EECS 498 HW2

Questions

Prob. 1

Define a point f in n-dimension space: f $\coloneqq X \to y, where\ y\ is\ constant\ R\ and\ X\ is\ a\ constant\ R^n$

$Let\ x_1, x_2 \in X\ \ (x_1 = x_2)$

$f(x_1) = f(x_2) = y, \forall t \in [0,1],$

$\Rightarrow f(tx_1 + (1-t)x_2) = f(x_1) = f(x_2) = y, \qquad tf(x_1) + (1-t)f(x_2) = ty + (1-t)y = y$

$\Rightarrow f(tx_1 + (1-t)x_2) = y\ \leq\ \ tf(x_1) + (1-t)f(x_2) = y$

$\therefore\ a\ single\ point\ is\ convex$

Prob. 2

$Let\ f_1(x) = |2 - 5x|, f_2(x) = 2x, f_3(x) = 8e^{(-4x)}, f_4(x) = -1$

$f_1 = |2 - 5x|$ is convex since it is a combination of hyperplane

$f_2 = 2x$ is convex since it is a hyperplane

$f_3 = 8e^{(-4x)}$ is convex since $f_3'' = 128e^{(-4x)} > 0$

$f_4 = -1$ is convex since it is a hyperplane

Since $f_1,\ f_2,\ f_3,\ f_4$ are convex and operation of sum of convex functions preserves convexity

$\Rightarrow f(x)$ is convex

Prob. 3

$$f'(x) = 0.5e^{(0.5x+2)} - 0.5e^{(-0.5x-0.5)} + 4 \Rightarrow f'(5) \approx 48.9837$$

By Symmetric Difference Quotient

$$h = 0.001, f'_{h=0.001}(5) - f'^{(5)} = -1.874 \times 10^{-6}$$

$$h = 10^{-4}, f'_{h=10^{-4}}(5) - f'(5) = -1.856 \times 10^{-8}$$

$$h = 10^{-5}, f'_{h=10^{-5}}(5) - f'(5) = 1.688 \times 10^{-9}$$

$$h = 10^{-6}, f'_{h=10^{-6}}(5) - f'^{(5)} = -5.658 \times 10^{-8}$$

Numerical results with $h = 10^{-3}, 10^{-4}, 10^{-5}$ are smaller than the analytical result, except the result with $h = 10^{-5}$ and it is also the closest result to the analytical one.

Prob. 4

$$\min_{x} \ [-3 \quad 2]\, x - 1$$

$$f \coloneqq \begin{bmatrix} 0 & -1 \\ 0 & -5 \\ -5 & 4 \end{bmatrix} x \ \le \begin{bmatrix} 3 \\ -2 \\ -2 \end{bmatrix}$$

$$g \coloneqq [1 \quad 1]\, x \ = \ 2.3$$

Prob. 5

There would be more than one subgradient when $3x^2 - 2, 2x - 1$ intersects

$\Rightarrow (3x^2 - 2) - (2x - 1) = 0 \Rightarrow x = 1/3 \; or \; 1$

$$\delta f(x) = \begin{cases} 6x, x > 1 \; or \; x < 1/3 \\ 2, \quad\quad 1/3 < x < \ 1 \\ \quad\quad [2,6], x = 1 \\ \quad\quad 2, \quad x = 1/3 \end{cases}$$

Prob. 6

(a) Lagrange dual function

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

Dual function:

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

$\lambda \; is \; a \; vector \; of \; Lagrange \; mutipliers$

ν is a vector of Lagrange multipliers

(b) Dual problem

$$\text{Maximize } -h^T \lambda - b^T \, \nu$$

$$\text{Subject to } G^T \lambda + A^T \nu + c = 0$$

$$\lambda \succeq 0$$

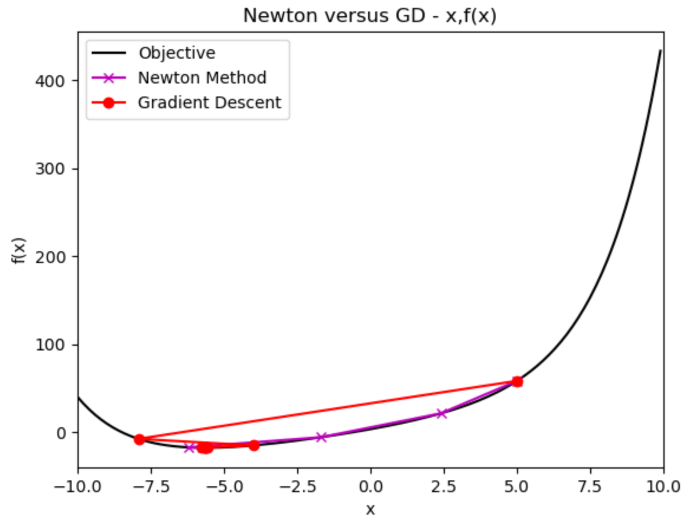
(c) Optimal value of dual problem relation with primal optimal?

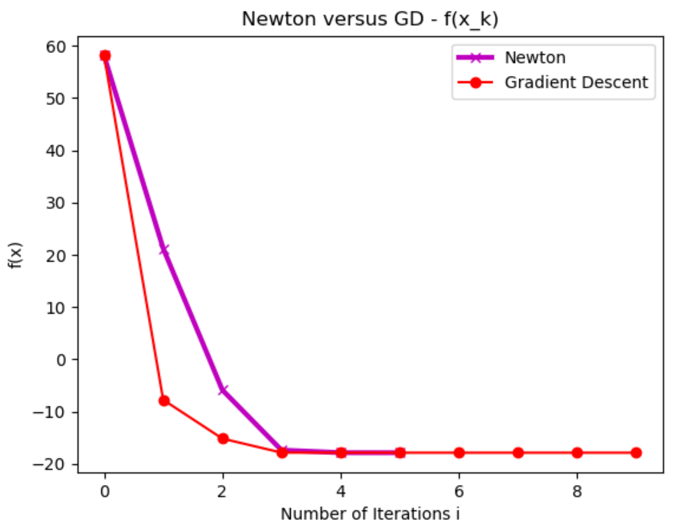
Since we assume primal is feasible and bounded, then the strong duality holds.

$\Rightarrow p^* = d^*$

Implementation

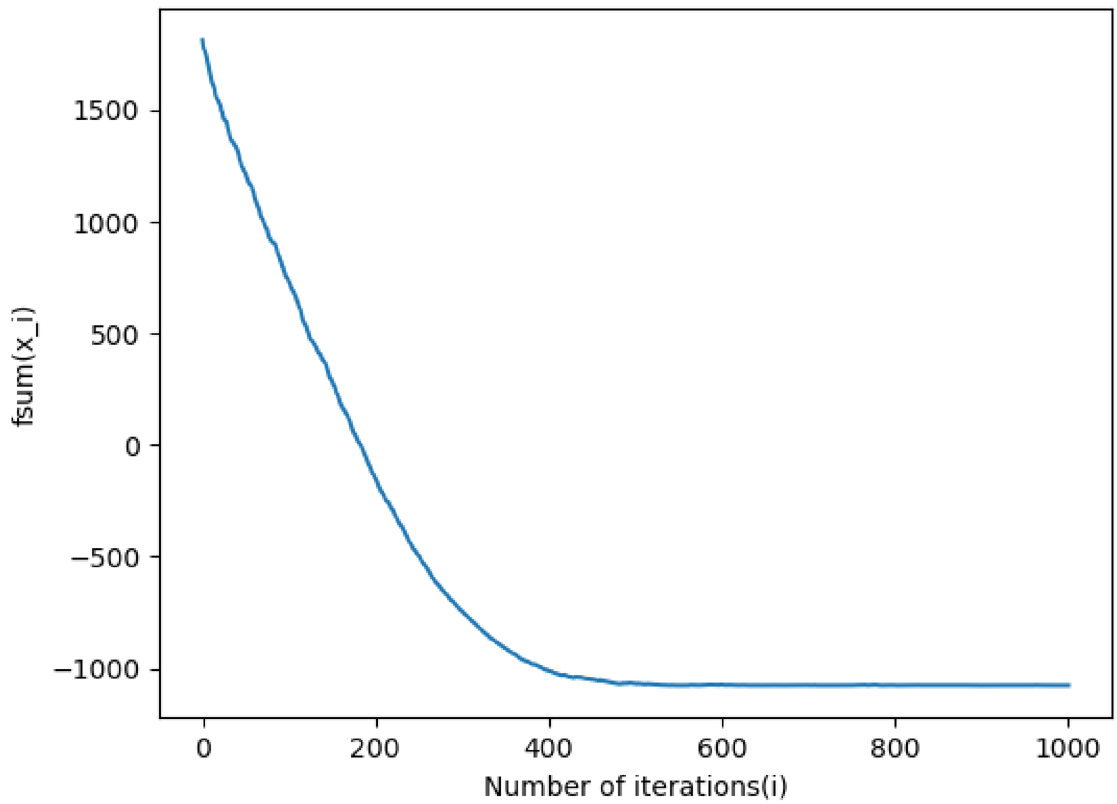
Prob 1d





In terms of total iterations, Newton method is better than Gradient Descent in this case. Since convergence rate of Gradient Descent is greatly depending on the condition number of Hessian, and we can observe that the condition number of the hessian around optimal is high(not sensitive), Gradient Descent would take more iterations to reach termination condition. In comparison, Newton method doesn't have this problem by utilizing second order approximation.

Prob 2b



$fsum(x^{(i)})$ is not always descending. SGD takes a random term of derivative fi to update x every iteration, which means it doesn't guarantee that the direction of $fsum(x^{(i)})$ is always toward the descending direction.

Prob 2c

```
PS D:\class\Introduction_to_Algorithmic Robotics\EECS-498-HW2> python.exe .\SGD_Comparison.py
sgd complete
750  iterations and 30 times, mean = 6.40794, var = 0.01466
1000 iterations and 30 times, mean = 6.37860, var = 0.02162
PS D:\class\Introduction_to_Algorithmic Robotics\EECS-498-HW2>
```

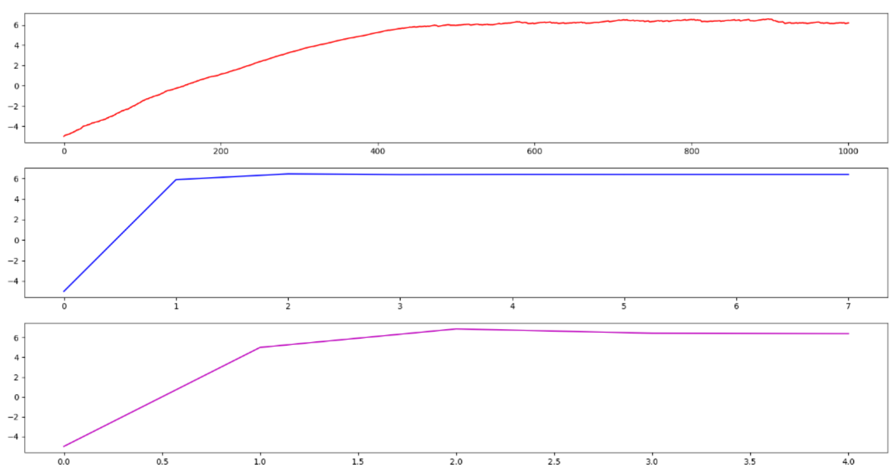
The result from 1000 iterations is more various and far away from optimal than the result of 750 iterations. The phenomenon results from the stochastic sense of SGD, which doesn't guarantee that the more iterations, the closer to the optimal and less various.

Prob 2d

```
PS D:\class\Introduction_to_Algorithmic Robotics\EECS-498-HW2> python.exe .\descentmethod_compare.py
SGD    time: 0.005873
fsum        = -1073.380458
GD     time: 8.407393
fsum        = -1075.782940
Newton time: 1.116890
fsum        = -1075.782928
```

i. SGD only computes one derivative of fi each iteration, which means the computation cost of SGD with the setting is proportional to the number of iterations (1000).

GD and Newton method, on the other hand, need to compute the derivative of current x (and even Hessian of x in terms of Newton method). In this case, there are 10000 of fi. That makes SGD significantly faster than 2 other methods.



From Prob 1, we know that GD may suffer from the condition number of f and takes more iterations. In the experiment, GD takes 7 iterations and Newton method takes 4, which makes Newton method faster than Gradient Descent.

ii. In terms of $fsum(x^*)$, performance of GD and Newton method are about the same, and SGD is the worst. As mention in Prob. 2b, since SGD randomly pick a term to update x, direction of $fsum(x^{(i)})$ doesn't always toward the descent direction by SGD. Update by SGD may bouncing around the optimal.

Prob 3a

$$tf_0(x) - \sum_{i=1}^{m} log\big(-f_i(x)\big)$$

t: scale for objective force
f_0 : objective function from the primal problem
f_i: inequality constraints

Prob 3b

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

c: coefficients of objective functions
a_i : coefficients of inequality constraints
b_i: constants values of inequality constraints

Prob 3c

$$tc \; - \sum_{i=1}^{m} \frac{-a_i^T}{\left(-a_i^t x + b_i\right)}$$

Prob 3d

$$\sum_{i=1}^{m} \frac{a_i a_i^T}{\left(-a_i^t x + b_i\right)^2}$$

Prob 3e

$$numplanes\ /\ t$$

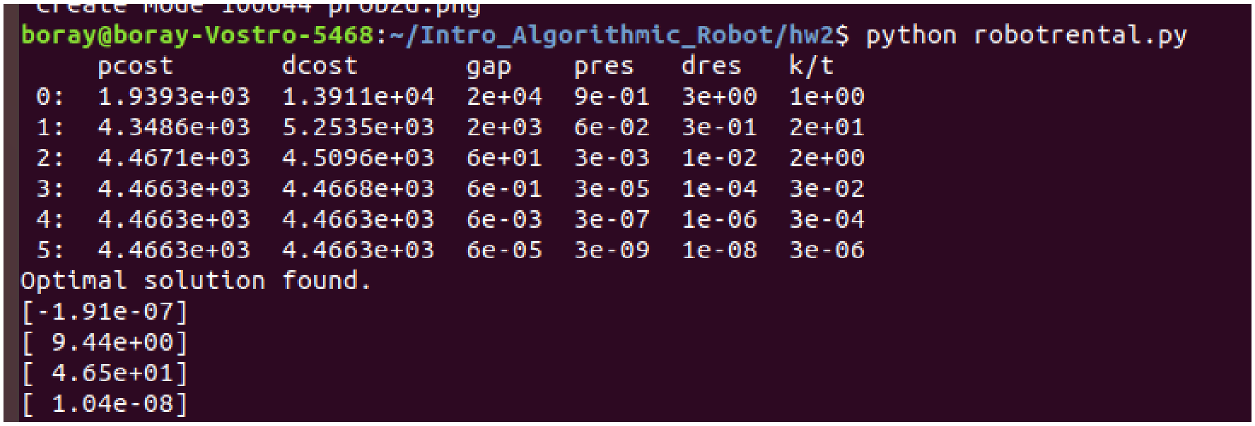
Prob 4a

Minimize $c^T x$

Subject to $Ax \;\geq b, x \;\geq 0$

$$x \;=\; \begin{bmatrix} rent\ hour\ of\ Spider \\ rent\ hour\ of\ Gigantimus \\ rent\ hour\ of\ VersaDroid \\ rent\ hour\ of\ Hedonism \end{bmatrix}, c \;=\; \begin{bmatrix} 75 \\ 128 \\ 70 \\ 34 \end{bmatrix}, A = \begin{bmatrix} 1.6 & 7.2 & 3.7 & 0.1 \\ 3.5 & 2.1 & 3.2 & 0.15 \\ 0.1 & 7.1 & 2.9 & 0.1 \\ 2.3 & 3.2 & 3.4 & 0.15 \\ 6.1 & 0.1 & 4.9 & 0.1 \end{bmatrix}, b = \begin{bmatrix} 51 \\ 48 \\ 202 \\ 120 \\ 229 \end{bmatrix}$$

Prob 4b



```
boray@boray-Vostro-5468:~/Intro_Algorithmic_Robot/hw2$ python robotrental.py
     pcost       dcost       gap    pres   dres   k/t
 0:  1.9393e+03  1.3911e+04  2e+04  9e-01  3e+00  1e+00
 1:  4.3486e+03  5.2535e+03  2e+03  6e-02  3e-01  2e+01
 2:  4.4671e+03  4.5096e+03  6e+01  3e-03  1e-02  2e+00
 3:  4.4663e+03  4.4668e+03  6e-01  3e-05  1e-04  3e-02
 4:  4.4663e+03  4.4663e+03  6e-03  3e-07  1e-06  3e-04
 5:  4.4663e+03  4.4663e+03  6e-05  3e-09  1e-08  3e-06
Optimal solution found.
[-1.91e-07]
[ 9.44e+00]
[ 4.65e+01]
[ 1.04e-08]
```

Rent SpiderBot P8 for 0hr

Rent Gigantimus Maximus for 9.441hr

Rent VersaDroid X17 for 46.542hr

Rent HedonismBot for 0hr