) 2/5

b) 10/10

Q2:

a) 20/20

b) 20/20

94/100

Problem 1

a)

i) Since, the cost of transport is directly proportional to Euclidian distance between the factories, the weights should be equal to each other. For simplicity, let them all be $w_i = w = 1$.

ii) Defining Notation:

Anchor Points $\bar{a}_1, \ldots, \bar{a}_4$ Positions of factories

Weight $w_i = w = 1$

Minimal point $\bar{x} \in \mathbb{R}^2$

$\bar{x}$ is the optimal point for building the assembly line.

Minimization Problem:

$$\min_{\bar{x} \in \mathbb{R}^2} \left\{ f(\bar{x}) = \sum_{i=1}^{4} \| \bar{x} - \bar{a}_i \|_2 \right\}$$

We are minimising the total Euclidian distance from the assembly line to the factories (anchor points)

Let $\bar{x} = (x, y)$

Let $f = f_1 + f_2 + f_3 + f_4$

$$\min_{\bar{x} \in \mathbb{R}^2} \left\{ \sum_{i=1}^{4} \| \bar{x} - \bar{a}_i \|_2 \right\} =$$

Note that $\bar{x}$ is assumed to be not an anchor point

$$\min_{\bar{x} \in \mathbb{R}^2} \{ \underbrace{\sqrt{(x-1)^2 + (y-1)^2}}_{f_1} + \underbrace{\sqrt{(x-1)^2 + (y-3)^2}}_{f_2} + \underbrace{\sqrt{(x-2)^2 + (y-5)^2}}_{f_3} + \underbrace{\sqrt{(x-3)^2 + (y-1)^2}}_{f_4} \}$$

To find the minimal point we need to take $\nabla f = \bar{0} = \begin{bmatrix} \partial f / \partial x \\ \partial f / \partial y \end{bmatrix} = \begin{bmatrix} \partial f_1/\partial x + \partial f_2/\partial x + \partial f_3/\partial x + \partial f_4/\partial x \\ \partial f_1/\partial y + \cdots + \partial f_4/\partial y \end{bmatrix}$

$$\frac{\partial}{\partial x} f_1 = \frac{(x-1)}{\sqrt{(x-1)^2 + (y-1)^2}} \qquad \frac{\partial f_2}{\partial x} = \frac{x-1}{\sqrt{(x-1)^2 + (y-3)^2}}$$

$$\frac{\partial f_3}{\partial x} = \frac{x-2}{\sqrt{(x-2)^2 + (y-5)^2}} \qquad \frac{\partial f_4}{\partial x} = \frac{x-3}{\sqrt{(x-3)^2 + (y-1)^2}}$$

$$\frac{\partial f_1}{\partial y} = \frac{y-1}{\sqrt{(x-1)^2 + (y-1)^2}} \qquad \frac{\partial f_2}{\partial y} = \frac{y-3}{\sqrt{(x-1)^2 + (y-3)^2}}$$

$$\frac{\partial f_3}{\partial y} = \frac{y-5}{\sqrt{(x-2)^2 + (y-5)^2}} \qquad \frac{\partial f_4}{\partial y} = \frac{y-1}{\sqrt{(x-3)^2 + (y-1)^2}}$$

~~Using the fact that $\vec{x}$.~~

~~Using~~ Rewriting this shorter:

$$\sum_{i=1}^{4} \frac{\bar{x} - \bar{a}_i}{\|\bar{x} - \bar{a}_i\|} = 0$$

Using the fact that $\nabla \|\bar{x}\|_2 = \frac{\bar{x}}{\|\bar{x}\|_2}$

$$\left(\sum_{i=1}^{4} \frac{1}{\|\bar{x} - \bar{a}_i\|} \right) \vec{x} = \sum_{i=1}^{4} \frac{a_i}{\|\bar{x} - \bar{a}_i\|}$$

$\therefore$

$$\vec{x} = \frac{1}{\sum_{i=1}^{4} \frac{1}{\|\bar{x} - \bar{a}_i\|}} \sum_{i=1}^{4} \frac{\bar{a}_i}{\|\bar{x} - \bar{a}_i\|}$$

~~Rewriting~~ Rewriting $\bar{x} = T(\bar{x})$

$$T(\bar{x}) = \frac{1}{\sum_{i=1}^{4} \frac{1}{\|\bar{x} - \bar{a}_i\|}} \sum_{i=1}^{4} \frac{\bar{a}_i}{\|\bar{x}_i - \bar{a}_i\|}$$

Finding the minimal point $\bar{x}$ is equivalent to solving the fixed point iteration problem

$$\bar{x}^{k+1} = T(\bar{x}^k)$$ Weiszfeld Method.

iii)

$\bar{x}^{k+1} = T(\bar{x}^k)$ can also be written as:

$$\bar{x}^{k+1} = \bar{x}^k - \frac{1}{\sum_{i=1}^{4} \frac{1}{\|\bar{x}^k - \bar{a}_i\|}} \sum_{i=1}^{4} \frac{\bar{x}^k - \bar{a}_i}{\|\bar{x}^k - \bar{a}_i\|}$$

Looks like gradient descent. It has a unique solution iff $f(x)$ has strong convexity.

The 1st and 2nd order conditions for convexity:

$$f(y) \geq f(x) + \nabla f(x)^T (y - x)$$ 1st Order

Hessian of f is positive semidefinite for $\forall \bar{x}$ 2nd order

$$\bar{x}^T \nabla^2 f(x) \bar{x} \geq 0$$

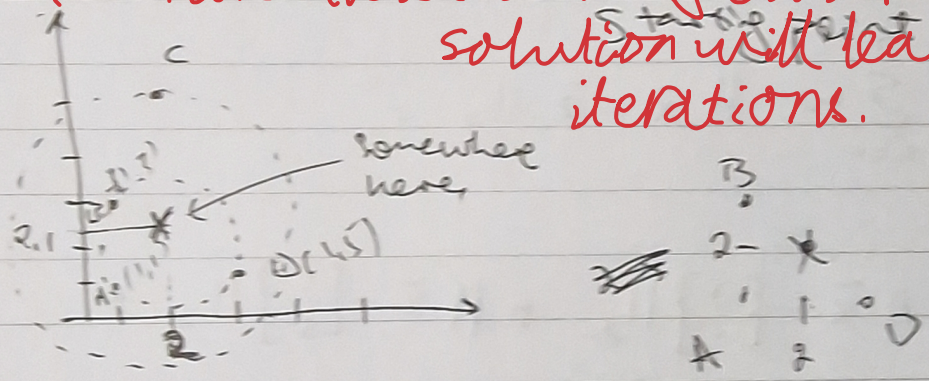
iv) [1.40005548 , 2.59999102], 50 iterations

v) No, $\bar{x}$ is assumed to be not equal to the anchor points

Need to mention that the anchor points can't be used

vi) Choose a starting point in the middle one factories:

Need to state that starting closer to optimal solution will lead to fewer iterations.

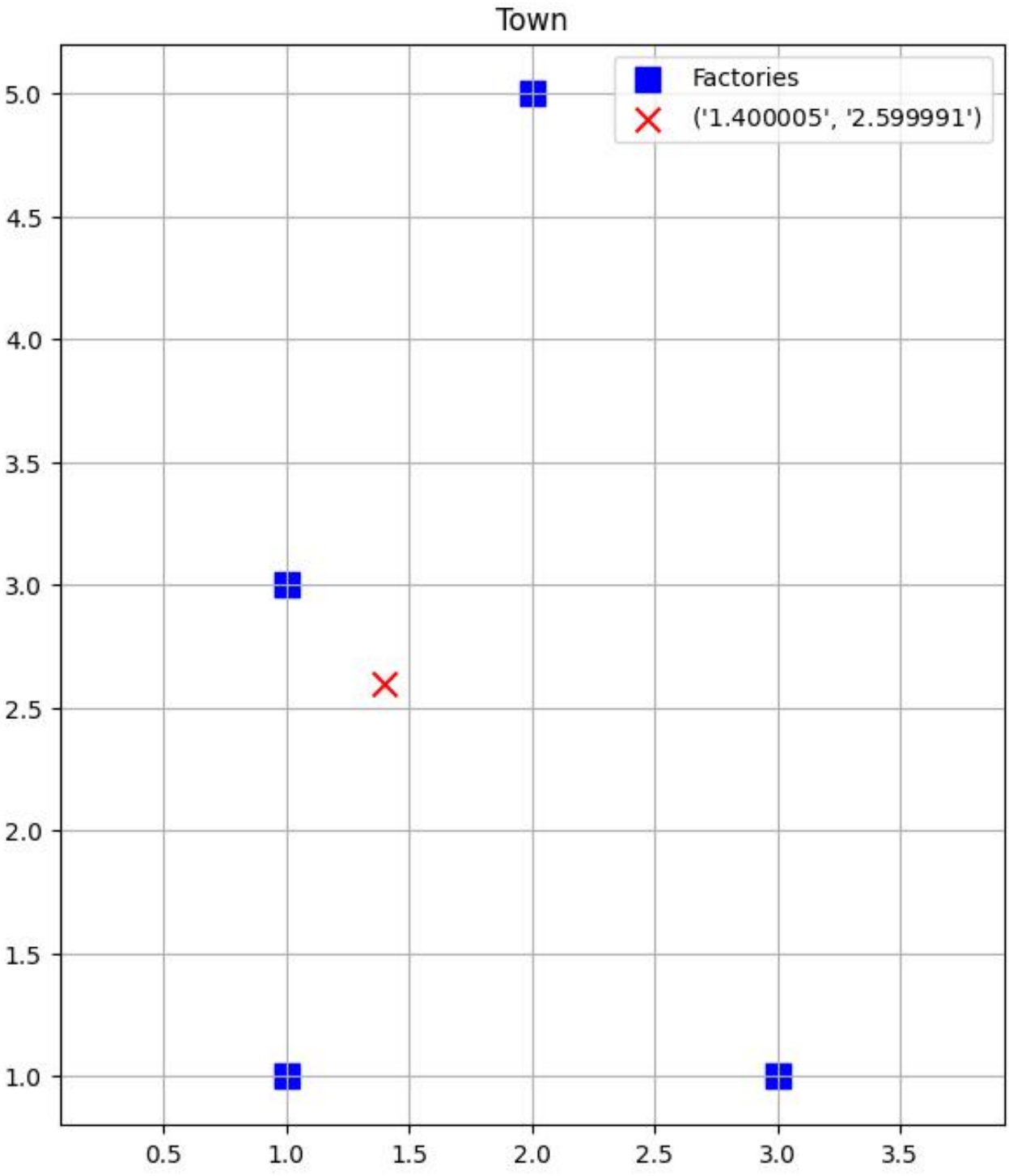


Plugging into the algorithm:

(2, 2.4) ~ 47 iterations

(2, 2) ~ 49

Original ~ 50 iterations


Town
Factories
('1.400005', '2.599991')
5.0
4.5
4.0
3.5
3.0
2.5
2.0
1.5
1.0
0.5
1.0
1.5
2.0
2.5
3.0
3.5

b)

The new weight:

$w_1 = 1.5$, $w_2 = w_3 = w_4 = 1.0$

$$h(\bar{x}) = \sum_{i=1}^{4} w_i \cdot$$

$$h(\bar{x}) = \sum_{i=1}^{4} w_i \| \bar{x} - \bar{a}_i \|_2 =$$

$$= 1.5 \| \bar{x} - \begin{bmatrix} 1 \\ 1 \end{bmatrix} \|_2 + \| \bar{x} - \begin{bmatrix} 1 \\ 3 \end{bmatrix} \|_2 +$$

$$+ \| \bar{x} - \begin{bmatrix} 2 \\ 5 \end{bmatrix} \|_2 + \| \bar{x} - \begin{bmatrix} 3 \\ 1 \end{bmatrix} \|_2$$

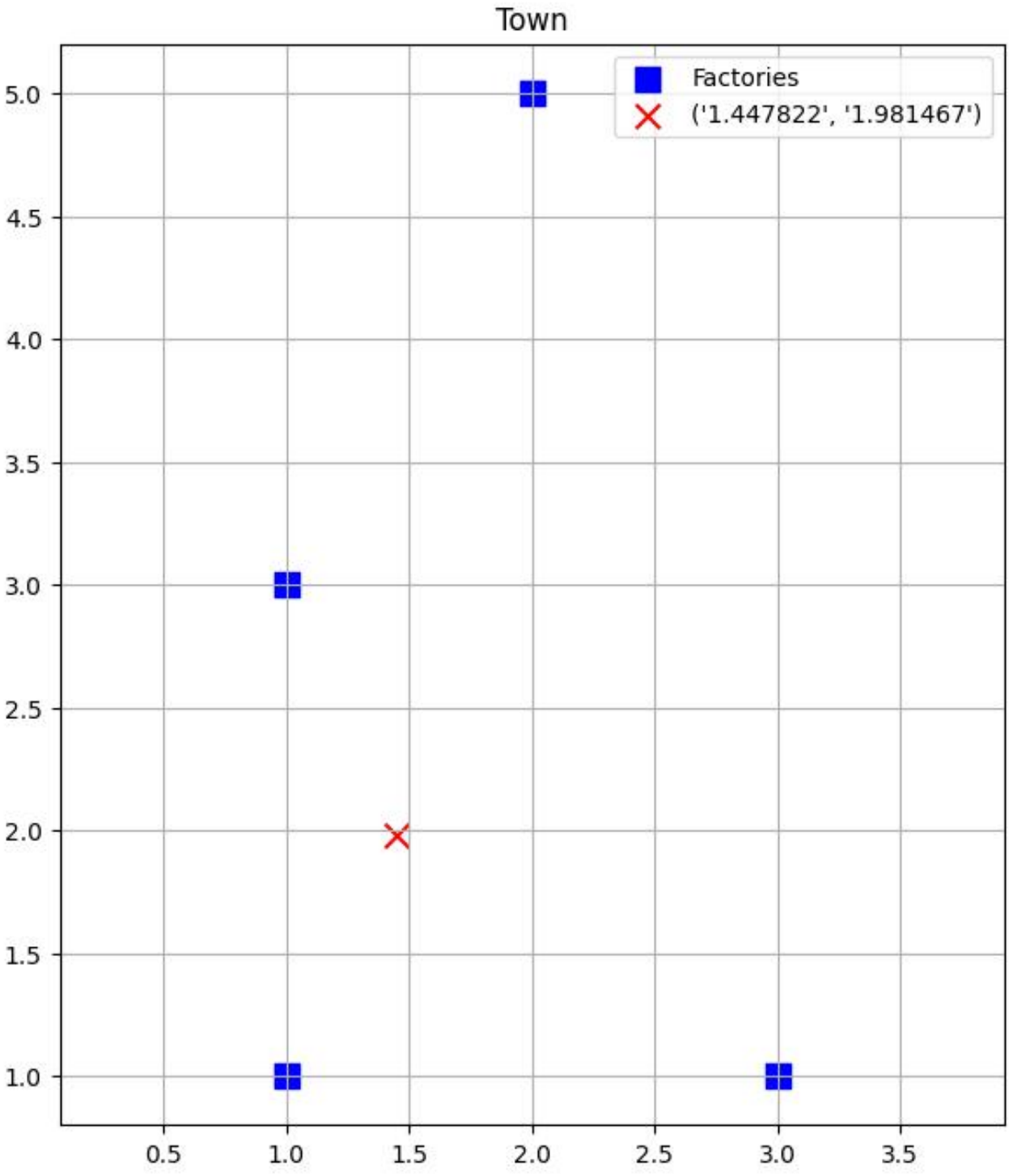
The Algorithm: Weiszfeld Method

```
def weiszfeld(x_k, w, a):
    grad_k, summa = grad_h(x_k, w, a)
    num = 0
    while np.linalg.norm(grad_k) > 1e-5:  #stopping criterion
        grad_k, summa_k = grad_h(x_k, w, a)
        x_k += (-1/summa_k) * grad_k
        num += 1
    return x_k, num
```

where grad_h is the function of ∇h.

This gives

[1.44782162, 1.98146687], 37 iterations


Town
Factories
('1.447822', '1.981467')
5.0
4.5
4.0
3.5
3.0
2.5
2.0
1.5
1.0
0.5
1.0
1.5
2.0
2.5
3.0
3.5

2.

$$\frac{\partial f}{\partial x} = -200(w - x) + 20.2(x - 1) + 19.8(z - 1)$$

$$\frac{\partial f}{\partial y} = 2(y - 1) + 360y(y^2 - z)$$

$$\frac{\partial f}{\partial z} = -180(y^2 - z) + 20.2(z - 1) + 19.8(x - 1)$$

$$\frac{\partial f}{\partial w} = 400w(w^2 - x) + 2(w - 1)$$

$$\nabla^2 f = \begin{bmatrix} 220.2 & 0 & 19.8 & -400w \\ 0 & 2y + 1080y^2 - 360z & -360z & 0 \\ 19.8 & -360y & 200.2 & 0 \\ -400w & 0 & 0 & 1200w^2 - 400x + 2 \end{bmatrix}$$

Putting this into python (NEXT PAGE)

Solution a) [1, 1, 1, 1]

b) 12 iterations

```
1  import numpy as np
2
3
4  # Defining The hessian
5  def Hessian(x,y,z,w):  # Output: 4x4 np array
   :
14 # defining The grad
15 def grad_f(x,y,z,w): # Output 4d np array
   :
23 # defining d
24 def d_k(x,y,z,w):
25     grad = grad_f(x,y,z,w)
26     hess = Hessian(x,y,z,w)
27     return np.linalg.solve(hess, -grad), grad
28
29
30 # Pure Newton Algorithm
31 def Pure_Newton(x,y,z,w):
32     xk = np.array([x,y,z,w])
33     d, gradf = d_k(xk[0], xk[1], xk[2], xk[3])
34     num = 0
35     while np.linalg.norm(gradf) > 1e-5:
36         num += 1
37         xk += d
38         d, gradf = d_k(xk[0], xk[1], xk[2], xk[3])
39
40     return xk, num
41
42 print(Pure_Newton(0.0, 1.0, 2.0, 3.0))
```