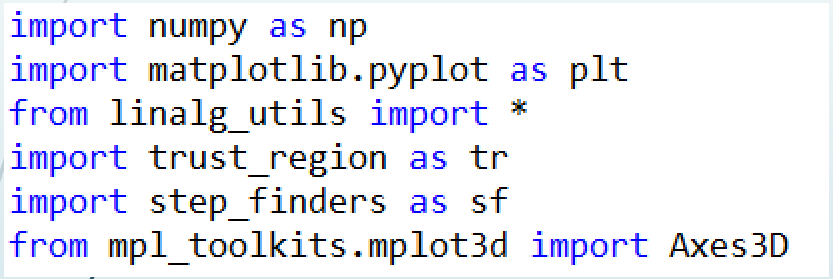
Constrained Optimization

WQD7011 Numerical Optimization

Let’s finish our tasks before mid term break...

- Trust Region exercises
- Introduction to Unconstrained Optimization
- Mid Term Test Answer
- Announcement of Assignment II

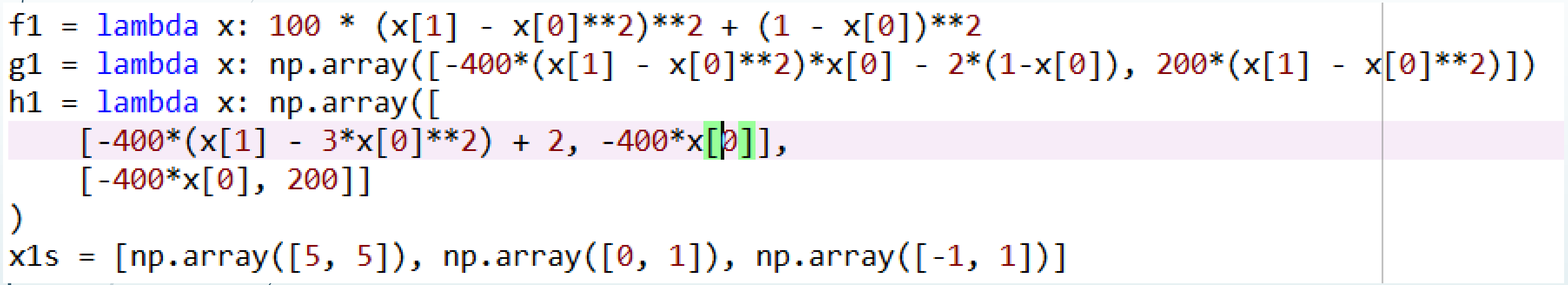
Trust Region Exercise - *The Code*



```
import numpy as np
import matplotlib.pyplot as plt
from linalg_utils import *
import trust_region as tr
import step_finders as sf
from mpl_toolkits.mplot3d import Axes3D
```

- Importing libraries:
  - From Python:
    - numpy
    - matplotlib
    - mpl_toolkits
  - From user:
    - linalg_utils
    - trust_region
    - step_finders

Trust Region Exercise - *The Code*



```
f1 = lambda x: 100 * (x[1] - x[0]**2)**2 + (1 - x[0])**2
g1 = lambda x: np.array([-400*(x[1] - x[0]**2)*x[0] - 2*(1-x[0]), 200*(x[1] - x[0]**2)])
h1 = lambda x: np.array([
    [-400*(x[1] - 3*x[0]**2) + 2, -400*x[0]],
    [-400*x[0], 200]]
)
x1s = [np.array([5, 5]), np.array([0, 1]), np.array([-1, 1])]
```

- Declaring and assigning operations (using lambda) / values to:
  - Objective function (f1)
  - First-order partial derivatives (g1) – the gradient information
  - Second-order partial derivatives (h1) – the hessian information
  - 3 sets of initial points (x1s)

Trust Region Exercise - *The Code*

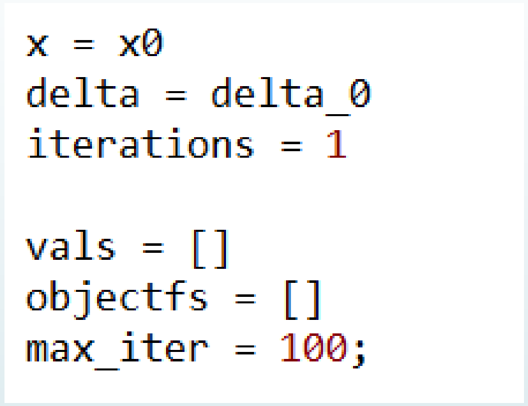
```
x, fx, iterations, vals, objectfs = tr.trust_region(f1, g1, h1, x1s[0], .1, 1,
.25, sf.cauchy_point_step_finder)
```

- Calling trust_region function from tr file.
- Passing parameters for trust_region function:
  - Objective function (f1)
  - First-order partial derivatives (g1)
  - Second-order partial derivatives (h1)
  - The first set of starting point (x1s[0])
  - delta_0
  - max_delta
  - etha
  - Chosen step_finder algorithm from sf file

$\left.\begin{array}{l}\text{delta\_0}\\ \text{max\_delta}\\ \text{etha}\end{array}\right\}$ Refer to our example in Lecture 06 for these variables

```
def trust_region(f, g, hf, x0, delta_0, max_delta, etha, step_finder,
repair_hessian=True, eps=1e-5):#
```

Trust Region Exercise - *The Code*

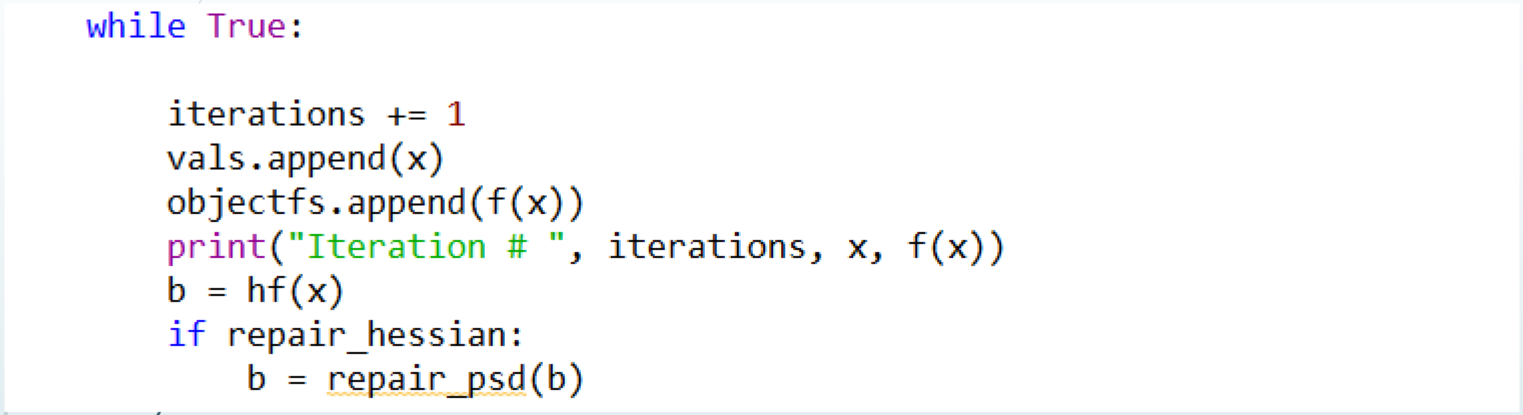


```
x = x0
delta = delta_0
iterations = 1

vals = []
objectfs = []
max_iter = 100;
```

- In trust_region function:
  - Declare and assign variables

Trust Region Exercise - *The Code*

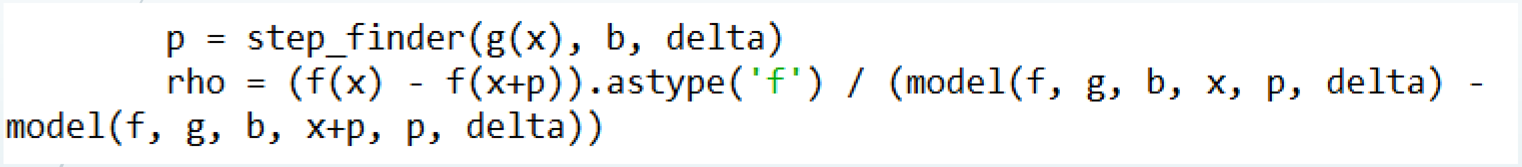


```
while True:

    iterations += 1
    vals.append(x)
    objectfs.append(f(x))
    print("Iteration # ", iterations, x, f(x))
    b = hf(x)
    if repair_hessian:
        b = repair_psd(b)
```

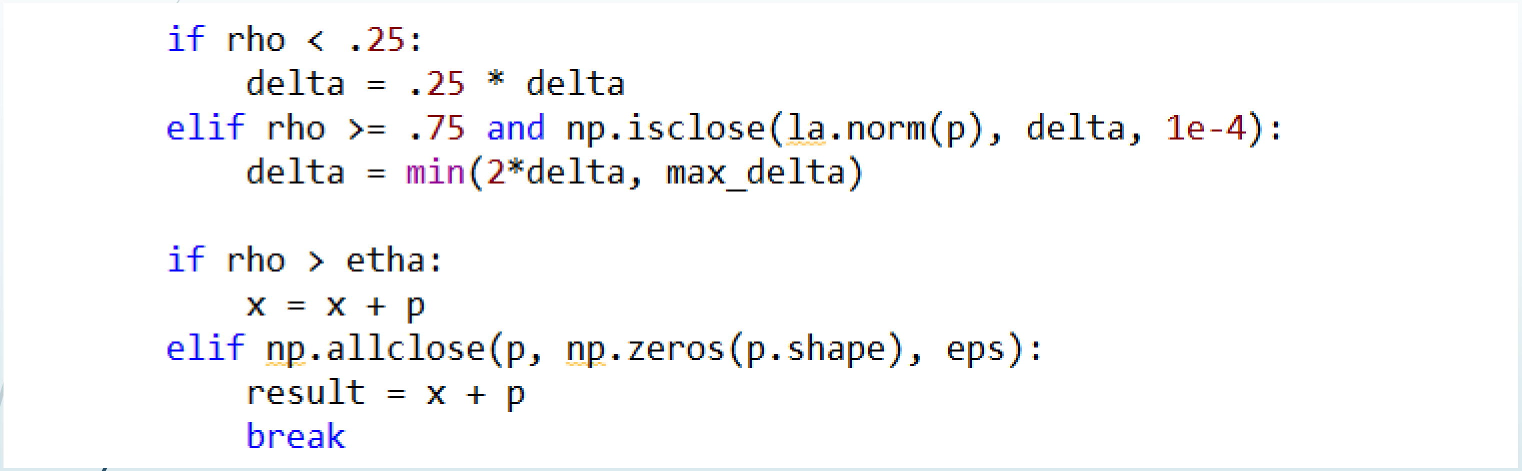
- In trust_region function:
  - while the status is still True, loop until it breaks
  - Add iteration number with 1
  - Append current starting points to vals
  - Append current value of f(x) to objectfs
  - Calculate hf(x) → the second order derivative functions and store it as b.
  - Hessian needs to be semi positive definite (for minimization) → repair if not

Trust Region Exercise - *The Code*


```
        p = step_finder(g(x), b, delta)
        rho = (f(x) - f(x+p)).astype('f') / (model(f, g, b, x, p, delta) - 
model(f, g, b, x+p, p, delta))
```

- In trust_region function:
  - Calculating step using and assign it to p (our direction)
  - Calculate the ratio → refer to the formula in Lecture 6, and assign it to rho.

Trust Region Exercise - *The Code*



```python
if rho < .25:
    delta = .25 * delta
elif rho >= .75 and np.isclose(la.norm(p), delta, 1e-4):
    delta = min(2*delta, max_delta)

if rho > etha:
    x = x + p
elif np.allclose(p, np.zeros(p.shape), eps):
    result = x + p
    break
```

- In trust_region function:
  - Refer back to our algorithm in Lecture 6 for this.

Exercise – the Q&A

- **Use one sentence to describe the purpose of each function in file (b), (c), and (d).**
- In trust_region.py:
  - model – calculate the values for m(x)
  - trust_region – main method for trust_region algorithm
- step_finders.py
  - cauchy_point_step_finder – cauchy point method
  - solve_taw_for_dogleg – calculating tau for dogleg
  - dogleg_step_finder – dogleg method

Exercise – the Q&A

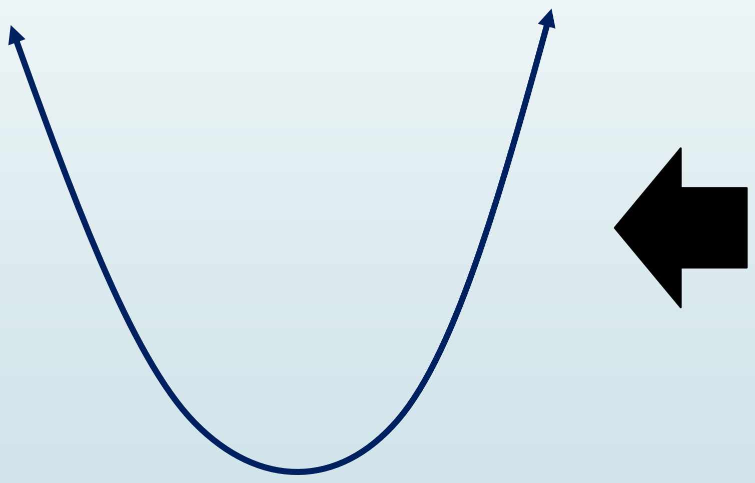
- **Modify the code to display ρ for each iteration.**
  - (goto trust_region function and display rho)
- What is(are) the convergence criteria(s) for the trust region method?
- Modify the code the use different step finder, and record the ρ.
- From the answers in Q4, compare the recorded results, and state your observation(s).

Constrained Optimization

- Introduction to Constrained Optimization
- Basic Linear Optimization
- Possible applications in data science

Constrained Optimization

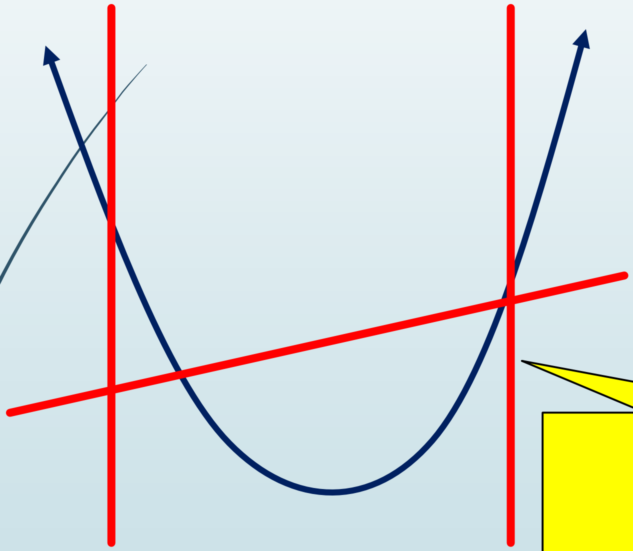
- First part of the course, unconstrained optimization:



No boundary or soft boundary(ies)

Constrained Optimization

- Second part of the course, ~~un~~**constrained optimization**:

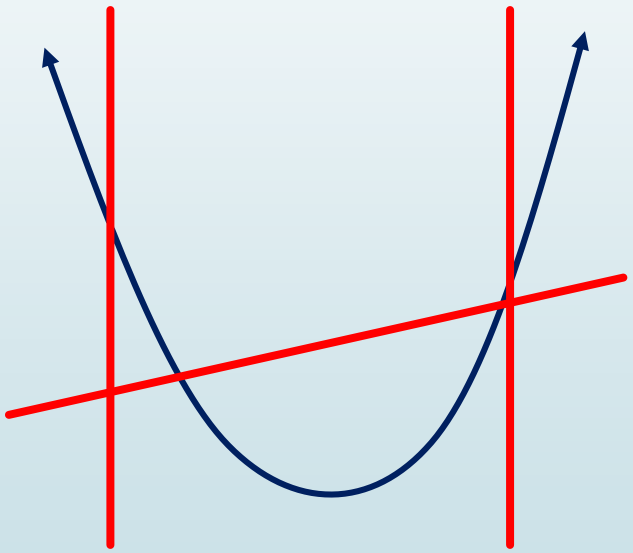


Has boundaries
- Hard limit put on variables / coefficients
- Limit infinity direction
- Limit the solution space / feasible region

Red lines –
Constraints!

Constrained Optimization

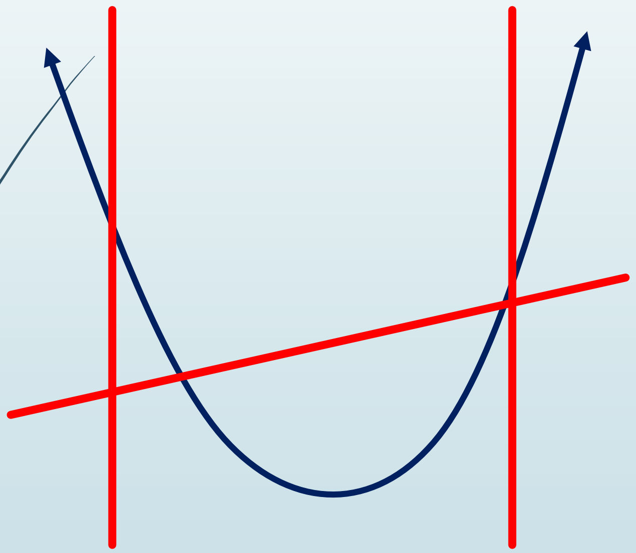
- Second part of the course, ~~un~~**constrained optimization**:



What can you observed from this figure?

Constrained Optimization

- Second part of the course, ~~un~~**constrained optimization**:



What can you observed from this figure?

- Objective function is of type quadratic.
- 3 constraints

Constrained Optimization *The General Representation*

- Second part of the course, ~~un~~**constrained optimization**:

$$\min_{x \in R^n} f(x) \text{ subject to } \begin{cases} c_i(x) = 0, & i \in \epsilon \\ c_i(x) \geq 0, & i \in \tau \end{cases}$$

Where f and the function c_i are all smooth, real-valued functions on a subset of $\mathbb{R}^n$, and τ and ϵ are two finite sets of indices.

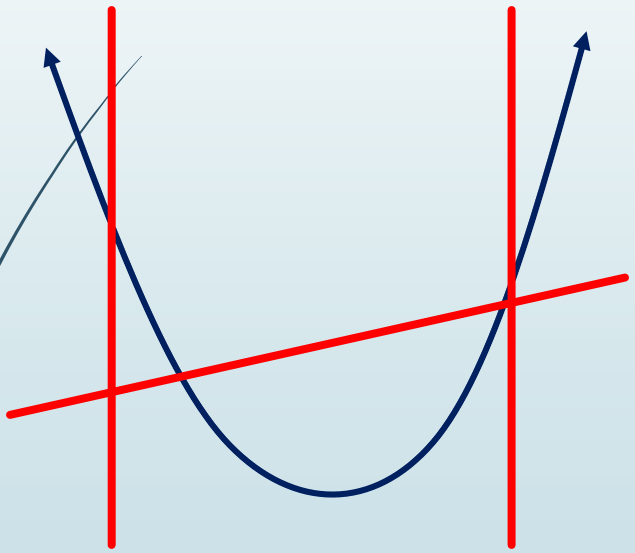
objective function

equality constraint

inequality constraint

Constrained Optimization

- Second part of the course, ~~un~~**constrained optimization**:

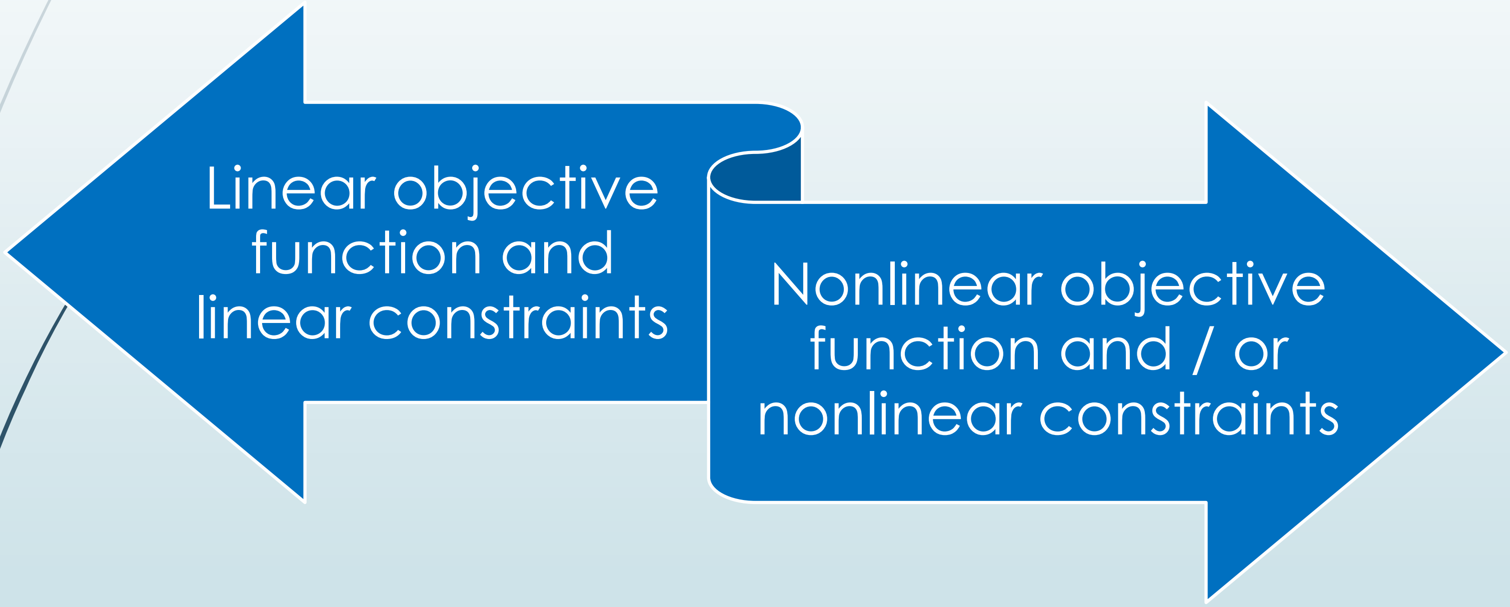


Can you find possible solution space from this graph?

What about the possible optimum value(s)?

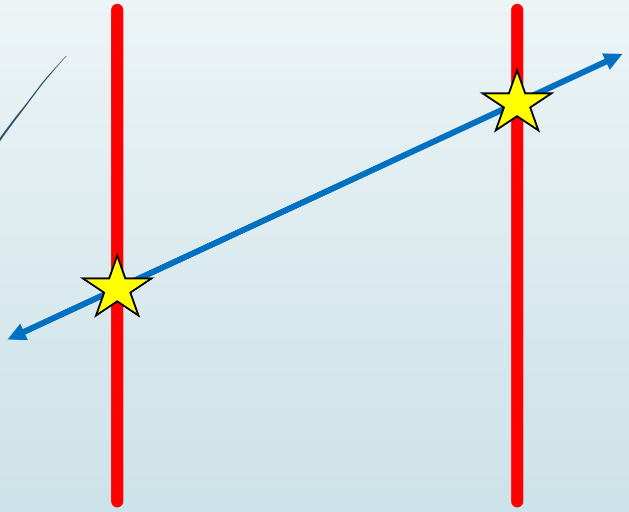
Constrained Optimization *Linear vs Nonlinear*

- Linear optimization versus non-linear optimization


Linear objective function and linear constraints
Nonlinear objective function and / or nonlinear constraints

Constrained Optimization *Optimum Values*

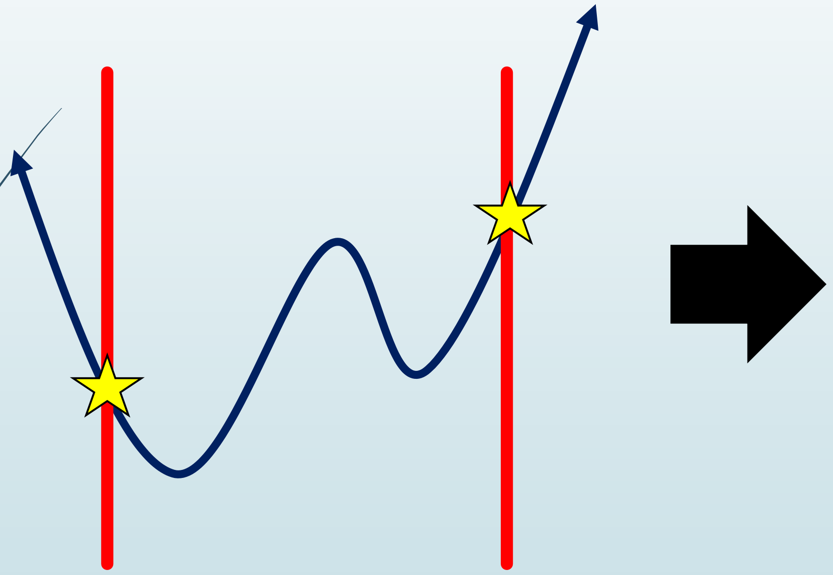
- Linear optimization



Optimum values can only occurred at the **boundaries**.

Constrained Optimization
Optimum Values

- Nonlinear optimization



Optimum values can occurred at the **boundaries** OR **between boundaries**.

Constrained Optimization Constraints

- Let's start with the basic, how to write constraints?
- Simple example:

If a single sausage-bun costs RM 4, then two sausage-bun cost RM 8, five sausage-bun cost RM 20, and x_1 sausage-bun cost $4x_1$.
If you buy x_1 sausage-bun at RM 4 and x_2 Pepsi at RM 2, then your total cost will be $4x_1 + 2x_2$.
If you only have RM 70 to spend at this food store, then your total cost must be…?

Constrained Optimization Constraints

- Let's start with the basic, how to write constraints?
- Simple example:

If a single sausage-bun costs RM 4, then two sausage-bun cost RM 8, five sausage-bun cost RM 20, and x_1 sausage-bun cost $4x_1$.
If you buy x_1 sausage-bun at RM 4 and x_2 Pepsi at RM 2, then your total cost will be $4x_1 + 2x_2$.
If you only have RM 70 to spend at this food store, then your total cost must be…?

linear inequality

boundary

$$\underbrace{4x_1 + 2x_2 \le 70}$$

Linear Constraint

Constrained Optimization
LO - Practice

- Simple practice:

a. A batch of cookies requires 3 cups of flour, and a cake requires 4. Write a constraint limiting the amount of cookies and cakes that can be made with 24 cups of flour.

b. Box type 1 can hold 20 books and box type 2 can hold 12. Write a constraint for the number of boxes needed in order to box up 100 books.

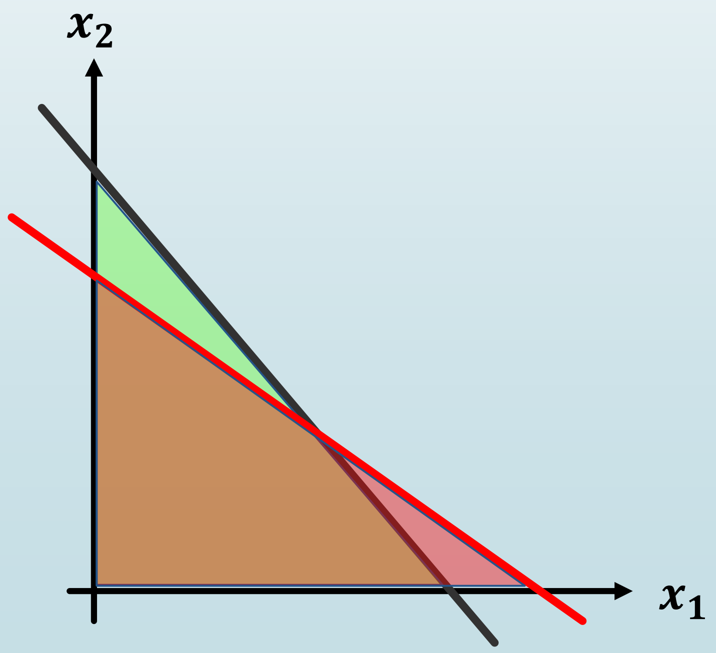
c. If it takes you 4 minutes to bike a mile, 9 minutes to run a mile and 14 minutes to walk a mile, write a constraint that limits how many miles of each type of exercise you can get in a 45-minute lunch break.

Constrained Optimization
LO – Graphical Solution

- constraints:

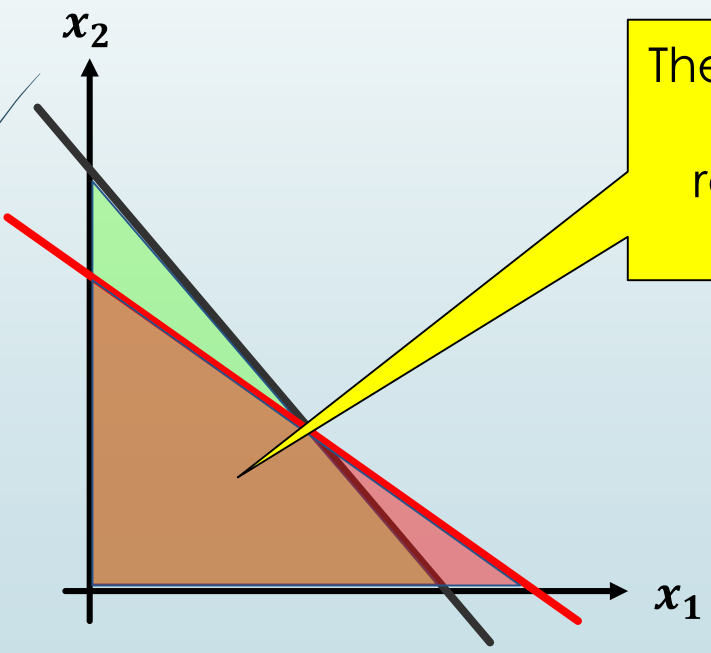
$$\begin{aligned} ax_1 + bx_2 &\le y_1 \\ cx_1 + dx_2 &\le y_2 \end{aligned}$$

- x_1 and x_2 must be ≥ 0.



Constrained Optimization *LO – Graphical Solution*

- Feasible region / solution space?

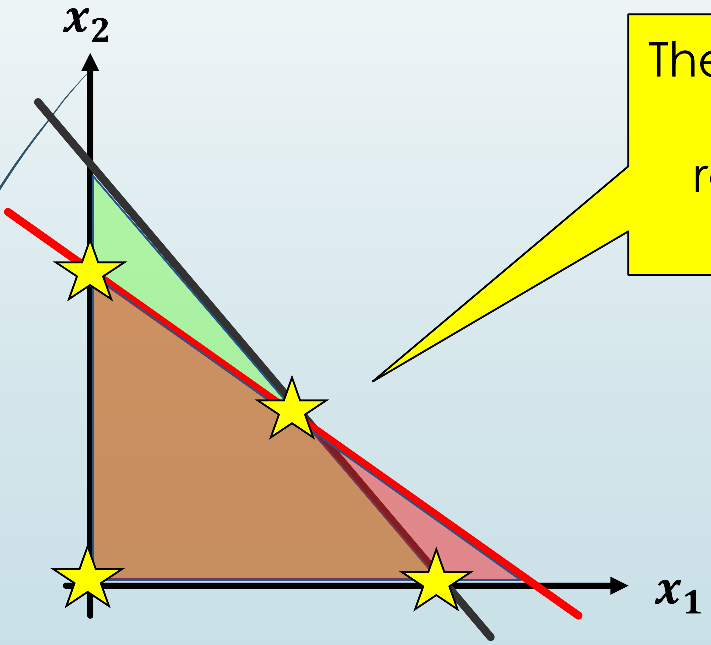


The overlapped are is the feasible region / solution space!

Where should we find the optimum values for x_1 and x_2?

Constrained Optimization *LO – Graphical Solution*

- Feasible region / solution space?



The overlapped are is the feasible region / solution space!

Where should we find the optimum values for x_1 and x_2?

Constrained Optimization
LO – Graphical Solution

An entrepreneur is starting her own business by selling various soaps and essential oils to the local shops. According to current market demand, she is expected to produce 200 soups and 150 essential oils every day. Nevertheless, due to the space and tools limitation, the maximum production for her products is 300 soaps and 200 essential oils per day. In terms of shipping, she has to ship-out at least 375 products each day in fulfilling her shipping contract.

Assume that each soup sold loses RM 2 profits and each essential oil sold earns RM 10,

a) Construct a linear program model with objective function, equality or inequality constraints to maximize her net profits.

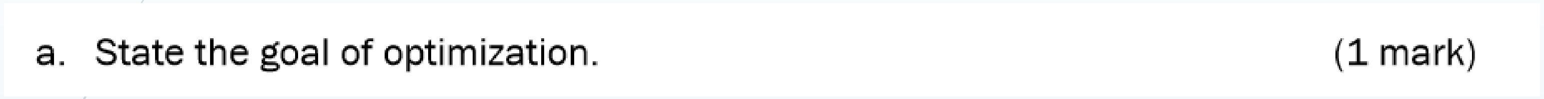
b) Draw a graph to show the feasible region of the possible solution space for model in Question a).

c) Calculate the number of each product she should produce daily to maximize her net profits.

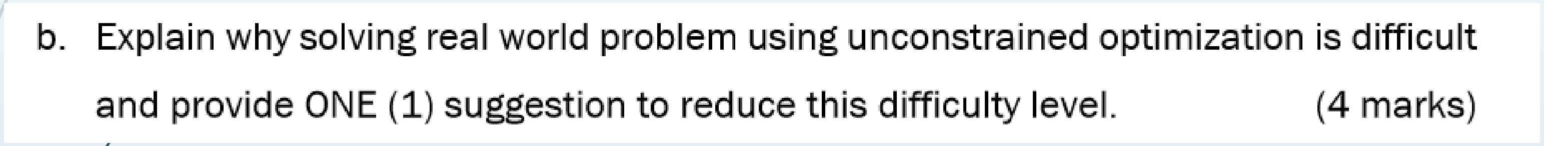
Constrained Optimization
LO – Graphical Solution

- Discussion:
    - In real life, do we solve linear problem in graphical way? Justify your answer.
    - What are the real-life industry examples (probably in your company) in data science that needed the use of optimization algorithm?

Mid Term Exam Discussion

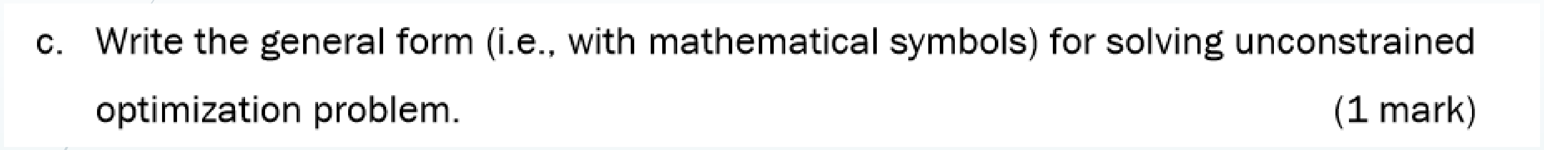

a. State the goal of optimization. (1 mark)

Goal: find the values of variables that optimize the objective.

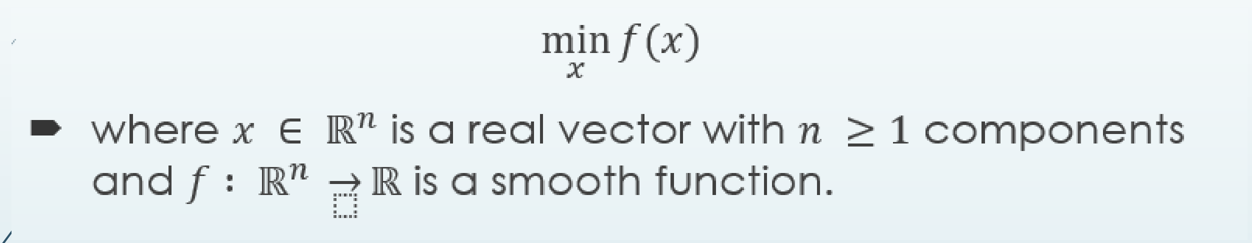

b. Explain why solving real world problem using unconstrained optimization is difficult and provide ONE (1) suggestion to reduce this difficulty level. (4 marks)

No universal algorithm.
Difficult to find global min or max because infinite solution space.
Others.

Mid Term Exam Discussion



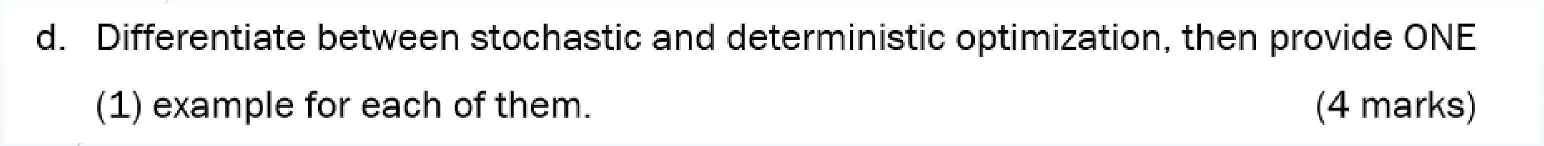
c. Write the general form (i.e., with mathematical symbols) for solving unconstrained optimization problem. (1 mark)



$$\min_{x} f(x)$$

- where $x \in \mathbb{R}^n$ is a real vector with $n \geq 1$ components and $f : \mathbb{R}^n \to \mathbb{R}$ is a smooth function.

Mid Term Exam Discussion


d. Differentiate between stochastic and deterministic optimization, then provide ONE (1) example for each of them. (4 marks)

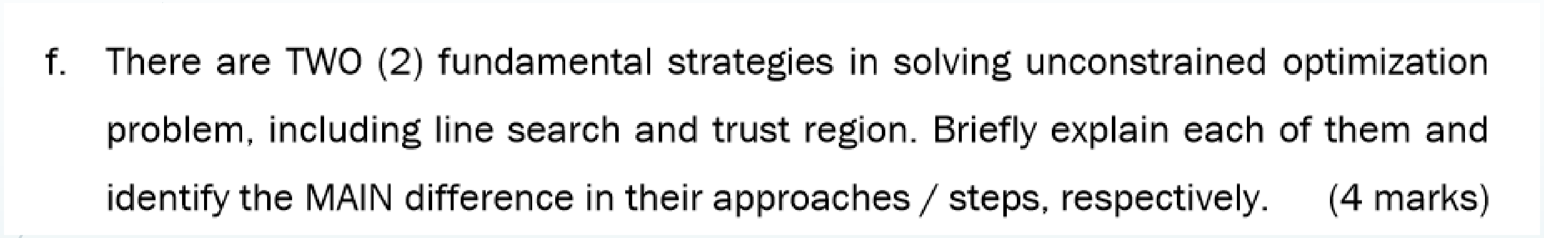
- Stochastic Optimization
  - Model cannot be fully specified because it depends on quantities that are unknown at the time of formulation.
  - Example: future interest rates, future demands for a product.
  - Use quantifications (based on users knowledge) of the uncertainty to produce solutions that optimize the expected performance of the model.
- Deterministic Optimization
  - Model is completely known

Mid Term Exam Discussion

e. Differentiate between local and global solution by using figure. Which solution is more difficult to achieve? Justify your answer. (5 marks)

- Local solution
  - A point at which the objective function is smaller than at all other feasible nearby points
- Global solution
  - The point with lowest function value among ALL feasible points.
  - Difficult to recognize and locate

Mid Term Exam Discussion

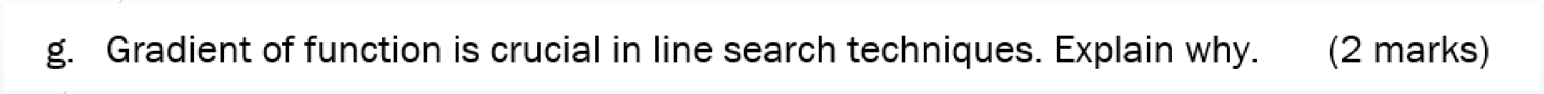


f. There are TWO (2) fundamental strategies in solving unconstrained optimization problem, including line search and trust region. Briefly explain each of them and identify the MAIN difference in their approaches / steps, respectively. (4 marks)

Line search finds direction, then step size while trust region in general finds step size (radius) and then direction.

Refer to Lecture 2 for more information.

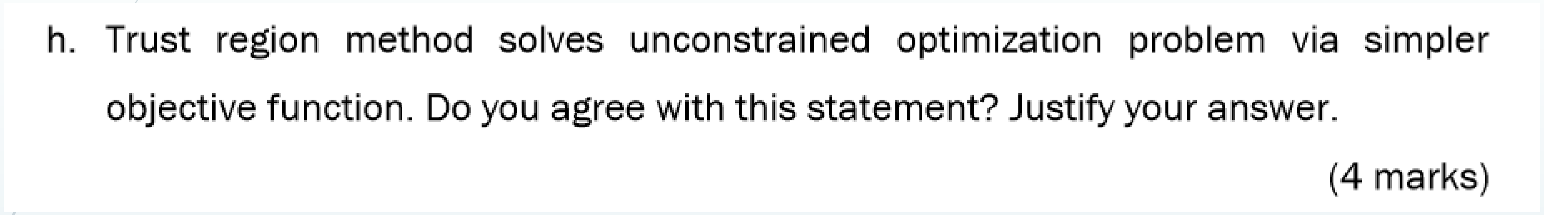
Mid Term Exam Discussion



g. Gradient of function is crucial in line search techniques. Explain why. (2 marks)

Gradient will determine the direction towards local minima.
Or other answer.

Mid Term Exam Discussion

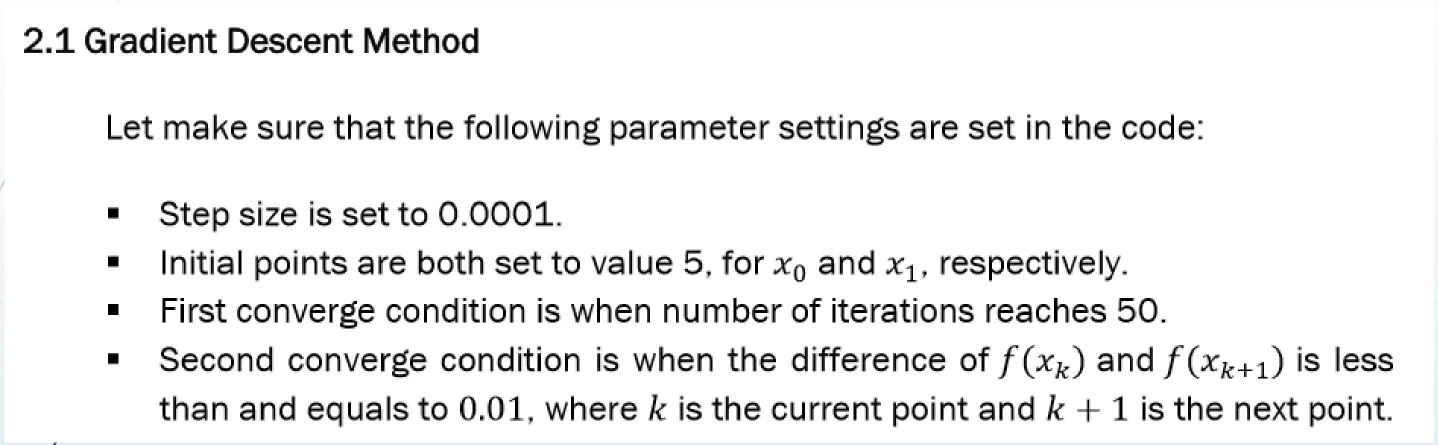


h. Trust region method solves unconstrained optimization problem via simpler objective function. Do you agree with this statement? Justify your answer.

(4 marks)

Yes or No [1 mark]
Justification – divide into n-dimensional, divide into constrained optimizations, etc…

Mid Term Exam Discussion



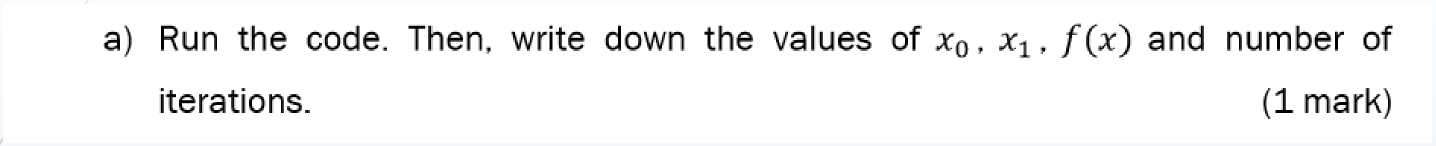
2.1 Gradient Descent Method

Let make sure that the following parameter settings are set in the code:

- Step size is set to 0.0001.
- Initial points are both set to value 5, for x_0 and x_1, respectively.
- First converge condition is when number of iterations reaches 50.
- Second converge condition is when the difference of $f(x_k)$ and $f(x_{k+1})$ is less than and equals to 0.01, where k is the current point and $k+1$ is the next point.

Basically this is the settings given in the provided codes…

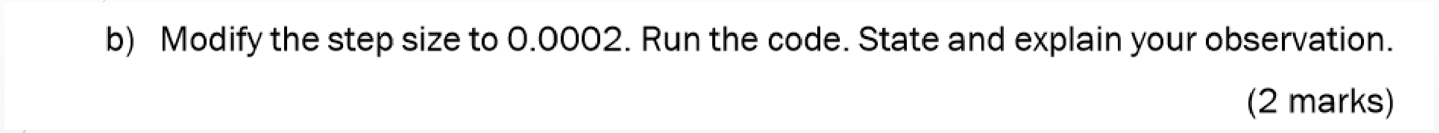
Mid Term Exam Discussion



a) Run the code. Then, write down the values of x_0, x_1, $f(x)$ and number of iterations. (1 mark)

```
[0.9992 5.4   ] 1937.4076932352416 1
[1.17512328 5.31196801] 1545.3486587826624 2
[1.35986715 5.23334695] 1145.3483938946076 3
[1.54387268 5.16566478] 774.3160368917922 4
[1.71557359 5.11002234] 470.0270987692811 5
[1.8641247  5.06668575] 254.1055126835174 6
[1.98263882 5.03485125] 122.84597900325505 7
[2.06999519 5.01277136] 54.127460033900505 8
[2.13005045 4.99821354] 22.538207813017163 9
[2.16911097 4.98899156] 9.429532738570803 10
[2.19351384 4.98331258] 4.376329814634371 11
[2.20834981 4.97987639] 2.522400568669064 12
[2.2172125  4.97781504] 1.863329534402812 13
[2.22244857 4.97657936] 1.633522351280844 14
[2.22552013 4.97583333] 1.5543108119816416 15
[2.22731302 4.97537546] 1.5271837782347222 16
```

Mid Term Exam Discussion

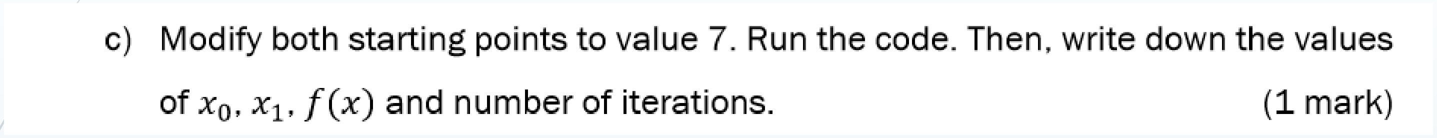

b) Modify the step size to 0.0002. Run the code. State and explain your observation.

(2 marks)

```
[-3.0016  5.8   ] 1046.167661875855 1
[-2.22928392  5.9283841 ] 102.33449541515843 2
[-2.39896532  5.89003701] 13.375530404769872 3
[-2.42351502  5.88463691] 11.733025673237677 4
```

When the step size is set to larger value, it tends to convert faster.
The result is not as good as the previous result in (a).
And others…then explain based on what you stated (link to the change of step size).

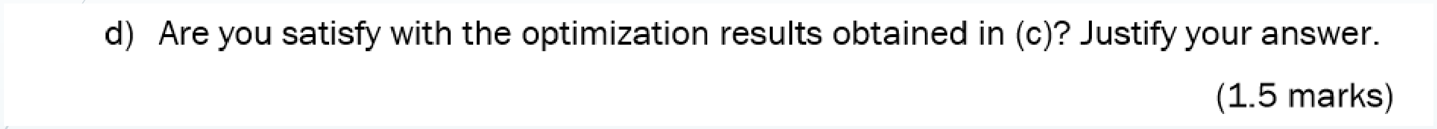
Mid Term Exam Discussion



c) Modify both starting points to value 7. Run the code. Then, write down the values of x_0, x_1, $f(x)$ and number of iterations. (1 mark)

```
[-4.7612  7.84  ] 22023.190975456735 1
[-1.93588952  8.13658051] 1934.87453298565 2
[-2.27516032  8.04880226] 835.8223119223701 3
[-2.53591645  7.99135331] 256.01281525175 5 4
[-2.69349925  7.96014369] 63.37341097927314 6 5
[-2.76873937  7.94603958] 22.05022360721308 4 6
[-2.799009    7.94043714] 15.55767417702908 7
[-2.8101154   7.93831742] 14.68977632494004 5 8
[-2.81402591  7.93748605] 14.58192824393634 3 9
[-2.81537297  7.93711116] 14.568705098340281 10
```

Mid Term Exam Discussion

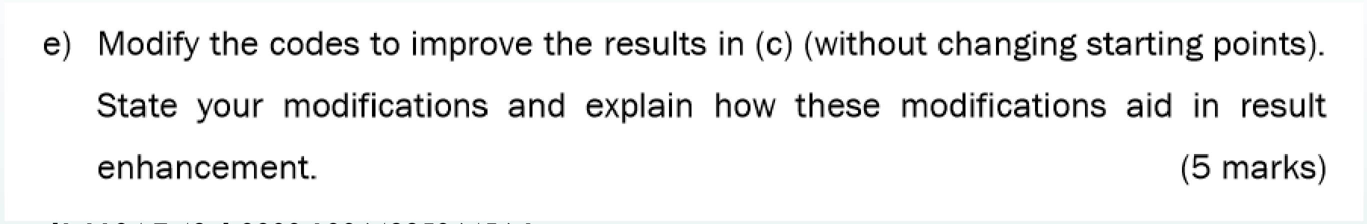


d) Are you satisfy with the optimization results obtained in (c)? Justify your answer.

(1.5 marks)

Converge faster, explain why.
Bad result that (a), explain why.
The explanation should link with the importance of selecting *suitable* starting points or how.

Mid Term Exam Discussion



e) Modify the codes to improve the results in (c) (without changing starting points). State your modifications and explain how these modifications aid in result enhancement. (5 marks)

```
[1.1194 7.42  ] 3803.1336422536456 1
[1.25745359 7.35833056] 3337.602121163604 2
[1.40271758 7.30055915] 2844.1897882876037 3
[1.55228956 7.24722973] 2340.5683859834367 4
[1.70242228 7.19885346] 1850.0196128921186 5
[1.84878119 7.15584734] 1397.8767751913624 6
[1.98690585 7.11846879] 1006.2913089234257 7
[2.11280377 7.08676205] 689.1580065220413 8
[2.22352266 7.06053382] 449.4461006193417 9
[2.31752117 7.03936902] 280.11328644463794 10
[2.39472346 7.02268437] 167.83550939801145 11
[2.4562713  7.00980453] 97.48295381345957 12
[2.50409841 7.00003917] 55.48376317925672 13
[2.54048431 6.99274387] 31.391064288526557 14
[2.5677006  6.98735704] 18.002622743914515 15
[2.58779121 6.98341433] 10.743694425808126 16
[2.60247346 6.98054682] 6.880965373260716 17
[2.61312278 6.97847003] 4.853946405887816 18
[2.62080394 6.97696944] 3.8011109325048205 19
[2.62632146 6.97588588] 3.2583464174069444 20
[2.63027278 6.97510266] 2.98004740802213 21
[2.63309604 6.97453498] 2.8379039299412447 22
[2.63510979 6.97412158] 2.765502005613649 23
[2.6365441 6.9738184] 2.728693000326808 24
[2.63756445 6.97359387] 2.7100016739565103 25
```

Specify change(s).
Show output.
Justify why the change(s) improve the results.

Mid Term Exam Discussion

2.2 Trust Region Method

Let make sure that the following parameter settings are set in the code:

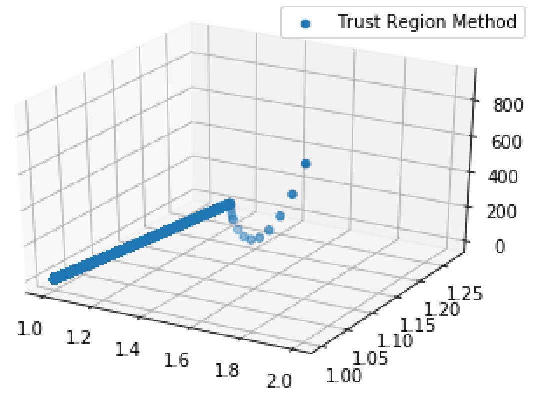
- Selected step finder algorithm is Cauchy Point.

Basically this is the settings given in the provided codes…

f) x1s variable in the codes are utilized to keep 3 set of starting point. Run the code using these 3 sets of starting points. Then, write down the values of x_0, x_1, $f(x)$ and number of iterations for each of them. (1.5 marks)

```
x, fx, iterations, vals, objectfs = tr.trust_region(f1, g1,
h1, x1s[0], .1, 1, .25, sf.cauchy_point_step_finder)
```

```
Result in 16531 iterations:
[1.00000544 1.00001091] -> 0.000000
```

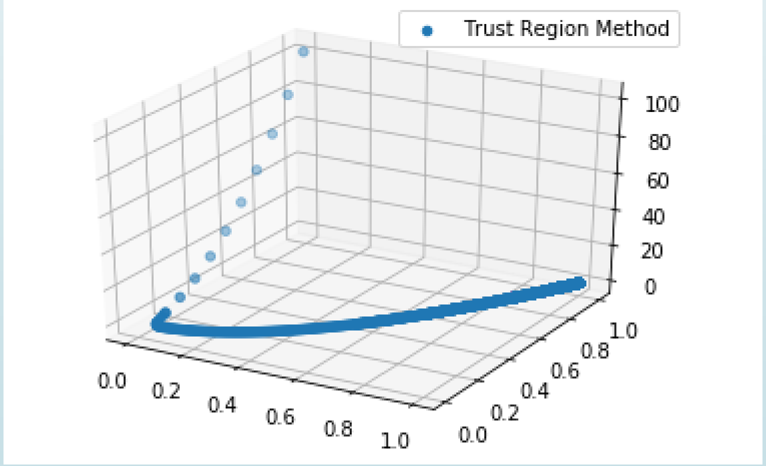


Mid Term Exam Discussion

f) x1s variable in the codes are utilized to keep 3 set of starting point. Run the code using these 3 sets of starting points. Then, write down the values of x_0, x_1, $f(x)$ and number of iterations for each of them. (1.5 marks)

```
x, fx, iterations, vals, objectfs = tr.trust_region(f1, g1,
h1, x1s[1], .1, 1, .25, sf.cauchy_point_step_finder)
```

```
Result in 16073 iterations:
[0.9999944  0.99998878] -> 0.000000
```

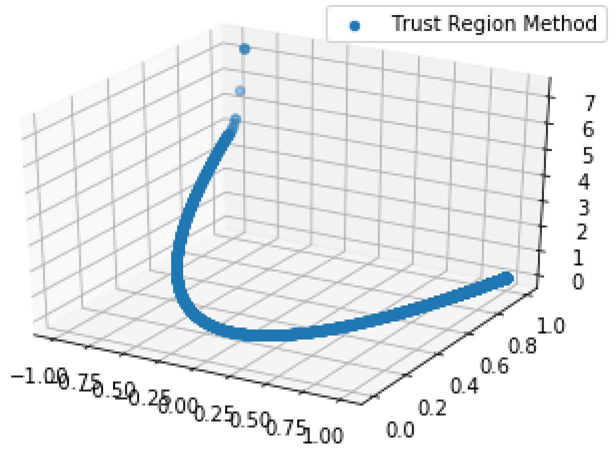


Mid Term Exam Discussion

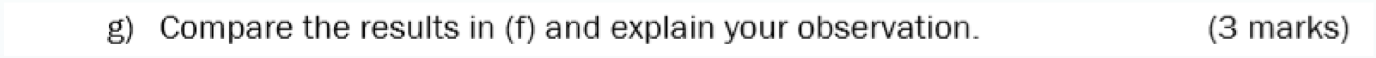
f) x1s variable in the codes are utilized to keep 3 set of starting point. Run the code using these 3 sets of starting points. Then, write down the values of x_0, x_1, $f(x)$ and number of iterations for each of them. (1.5 marks)

```
x, fx, iterations, vals, objectfs = tr.trust_region(f1, g1,
h1, x1s[2], .1, 1, .25, sf.cauchy_point_step_finder)
```

```
Result in 17679 iterations:
[0.99999541 0.99999081] -> 0.000000
```



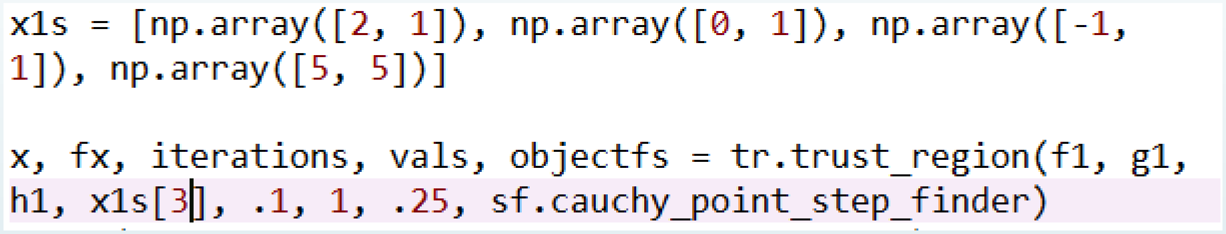
Mid Term Exam Discussion


g) Compare the results in (f) and explain your observation. (3 marks)

Compare no. of iteration, optimum results, etc. Explanation should be centered around changes of starting points.

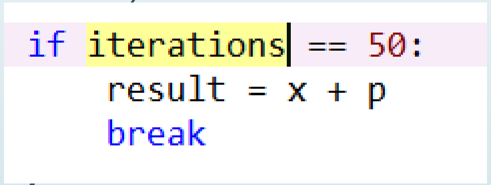
Mid Term Exam Discussion

h) Modify the code to add in another stopping condition. Set maximum number of iterations to 100. Set the starting point to [5, 5], run the code. Then, write down the values of x_0, x_1, $f(x)$ and number of iterations for each of them. (2 mark)



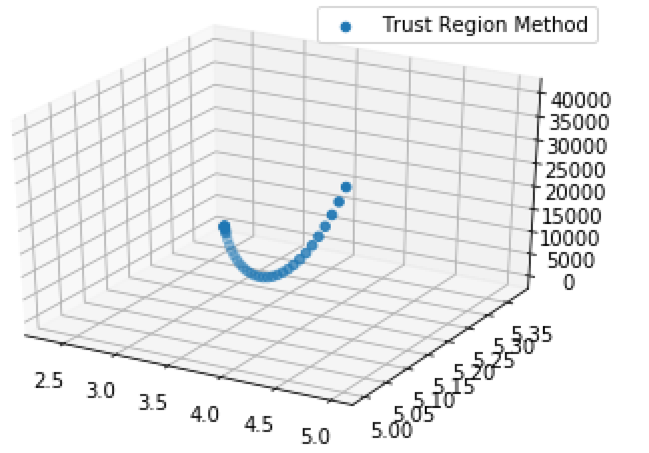
```
x1s = [np.array([2, 1]), np.array([0, 1]), np.array([-1,
1]), np.array([5, 5])]

x, fx, iterations, vals, objectfs = tr.trust_region(f1, g1,
h1, x1s[3], .1, 1, .25, sf.cauchy_point_step_finder)
```



```
if iterations == 50:
    result = x + p
    break
```

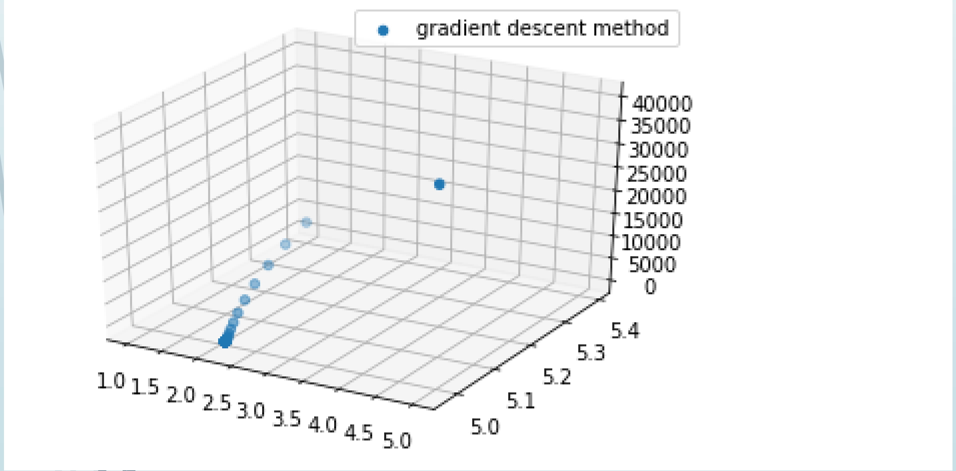
```
Result in 50 iterations:
[2.31839538 5.37770384] -> 1.738921
```



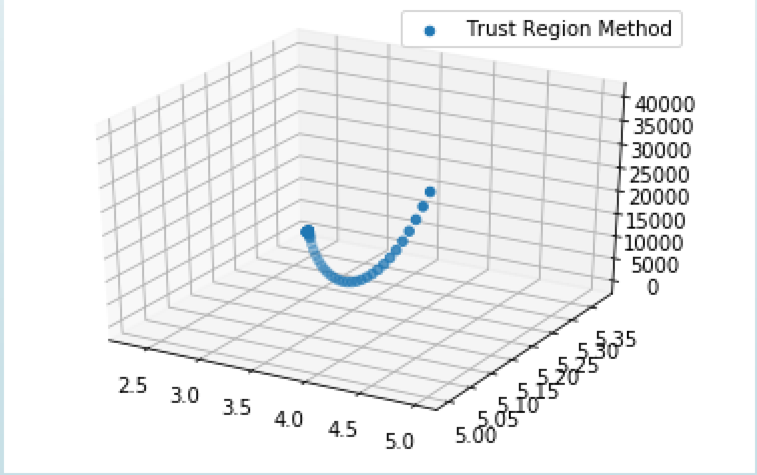
Mid Term Exam Discussion

i) Go back to your Gradient Descent method. If both methods are set to same initial point (i.e., [5, 5]) and same number of iterations (i.e., 100, note that you have to change the difference of $f(x_k)$ and $f(x_{k+1})$ to smaller value to enable 100 iterations), record the results (i.e., x_0, x_1, $f(x)$) for both methods. (2 marks)

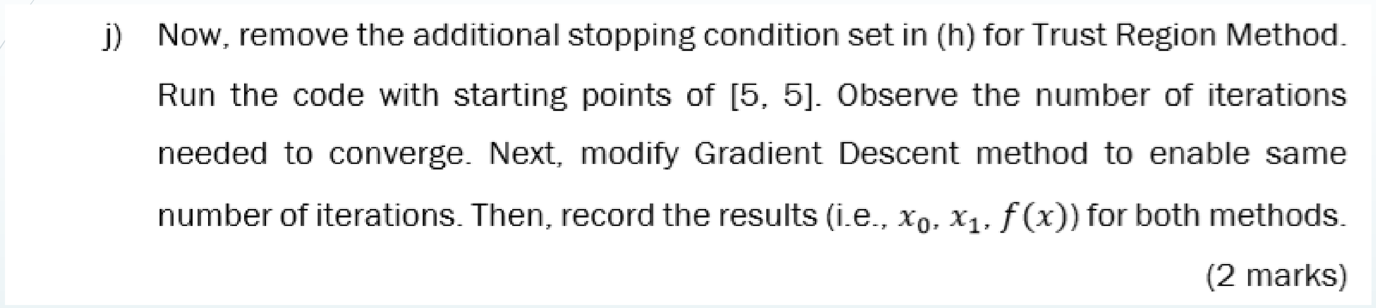
```
[2.22885033 4.97039843] 1.5107620035127982 100
```



```
Result in 100 iterations:
[2.31737306 5.37293387] -> 1.736209
```



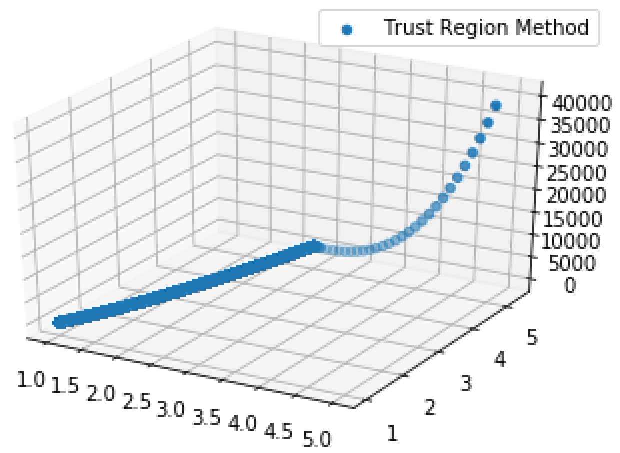
Mid Term Exam Discussion



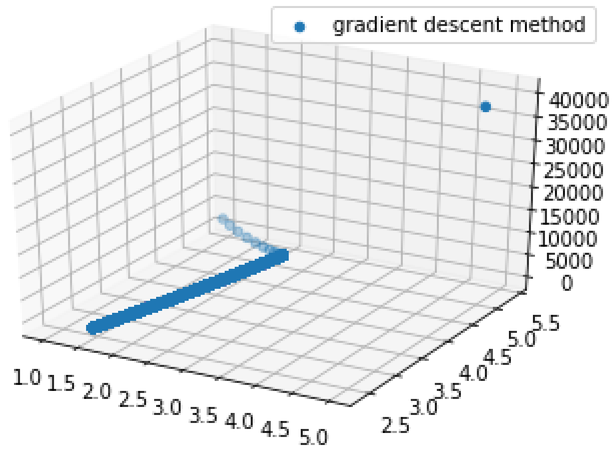
j) Now, remove the additional stopping condition set in (h) for Trust Region Method. Run the code with starting points of [5, 5]. Observe the number of iterations needed to converge. Next, modify Gradient Descent method to enable same number of iterations. Then, record the results (i.e., x_0, x_1, $f(x)$) for both methods.

(2 marks)

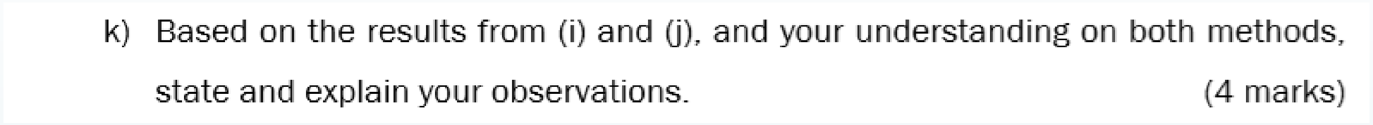
```
Result in 60334 iterations:
[1.0000054  1.00001083] -> 0.000000
```



```
[1.54086995 2.37586819] 0.29279247367505572 60334
```



Mid Term Exam Discussion



k) Based on the results from (i) and (j), and your understanding on both methods, state and explain your observations. (4 marks)

Compare no. of iteration, optimum results, graph curve / directions, etc.
Explanation should be on the comparison of both methods.

Assignment II

- Same group as Assignment I
- **Tasks**:
  - Identify ONE real-life problem (constrained) in your industry area and find a relevant dataset.
  - Discuss with your lecturer – must be agreed by lecturer.
  - Model the objective function and constraints.
  - Solve it using programming (justification needed for the solution chosen and result observations).
  - Submission:
    - Simple report
    - Code
    - Presentation

Assignment II

- Note: Plagiarism or direct usage of solution from the internet are strictly prohibited!
- References:
  - [1] https://archive.ics.uci.edu/ml/datasets.html
  - [2] http://stanford.edu/class/ee103/portfolio.html
  - [3] https://www.kaggle.com/datasets
  - [4] https://registry.opendata.aws/
  - [5] and others!!

Timeline

- Week 9 – Bye bye to unconstrained optimization and hello to constrained optimization
- Week 10 – Table Solution + implementation in Python + discussion on problem (Assignment II)
- Week 11 – Quiz 2 + discussion on problem (Assignment II)
- Week 12 – Quadratic Programming I
- Week 13 – Presentation
- Week 14 - Quadratic Programming II + Revision

Recall of the topic today…

Exercise

- Solve the following linear program:

$$\text{maximise } 5x_1 + 6x_2$$

- subject to

$$x_1 + x_2 <= 10$$

$$x_1 - x_2 >= 3$$

$$5x_1 + 4x_2 <= 35$$

$$x_1 >= 0$$

$$x_2 >= 0$$

- In graphical form.