ROB 422 : Introduction to Algorithmic Robotics

Homework Assignments -2

Name: Hemanth Murali mail ID: hems@umich.edu

UMID: 83941197

Q1.) Is a Single point Convex? Use definition of Convexity

Ans: Yes, a Single point is Convex. As per the definition of Convexity:

↳ Any two points $x, y \in C$; Contains line segment between any two points in the set. C is a convex set if:

$$\Rightarrow x, y \in C \; ; \; 0 \leq \theta \leq 1 \Rightarrow \theta x + (1-\theta) y \in C$$

Now,

for Single point set $C = \{a\}$. So the only point in C are $x = a$ and $y = a$. Thus the line segment between x and y is just point 'a' itself. $x, y \in C \; ; \; C = \{a\}$

For any $\theta \in [0, 1]$;

$$\theta a + (1-\theta) a = a \quad \Rightarrow \quad a = a$$

Since $a \in C$, the line segment is entirely contained within C.

$\therefore$ Single point set is Convex

Q2.) Is $f(x) = (|2-5x| + 2x + 8e^{(-4x)}) - 1$ convex? Using principles of composition and the common convex functions.

Ans: Given,

$$f(x) = (|2-5x| + 2x + 8e^{(-4x)}) - 1$$

Now,

In order to use the principles of composition and properties of convex functions. let's rewrite as each component:

$\longrightarrow f(x) = f_1(x) + f_2(x) + f_3(x)$ and we have to prove that $f_1(x)$, $f_2(x)$, $f_3(x)$ as convex

Thus,

$f_1(x) = |2-5x|$

$|2-5x|$ is a Convex function, because it is an absolute value of a linear function.

$f_2(x) = 2x$

$2x$ is an affine / linear function. So it is both Concave & Convex

$f_3(x) = 8e^{(-4x)}$

$8e^{(-4x)}$ is an exponential function of the form e^{ax} (where $a = -4 < 0$), thus it is Convex.

Since $f_1(x)$, $f_2(x)$ & $f_3(x)$ are Convex, their sum ($f(x) = f_1(x) + f_2(x) + f_3(x)$) is also Convex.

$\therefore |2-5x| + 2x + 8e^{-4x}$ is Convex.

Now,

Subtracting 1 from the sum does not affect its Convexity.

Thus, $f(x) = (|2-5x| + 2x + 8e^{-4x})$ is Convex

Q3.) Rewrite the optimization problem in Standard form ($x = [x_1, x_2]^T$):

Maximize x :

$$-4x_2 + 3x_1 - 3$$

Subject to :

$$x_2 \leq -3,$$

$$-x_1 - 2 \geq x_1 - 5x_2,$$

$$x_2 + 6.3 = x_1,$$

$$-x_2 + 5 + 4x_1 \leq 4x$$

Ans:

Maximize ; $-4x_2 + 3x_1 - 3$

Let's convert each constraint to Standard form:

→ Constraint 1 : $x_2 \leq -3$

$\Rightarrow x_2 + 3 \leq 0 \Rightarrow f_1(x) = x_2 + \leq 0$

inequality constraints

→ Constraint 2: $-x_1 - 2 \geq x_1 - 5x_2$

$\Rightarrow -x_1 - 2 \geq x_1 - 5x_2$

$\Rightarrow -2x_1 + 5x_2 - 2 \geq 0$ $\quad (\times -1)$

$\Rightarrow 2x_1 - 5x_2 + 2 \leq 0$

$\Rightarrow f_2(x) = 2x_1 - 5x_2 + 2 \leq 0$

[inequality constraint]

→ Constraint 3: $x_2 + 6.3 = x_1$

$\Rightarrow x_1 - x_2 - 6.3 = 0$

$\Rightarrow h_1(x) = x_1 - x_2 - 6.3 = 0$

[equality constraints]

→ Constraint 4: $-x_2 + 5 + 4x_1 \leq 4x_1$

$\Rightarrow -x_2 + 5 + 4x_1 \leq 4x_1$

$\Rightarrow -x_2 + 5 \leq 0$

$\Rightarrow f_3(x) = 5 - x_2 \leq 0$

[inequality Constraint]

Q4.) $f(x) = \max\{3x^2 - 2, 2x - 1\}$. At what value(s) of x will the subdifferential $\partial f(x)$ contain more than one subgradient? What is $\partial f(x)$ at each such x value?

Ans:

$$f(x) = \max\{3x^2 - 2, \; 2x - 1\}$$

$f(x)$ will have more than one subgradient at the point where the graph of

$f(x) = 3x^2 - 2$

$f_1(x) = 2x - 1$ will intersect

Thus,

$$3x^2 - 2 = 2x - 1$$

$$\Rightarrow 3x^2 - 2x - 1 = 0$$

$$\Rightarrow x^2 - 2/3\,x - 1/3 = 0$$

$$\Rightarrow x = -1/3 \quad \text{and} \quad x = 1$$

$\partial f(x)$ at each x :

$$f'(x) = \frac{d}{dx} f(x) \Rightarrow 6x$$

$$\Rightarrow f'(-1/3) = -2$$

$$\Rightarrow f'(1) = 6$$

$$f_1'(x) = \frac{d}{dx}\left[f_1(x)\right] = 2.$$

$$\Rightarrow f_1'(-1/3) = 2$$

$$\Rightarrow f_1'(1) = 2$$

Therefore,

Values $x = -1/3$ and 1 at which the Subdifferential $\partial f(x)$ contains more than one Subgradient.

Also,

$$\partial f(-1/3) = [-2, 2] \text{ and,}$$

$$\partial f(1) = [2, 6]$$

Q5) $\underset{x}{\text{Minimize}} \quad c^T x$

Subject to

$$Gx \leq h,$$
$$Ax = b.$$

a.) Write down the Lagrange dual function & define the variables.

Ans:

$$f(x) = c^T x$$

Converting to standard form. → primal optimization

$$f_i(x) = Gx - h \leq 0 \quad \text{[inequality constraint]}$$

$$h(x) = Ax - b = 0 \quad \text{[equality constraint]}$$

Using Lagrangian Function:

$$L(x, \lambda, \nu) = f_0(x) + \sum_{i=1}^{m} \lambda_i f_i(x) + \sum_{i=1}^{p} \nu_i h_i(x)$$

λ_i → Lagrangian multiplier with $f_i(x) \leq 0$

ν_i → Lagrangian multiplier with $h_i(x) = 0$

Thus,

$$L(x, \lambda, \nu) = c^T x + \lambda(Gx - h) + \nu(Ax - b)$$

b.) Write down the dual problem for this LP.

Ans:

Lagrangian dual function:

$$g(\lambda, \nu) = \inf_{x \in D} L(x, \lambda, \nu)$$

$$g(\lambda, \nu) = \inf_{x \in D} \left(c^T x + \lambda (Gx - h) + \nu (Ax - b) \right)$$

$$= \inf_{x \in D} \left(c^T x + \lambda G x - \lambda h + A\nu x - b\nu \right)$$

$$= \inf_{x \in D} \left[(c^T + \lambda G + A\nu) x - (\lambda h - b\nu) \right]$$

Hence,

$g(\lambda, \nu)$ to be bounded from below in terms of x,

which implies;

$$\left[c^T + \lambda G + A\nu \right] = 0$$

$$g(\lambda, \nu) = \inf_{x \in D} \left[-\lambda h \quad -b\nu \right]$$

maximize: $(-\lambda h \quad -\nu b)$

Subject to:

$$\left[\begin{array}{l} c^T + \lambda h + A\nu = 0 \\ \text{and,} \\ \lambda \geq 0 \end{array} \right]$$

c) How does d^* relate to the solution of the primal problem p^* ?

Ans:

Let the primal problem be feasible and bounded,

→ To solve the dual problem, let d^* be the optimal value

→ To solve the primal problem, let p^* be the optimal value

Thus the relationship between d^* and p^* can be,

Case 1 : $p^* \geq d^*$

This shows that the optimal solution of the primal problem is always greater than or equal to the optimal solution of the dual problem.

This case is called **Weak duality**.

Case 2: $p^* = d^*$

This shows that the optimal solution of the primal problem is equal to the optimal solution of the dual problem.

This case is called **Strong duality**.

Software Implementation:

Q 1.) Descent Methods:

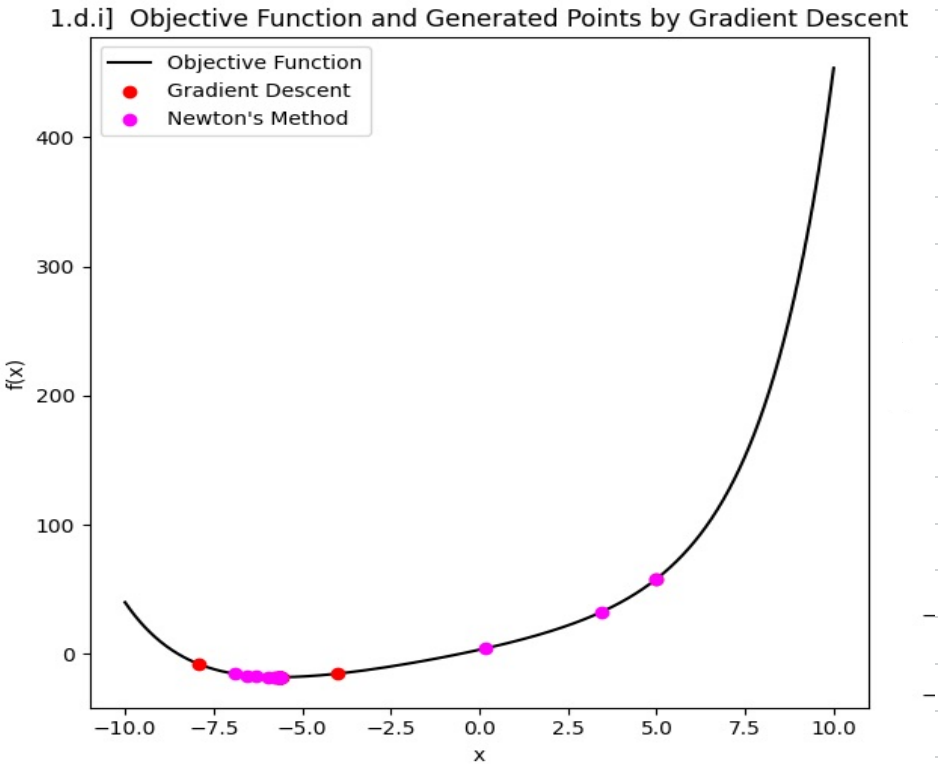
a.) Backtracking implemented in "backtracking.py" file and submitted as Python code.

b.) Gradient Descent algorithm implemented in "gradientdescent.py" file and submitted as Python code

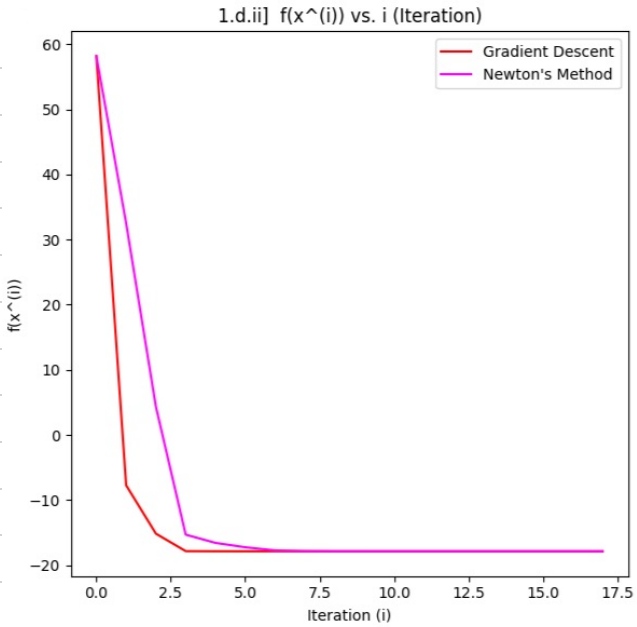
c.) Succesfully implemented Newton's method algorithm in "newtonsmethod.py" and submitted.

d.) i)

```
Gradient Descent Method value:  -5.637388220655002
Newtons Method value:  -5.637374007550397
```



d.) ii)

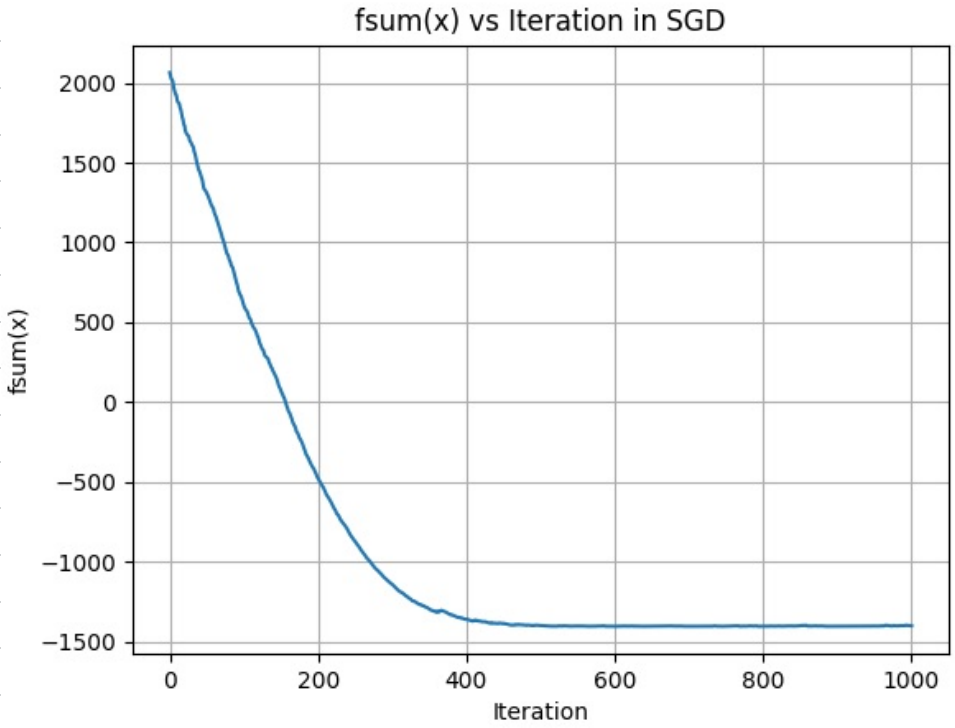


Newton's Method algorithm performs better as it converges faster. Because it uses Hessian, which makes more accurate steps.

Q.2) Stochastic Gradient Descent:

a.) Implemented in "SDGtest.py"

b.) In the graph, it's clearly shown that the graph is decreasing & there are some small hikes. This is due to some Random Noise in gradient estimates which leads to occasional increases in fsum(xCi))



c.)

```
Stochastic Gradient Descent with 1000 iterations - Mean: -1403.611468562707, Variance: 6.316517132889191
Stochastic Gradient Descent with 750 iterations  - Mean: -1403.1701131025695, Variance: 5.864419929508063
```

Mean:

Both iterations resulted in negative mean value, indicating that the algorithm was effective in minimizing the Objective function.

As the values are nearby, increasing in the no. of iterations did not improve the mean.

Variance :

Variance increased, this indicates that the estimates are stable.

d) i)

```
Time taken by Stochastic Gradient Descent:  0.00200295448303222 66
Time taken by Gradient Descent:  17.587599515914917
Time taken by Newton's Method:  0.18494939804077148
```

SGD: In each iteration, SGD only computes the gradient using a single or a small subset
of data points and not the whole thing

GD: Gradient Descent computes the gradient using all data points, making each iteration
take more time .

NM: Newton's Method requires computing the Hessian (second-order derivative matrix)
and solving a linear system in each iteration, making each step computationally much heavier, though it may converge in fewer iterations.

```
Stochastic Gradient Descent: fsum found:  -1404.4384959450545
Gradient Descent: fsum found:  -1405.267040990034
Newton's Method fsum found:  -1405.26704099006
```

The SGD, GD and NM gives more or less the same answer. Its just the efficiency of the 3 algorithms that differ. so al 3 converge to the same minimum point but varies

3) a) Centering problem :

optimal Solution : $f \to 0 \to x^*$

$$\text{Min } f(x) = f_0(x) - \sum_{i=1}^{m} \log(-f_i(x))$$

$t f_0(x)$: for force field $f_0(x) = -t \nabla f_0(x)$

$-\log(-f_i(x)) \to$ potential of force field.

$$f_i(x) = \left(\tfrac{1}{f_i(x)}\right) \nabla f_i(x)$$

here $f_0(x) \to$ objective function.

(b) In linear program, we have

$$\min_x \; c^T x$$

here

$c \in R^n$:

$A \in R^{m \times n}$:

$b \in R^m$

$f_0(x) = c^T x$

$f_i(x) = Ax - b \leq 0$

$$\phi(x) = +c^T x - \sum_{i=1}^{m} \log(-a_i x - b_i)$$

$$= +c^T x - \sum_{i=1}^{m} \log(-a_i x + b_i$$

c) derivative..

$$\phi(x) = +c^T x - \sum_{i=1}^{m} \log(-a_i^T x + b)$$

$$\nabla \phi(x) = c + + \sum_{i=1}^{m} \frac{a_i}{b_i - a_i^T x}$$

d) $$\nabla^2 \phi(x) = \sum_{i=1}^{m} \frac{a_i}{b_i - a_i^T x}$$

$$\frac{a_i^T}{(b_i - a_i^T x)}$$

$$= \sum_{i=1}^{m} \frac{a_i}{b_i - a_i^T x)^2}$$

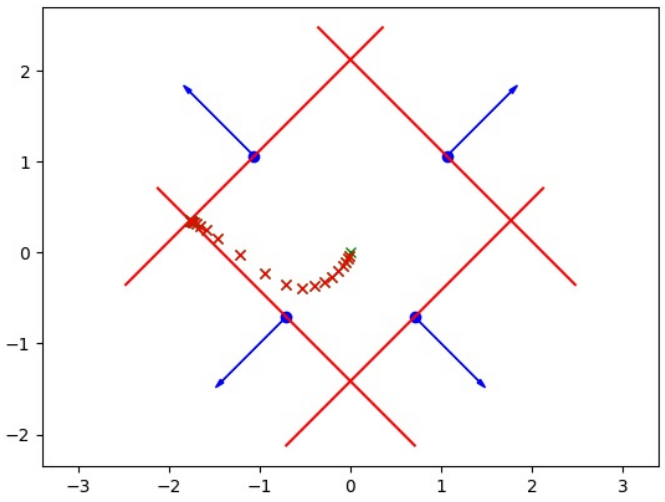
e) duality gap

numplanes = no. of hyperplanes

t = barrier parameter.

duality gap = $\frac{\text{numplanes}}{t}$.

f)




```
OUTER loop iteration 26: Number of INNER loop iterations: 1
OUTER loop iteration 27: Number of INNER loop iterations: 1
OUTER loop iteration 28: Number of INNER loop iterations: 1
The optimal point: (-1.767269, 0.353299)
Total number of outer loop iterations: 28
```

g) Solution x^* is

$(-1.767269, 0.3532)$

4a)

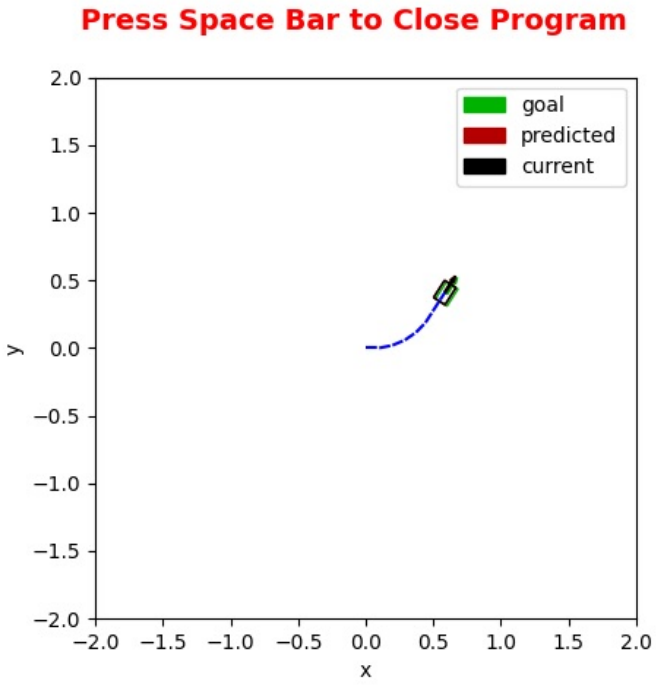
```
Testing linearization of dynamics for
  Current state:
    [0.2 0.1 0.4]
  Reference control:
    [0. 0.]
  h:
    0.01
[0.2 0.1 0.4 0.  0. ]
A:
 [[1. 0. 0.]
 [0. 1. 0.]
 [0. 0. 1.]]
B:
 [[0.0921061  0.        ]
 [0.03894183 0.        ]
 [0.         0.1       ]]

Test control:
    [0.1 0.2]

Predicted state using linearized dynamics Ax + Bu:
   [0.20921061 0.10389418 0.42      ]
True state (using true dynamics):
   [0.20921061 0.10389418 0.42      ]
Prediction error:
   6.206335383118183e-17
```

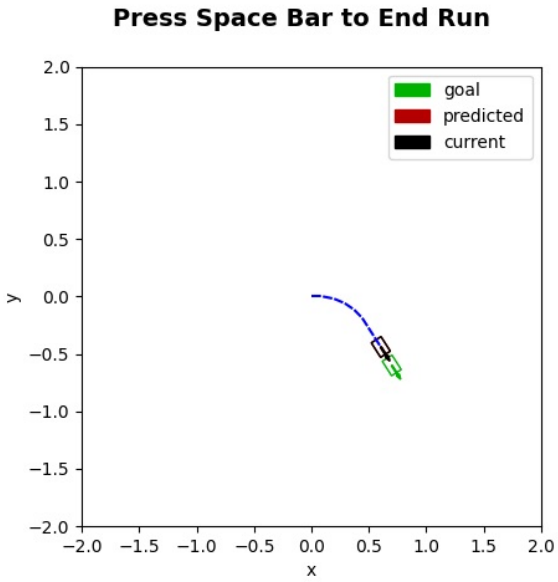
4.) b.)

Test 0:



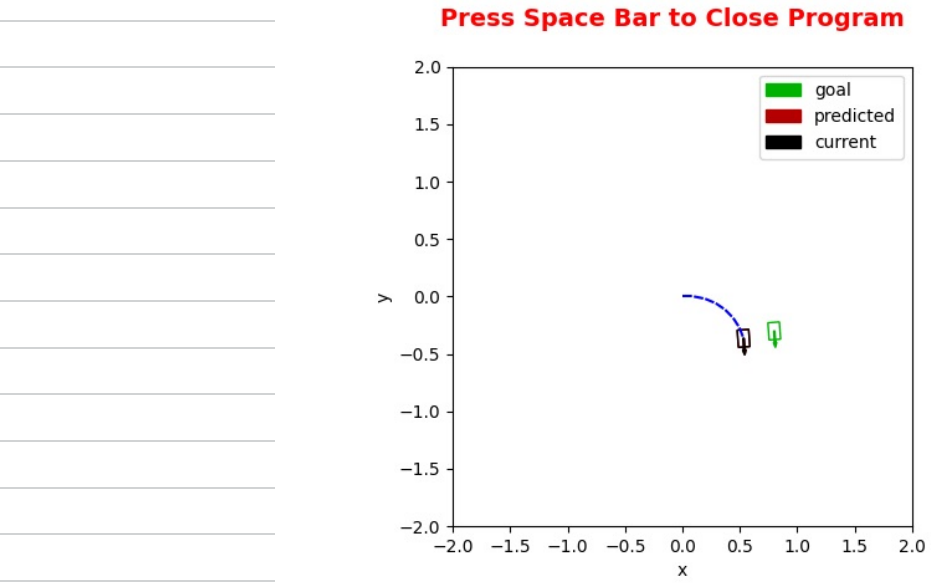
Distance to goal: 0.0176059...

Test 1:



Dist. to goal: 0.0068307...

Test 2:



Dist. to goal: 0.2750527...