Optimization Algorithms: Theory and Software Implementation

Prof. Thirumulanathan D

Department of Mathematics

Institute of IIT Kanpur

Lecture: 38

Hello everyone, this is Lecture Three in Week Eight. Recall that in the last two lectures, we were learning about the Augmented Lagrangian Method.

We learned about the differences between the Quadratic Penalty Method and the Augmented Lagrangian Method. We learned how to get the Augmented Lagrangian Method by modifying the Quadratic Penalty Method when you have only equality constraints.

And in the last lecture, we also learned about how to handle inequality constraints. So, that had more differences when you compare it with the Quadratic Penalty Method, right?

I will work out an example which has only inequality constraints. If you recall what we were seeing in the last class, we derived the augmented Lagrangian function L_γ.

L_γ is a function of x, λ, and μ, which is as given at the top of your screen. And if you recall, this is actually for an optimization problem that is given here.

You are minimizing f(x) subject to both a bunch of inequality constraints as well as equality constraints. When you have them, we derived this to be the L_γ that you see here. We derived it to be the augmented Lagrangian function.

We will first derive the gradient and Hessian of this function. Since we are using this algorithm, we will need the gradient and Hessian anyway because we are going to minimize this augmented Lagrangian function in an unconstrained fashion.

If you use Newton's method to minimize this, we need the gradient as well as the Hessian.

We will first write down the expressions for the gradient and the Hessian of the augmented Lagrangian function.

We have the gradient of $L_\gamma(x, \lambda, \mu)$:

$$\nabla L_\gamma(x, \lambda, \mu) = \nabla f(x)+\sum_{j=1}^{m} \lambda_j \nabla h_j(x)+2\gamma \left[\sum_{j=1}^{m} h_j(x) \nabla h_j(x)+ \sum_{i=1}^{p} \max(0, g_i(x) + \mu_i/(2\gamma))\nabla g_i (x)\right]$$

Note that the term $-\mu_i^2/(4\gamma^2)$ does not depend on x. So, this gradient I am taking with respect to x.

This term vanishes when you take the derivative. We just differentiate this, and so here again we will have a 2, which I have pulled out already. We will have summation i equal to 1 to p. We will have this term as such: $\max(0, g_i(x) + \mu_i/(2\gamma))$, and we will have to differentiate this, for which we will have $\nabla g_i(x)$.

It is easy to see that this is actually differentiable, though we have the max component.

That is because if you have $g_i(x) + \mu_i/(2\gamma)$ less than or equal to 0, you will put 0, and if it is greater than 0, it is just $g_i(x) + \mu_i/(2\gamma)$.

When $g_i(x) + \mu_i/(2\gamma)$ is equal to 0, this term becomes 0 irrespective of whether you have this max term or not.

So, no issues; the $\nabla L_\gamma(x, \lambda, \mu)$ is well-defined and it exists when $g_i(x) + \mu_i/(2\gamma) = 0$.

It is the same as how we had it for the quadratic penalty function.

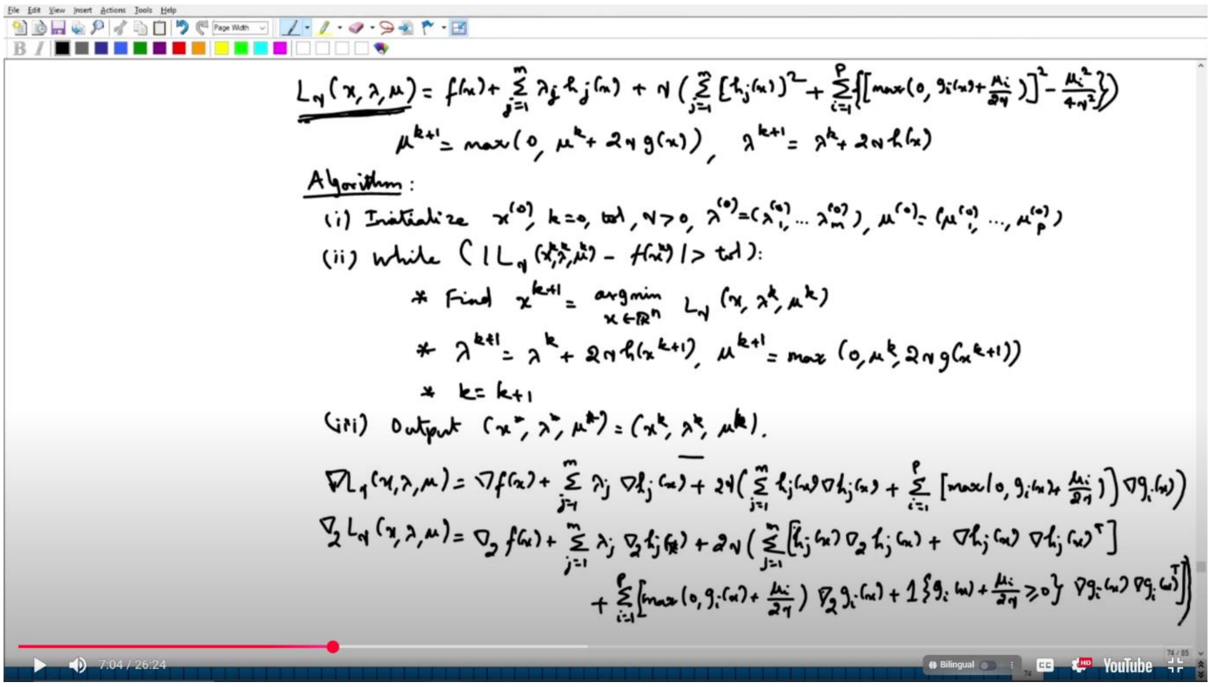
Now, about the Hessian.

Let us write down the expression for the Hessian of the augmented Lagrangian function.

$$\nabla_2 L_\gamma(x, \lambda, \mu)= \nabla_2 f(x) + \sum_{j=1}^{m} \lambda_j \nabla_2 h_j(x) + 2\gamma\Big[\sum_{j=1}^{m} (h_j(x) \nabla_2 h_j(x) + \nabla h_j(x) \nabla h_j(x)^T)$$

$$+\sum_{i=1}^{p} (\max(0, g_i(x) + \mu_i/(2\gamma)) \nabla_2 g_i(x)+1\{ g_i(x) + \mu_i/(2\gamma) >=0\}\nabla g_i(x) \nabla g_i(x)^T)\Big]$$

(Refer Slide Time 7:04)



$$\underline{L_\gamma(x,\lambda,\mu)} = f(x) + \sum_{j=1}^{m} \lambda_j h_j(x) + \gamma\left(\sum_{j=1}^{m} [h_j(x)]^2 + \sum_{i=1}^{p}\left\{\left[\max\left(0, g_i(x) + \frac{\mu_i}{2\gamma}\right)\right]^2 - \frac{\mu_i^2}{4\gamma^2}\right\}\right)$$

$$\mu^{k+1} = \max(0, \mu^k + 2\gamma g(x)), \quad \lambda^{k+1} = \lambda^k + 2\gamma h(x)$$

Algorithm:

```
(i) Initialize $x^{(0)}$, $k=0$, tol, $\gamma>0$, $\lambda^{(0)}=(\lambda_1^{(0)} \dots \lambda_m^{(0)})$, $\mu^{(0)}=(\mu_1^{(0)}, \dots, \mu_p^{(0)})$
(ii) while ($|L_\gamma(x^k,\lambda^k,\mu^k) - f(x^k)| >$ tol):
        * Find $x^{k+1} = \underset{x\in\mathbb{R}^n}{\operatorname{argmin}} L_\gamma(x, \lambda^k, \mu^k)$
        * $\lambda^{k+1} = \lambda^k + 2\gamma h(x^{k+1})$, $\mu^{k+1} = \max(0, \mu^k, 2\gamma g(x^{k+1}))$
        * $k = k+1$
(iii) Output $(x^*, \lambda^*, \mu^*) = (x^k, \lambda^k, \mu^k)$.
```

$$\nabla L_\gamma(x,\lambda,\mu) = \nabla f(x) + \sum_{j=1}^{m} \lambda_j \nabla h_j(x) + 2\gamma\left(\sum_{j=1}^{m} h_j(x)\nabla h_j(x) + \sum_{i=1}^{p}\left[\max\left(0, g_i(x) + \frac{\mu_i}{2\gamma}\right)\right]\nabla g_i(x)\right)$$

$$\nabla_2 L_\gamma(x,\lambda,\mu) = \nabla_2 f(x) + \sum_{j=1}^{m} \lambda_j \nabla_2 h_j(x) + 2\gamma\Big(\sum_{j=1}^{m}\left[h_j(x)\nabla_2 h_j(x) + \nabla h_j(x)\nabla h_j(x)^T\right]$$

$$+ \sum_{i=1}^{p}\left[\max\left(0, g_i(x) + \frac{\mu_i}{2\gamma}\right)\nabla_2 g_i(x) + 1\left\{g_i(x) + \frac{\mu_i}{2\gamma} \geq 0\right\}\nabla g_i(x)\nabla g_i(x)^T\right]\Big)$$

All of this is the summation j equal to 1 to m. Now we will have to differentiate this right. This bracket is still open. Please note that. Here, the first term is simple.

You just write summation over i equal to 1 to p: $\max(0, g_i(x) + \mu_i/(2\gamma))$ Hessian of $g_i(x)$.

But the other term will be the following. I can open this summation outside. And here we will have, if $g_i(x) + \mu_i/(2\gamma)$ were greater than or equal to 0, then we will have $\nabla g_i(x) \nabla g_i(x)^T$.

So, this bracket for 2γ closes here. Of course, the Hessian here as well has a problem.

You can see that when $g_i(x) + \mu_i/(2\gamma)$ is greater than 0, we will have $\nabla g_i(x) \nabla g_i(x)^T$. When it is less than 0, we would not have it.

But at equal to 0, the left-hand side derivative of ∇L_γ and the right-hand side derivative of ∇L_γ are different. So, the Hessian actually does not exist when $g_i(x) + \mu_i/(2\gamma)$ is equal to 0.

That is true, but nevertheless, when you work this out, this works for many of the problems. We will continue using it.

We did that even when we had worked this for the quadratic penalty method, but nevertheless we were able to use the Hessian for many of the problems. We can continue doing that here as well.

And now that we have written down the gradient as well as the Hessian, we will start working out an example. We will again work out an example where you have only inequality constraints.

We will work out a problem which we already solved. If you see here, we have this particular problem where we are minimizing the distance from the point (3, 2) subject to the feasible set being a triangle with vertices (0, 0), (1, 0), and (½, ½).

We will work out the same problem using the Augmented Lagrangian Method. So, I will write down the problem here more clearly.

We are minimizing over x_1, x_2:

$\frac{1}{2}(x_1 - 3)^2 + \frac{1}{2}(x_2 - 2)^2$

Subject to the constraints such that the following constraints hold:

$-x_1 + x_2 \leq 0$

$x_1 + x_2 - 1 \leq 0$

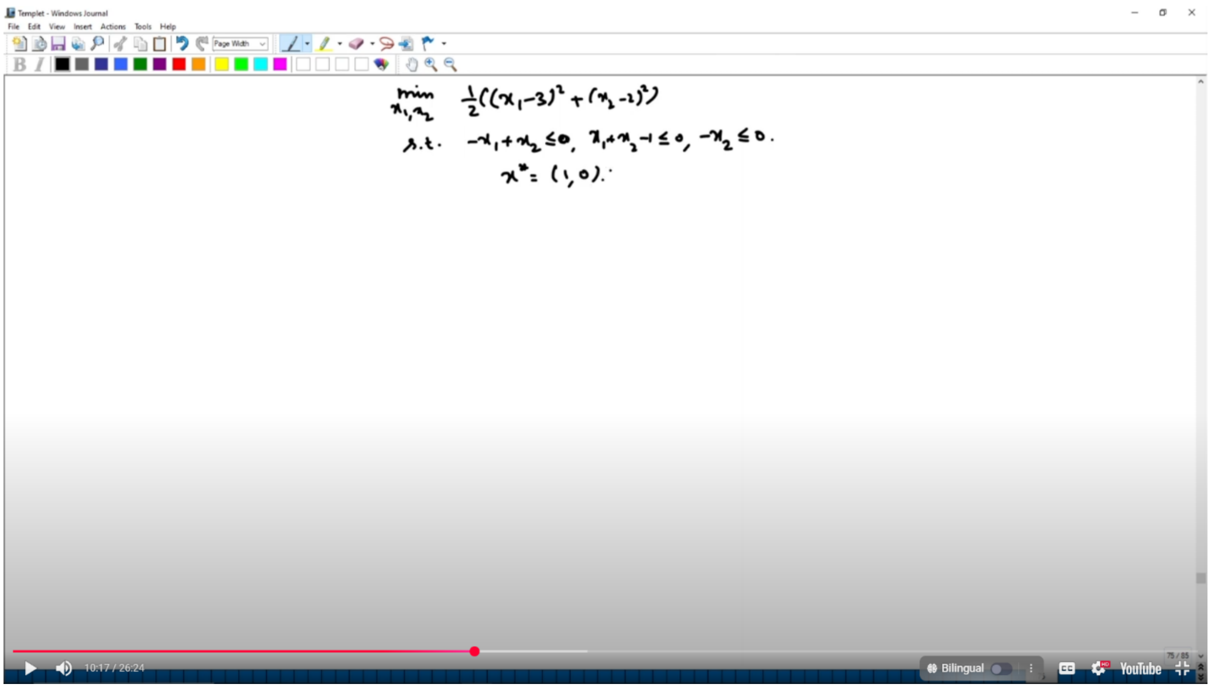
$-x_2 \leq 0$

I have just copied down what we had here.

So, $x_1 \geq x_2$ becomes $-x_1 + x_2 \leq 0$, and $x_1 + x_2 - 1 \leq 0$, and $-x_2 \leq 0$.

That is what I have written down here. And recall that the answer is (1, 0).

(Refer Slide Time 10:17)


Templet - Windows Journal
File Edit View Insert Actions Tools Help
10:17 / 26:24
Bilingual
YouTube

$$\min_{x_1,x_2} \frac{1}{2}((x_1-3)^2+(x_2-2)^2)$$

s.t. $-x_1+x_2 \leq 0,\ x_1+x_2-1 \leq 0,\ -x_2 \leq 0.$

$x^* = (1,0).$

We will try to solve this problem using Python now. This is a problem that we have solved earlier. We will take up f, g, and grad g from this that we have already used. I can also import NumPy, which will be required. And we can also include the name.

This is "Augmented Lagrangian method for inequality constraints".

So, we have defined the function which is $\frac{1}{2}(x_1 - 3)^2 + \frac{1}{2}(x_2 - 2)^2$ and the constraints:

$g_1(x) = -x_1 + x_2 \leq 0$

$g_2(x) = x_1 + x_2 - 1 \leq 0$

$g_3(x) = -x_2 \leq 0$

And this is the gradient. Now, we will copy this down.

Since this was the augmented Lagrangian method for equality constraints, we will copy this down and change it: the augmented Lagrangian function, its gradient, Hessian, and the code.

Of course, there needs to be a change here as well. L_γ, if you recall the function, we do not have h because there are no equality constraints. After f(x), we need to write this down.

Since we have a squared here and squared under summation, we will write them as the dot product of the vectors. We will do that here as well. we will have $\max(0, g_i(x) + \mu_i/(2\gamma))$ right.

We will write it as follows. We will write $(g(x) + \mu/(2\gamma)) > 0$ multiplied by $(g(x) + \mu/(2\gamma))$.

So, this will actually give us $g_i(x) + \mu_i/(2\gamma)$ if it is greater than 0; otherwise, it will give us 0.

This Boolean function is something that we are already familiar with. And instead of λ, we will have to use μ because this is μ.

This does this particular part: the max(0, $g_i(x) + \mu_i/(2\gamma)$) whole squared summation.

Now, you also have this summation: $-\sum \mu_i^2/(4\gamma^2)$. So, I can do a dot product here as well.

We will do $\mu \cdot \mu$ which is $\sum \mu_i^2$, divided by $(4\gamma^2)$.

What I have now done is I have written down L_γ. We will now write down the gradient.

The gradient of f(x) is $[x_0 - 3, x_1 - 2]$. Now we will write that here: np.array([x[0] - 3, x[1] - 2]).

And here again, we will have to make a change. The ∇h parts can be crossed out because there are no equality constraints. But we will still have this set of constraints, so we will write that down. Again we will use this term.

We will have 2γ * [(g(x) + μ/(2γ)) > 0] and then matrix multiply with grad g(x). I have just done 2γ into max($g(x) + \mu_i/(2\gamma)$, 0) and you are multiplying that with the grad g(x) matrix.

For the Hessian, the Hessian of f is the identity matrix I of size 2. And here again, we will have to write down all of this. The h can be crossed out because we do not have equality constraints.

And similarly, the Hessian of the g_i's can also be crossed out because the constraints are linear.

Therefore, the Hessian is 0.

You only have to have this right. So, we will write this down when $g_i(x) + \mu_i/(2\gamma)$ is greater than or equal to 0. Again we can have 2γ remains outside, and we can have this.

You should have grad g and this is grad g as well. So, you change this as μ. x is two-dimensional, so we have two zeros. μ is three-dimensional because we have three constraints, we have three zeros. γ and k: γ is one and k is zero. And these constraints remain the same.

λ's have to be modified into μ's; we will do them. So here as well, we can write this as x. So x is two-dimensional. Since μ is three-dimensional, I will put this here. So, μ here has to be modified: μ + 2γ g(x).

And again, if μ is greater than something, but that is something we will have to change as well.

And here we have this as μ; μ has three components.

Just going through the code, I have just defined f(x) to be $\frac{1}{2}(x_1 - 3)^2 + \frac{1}{2}(x_2 - 2)^2$, g(x) by these three constraints, the gradient of g, L_γ gradient, and the Hessian.

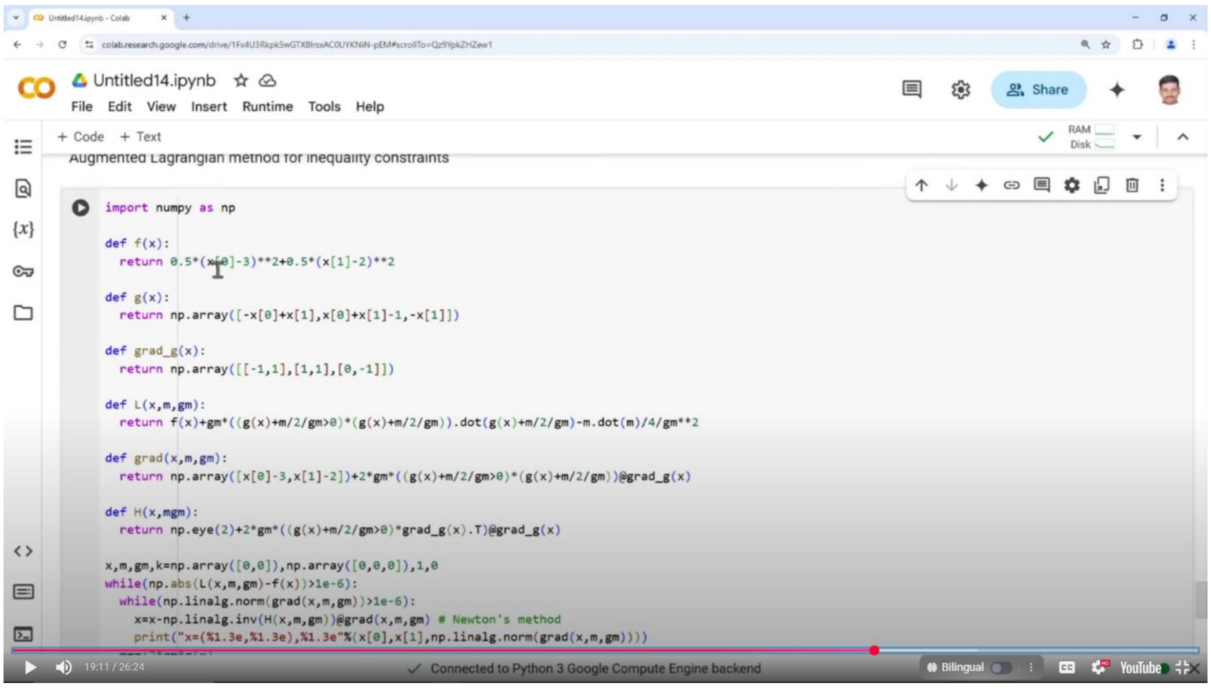
This code is just what we have here. So the initializations: whatever I have written down as the algorithm.

Initializations we have. There is a loop that is going on. In each loop, we are actually finding the minimum of the augmented Lagrangian function when λ_k and μ_k are constants. And after that, we modify λ and μ.

This loop goes on, and whenever the augmented Lagrangian function is close to the actual function value, then we stop.

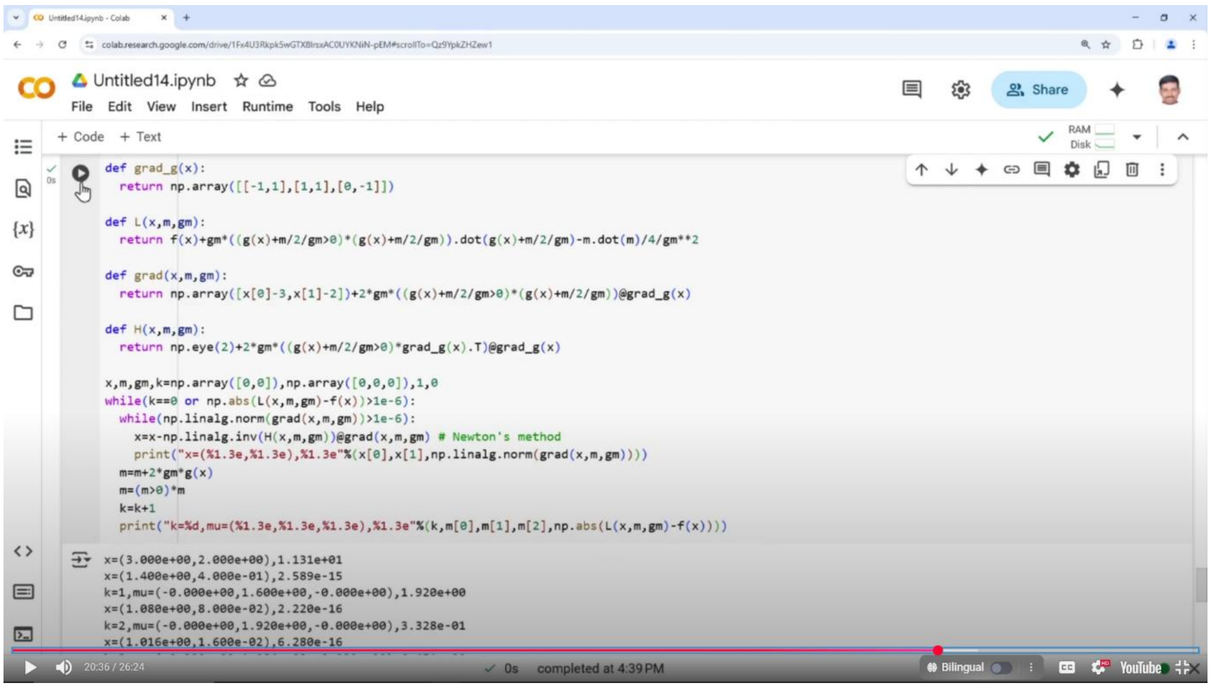
That is exactly what we are doing here. So, let us check if this works. Since I have put (0, 0), it is a feasible point. I need to add k = 0. That is the first step, k = 0. I think I missed a comma here. So, you can see that x is (1, 0) and μ is (0, 2, 0). This is something that we did not calculate, but it can be checked; not a big issue.

(Refer Slides Time 19:11-20:48)


Untitled14.ipynb
File Edit View Insert Runtime Tools Help
+ Code + Text
Share
RAM
Disk
19:11 / 26:24
Connected to Python 3 Google Compute Engine backend
Bilingual
YouTube

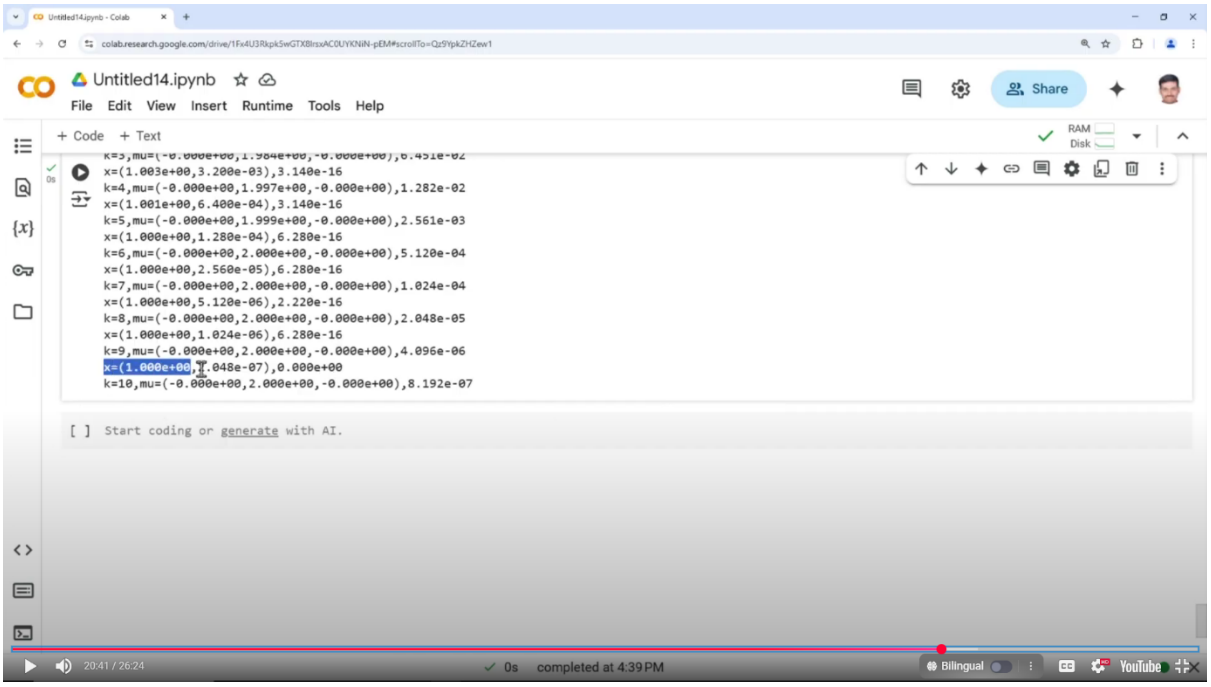
Augmented Lagrangian method for inequality constraints

```
import numpy as np

def f(x):
  return 0.5*(x[0]-3)**2+0.5*(x[1]-2)**2

def g(x):
  return np.array([-x[0]+x[1],x[0]+x[1]-1,-x[1]])

def grad_g(x):
  return np.array([[-1,1],[1,1],[0,-1]])

def L(x,m,gm):
  return f(x)+gm*((g(x)+m/2/gm>0)*(g(x)+m/2/gm)).dot(g(x)+m/2/gm)-m.dot(m)/4/gm**2

def grad(x,m,gm):
  return np.array([x[0]-3,x[1]-2])+2*gm*((g(x)+m/2/gm>0)*(g(x)+m/2/gm))@grad_g(x)

def H(x,mgm):
  return np.eye(2)+2*gm*((g(x)+m/2/gm>0)*grad_g(x).T)@grad_g(x)

x,m,gm,k=np.array([0,0]),np.array([0,0,0]),1,0
while(np.abs(L(x,m,gm)-f(x))>1e-6):
  while(np.linalg.norm(grad(x,m,gm))>1e-6):
    x=x-np.linalg.inv(H(x,m,gm))@grad(x,m,gm) # Newton's method
    print("x=(%1.3e,%1.3e),%1.3e"%(x[0],x[1],np.linalg.norm(grad(x,m,gm))))
```



```python
def grad_g(x):
  return np.array([[-1,1],[1,1],[0,-1]])

def L(x,m,gm):
  return f(x)+gm*((g(x)+m/2/gm>0)*(g(x)+m/2/gm)).dot(g(x)+m/2/gm)-m.dot(m)/4/gm**2

def grad(x,m,gm):
  return np.array([x[0]-3,x[1]-2])+2*gm*((g(x)+m/2/gm>0)*(g(x)+m/2/gm))@grad_g(x)

def H(x,m,gm):
  return np.eye(2)+2*gm*((g(x)+m/2/gm>0)*grad_g(x).T)@grad_g(x)

x,m,gm,k=np.array([0,0]),np.array([0,0,0]),1,0
while(k==0 or np.abs(L(x,m,gm)-f(x))>1e-6):
  while(np.linalg.norm(grad(x,m,gm))>1e-6):
    x=x-np.linalg.inv(H(x,m,gm))@grad(x,m,gm) # Newton's method
    print("x=(%1.3e,%1.3e),%1.3e"%(x[0],x[1],np.linalg.norm(grad(x,m,gm))))
  m=m+2*gm*g(x)
  m=(m>0)*m
  k=k+1
  print("k=%d,mu=(%1.3e,%1.3e,%1.3e),%1.3e"%(k,m[0],m[1],m[2],np.abs(L(x,m,gm)-f(x))))
```

```
x=(3.000e+00,2.000e+00),1.131e+01
x=(1.400e+00,4.000e-01),2.589e-15
k=1,mu=(-0.000e+00,1.600e+00,-0.000e+00),1.920e+00
x=(1.080e+00,8.000e-02),2.220e-16
k=2,mu=(-0.000e+00,1.920e+00,-0.000e+00),3.328e-01
x=(1.016e+00,1.600e-02),6.280e-16
```



```
k=3,mu=(-0.000e+00,1.984e+00,-0.000e+00),6.451e-02
x=(1.003e+00,3.200e-03),3.140e-16
k=4,mu=(-0.000e+00,1.997e+00,-0.000e+00),1.282e-02
x=(1.001e+00,6.400e-04),3.140e-16
k=5,mu=(-0.000e+00,1.999e+00,-0.000e+00),2.561e-03
x=(1.000e+00,1.280e-04),6.280e-16
k=6,mu=(-0.000e+00,2.000e+00,-0.000e+00),5.120e-04
x=(1.000e+00,2.560e-05),6.280e-16
k=7,mu=(-0.000e+00,2.000e+00,-0.000e+00),1.024e-04
x=(1.000e+00,5.120e-06),2.220e-16
k=8,mu=(-0.000e+00,2.000e+00,-0.000e+00),2.048e-05
x=(1.000e+00,1.024e-06),6.280e-16
k=9,mu=(-0.000e+00,2.000e+00,-0.000e+00),4.096e-06
x=(1.000e+00,[illegible].048e-07),0.000e+00
k=10,mu=(-0.000e+00,2.000e+00,-0.000e+00),8.192e-07
```

The answer here turns out to be μ* is (0, 2, 0). Verifying this is not very hard.

If you write down the Lagrangian (not the augmented Lagrangian), for inequality constraints we use μ.

So, the Lagrangian would be:

$\frac{1}{2}(x_1 - 3)^2 + \frac{1}{2}(x_2 - 2)^2 + \mu_1(-x_1 + x_2) + \mu_2(x_1 + x_2 - 1) + \mu_3(-x_2)$

If you write down the KKT conditions, so we will have:

For x_1: $(x_1 - 3) - \mu_1 + \mu_2 = 0$

For x_2: $(x_2 - 2) + \mu_1 + \mu_2 - \mu_3 = 0$

And the other constraints are, of course, that these three constraints have to be satisfied:

$-x_1 + x_2 \leq 0$

$x_1 + x_2 - 1 \leq 0$

$-x_2 \leq 0$

And other than this, we need μ's to be greater than or equal to 0. And we also have complementary slackness conditions:

$\mu_1(-x_1 + x_2) = 0$

$\mu_2(x_1 + x_2 - 1) = 0$

$\mu_3(-x_2) = 0$

You can solve this, because this set of conditions forms the KKT conditions.

But the other way is you can substitute this to check if it works or not. So, you can see that the first equation is satisfied because μ_1 is 0 and you have $(1 - 3) + 2 = 0$.

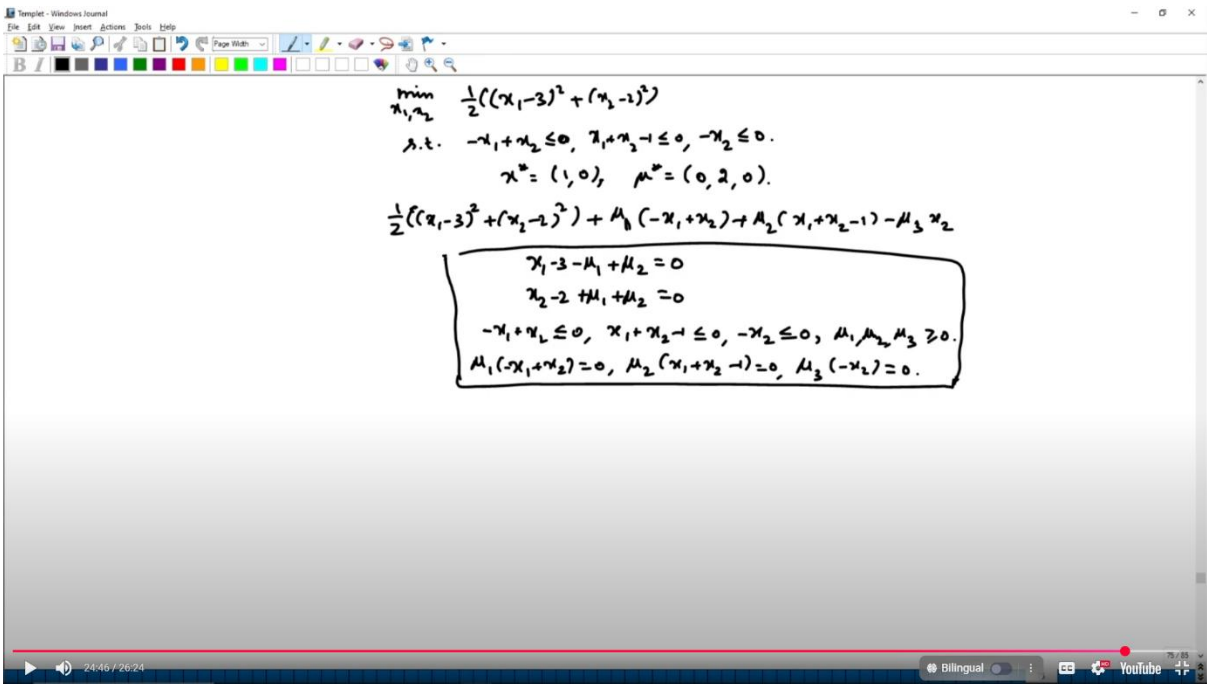
Similarly, here μ_1 is 0, so we have $(0 - 2) + 2 = 0$. And $-1 + 0 = -1 \leq 0$, $1 + 0 - 1 = 0$, and $-0 = 0$.

You can see that μ's are all greater than or equal to 0. And since $-x_1 + x_2$ is non-zero (it is -1), μ_1 has to be 0.

And $x_1 + x_2$ is equal to 1, so μ_2 can be greater than 0; it is 2. And x_2 is also 0, so μ_3 is greater than or equal to 0, but it happens to be 0 here.

This is just to justify that the Lagrange multipliers that we have here, (0, 2, 0), are actually the correct Lagrange multipliers.

(Refer Slide Time 24:46)


$$\min_{x_1,x_2} \frac{1}{2}((x_1-3)^2+(x_2-2)^2)$$

$$\text{s.t.} \quad -x_1+x_2\le 0,\ x_1+x_2-1\le 0,\ -x_2\le 0.$$

$$x^* = (1,0), \quad \mu^* = (0,2,0).$$

$$\frac{1}{2}((x_1-3)^2+(x_2-2)^2) + \mu_1(-x_1+x_2) + \mu_2(x_1+x_2-1) - \mu_3 x_2$$

$$\boxed{\begin{array}{c} x_1-3-\mu_1+\mu_2=0 \\ x_2-2+\mu_1+\mu_2=0 \\ -x_1+x_2\le 0,\ x_1+x_2-1\le 0,\ -x_2\le 0,\ \mu_1,\mu_2,\mu_3\ge 0. \\ \mu_1(-x_1+x_2)=0,\ \mu_2(x_1+x_2-1)=0,\ \mu_3(-x_2)=0. \end{array}}$$

That is one advantage of the Augmented Lagrangian Method. You not only get the solutions x_1, x_2, ..., x_n, but you also get the Lagrange multiplier values λ's and μ's.

This is for when you have a problem with only inequality constraints. In the forthcoming lecture, we will actually work out some practical examples.

If you recall in the very initial lectures, we gave a few practical examples, but we left them there because they were constrained optimization problems and we were looking at unconstrained optimization problems for a long time.

Now, given that we are actually working on constrained optimization problems, it is time that we look at those problems.

In the next lecture, we will work out one of those examples and solve it using the Augmented Lagrangian Method.

Thank you.