Optimization Algorithms: Theory and Software Implementation

Prof. Thirumulanathan D

Department of Mathematics

Institute of IIT Kanpur

Lecture: 37

Hello everyone, this is the second lecture in week 8. Recall that in the previous lecture, we introduced the augmented Lagrangian method. We discussed the differences between the quadratic penalty method and the augmented Lagrangian method, noting that while the procedural steps are similar, the augmented Lagrangian method resolves the issues associated with the quadratic penalty method.

Let us now consider an example to illustrate the augmented Lagrangian method. We will use the same example from earlier:

minimize $f(x) = x_1^2 + x_2^2 + x_3^2$

subject to the constraints

$h_1(x) = 3x_1 + x_2 + x_3 - 5 = 0$ and $h_2(x) = x_1 + x_2 + x_3 - 1 = 0$.

We already know the solution is

$x^* = (2, -0.5, -0.5)$ and the Lagrange multipliers are $\lambda^* = (-2.5, 3.5)$.

Notation: $\nabla_2 f(x) = \nabla^2 f(x)$=Hessian, $\mu_{k+1} = \mu^{k+1}$, $\lambda_k = \lambda^k$, $x_k = x^k$, $\lambda_{m,k} = \lambda_m^{(k)}$, $\mu_{n,k} = \mu_n^{(k)}$

The augmented Lagrangian function is defined as:

$L_\gamma(x, \lambda) = f(x) + \sum_j \lambda_j h_j(x) + \gamma \sum_j [h_j(x)]^2$.

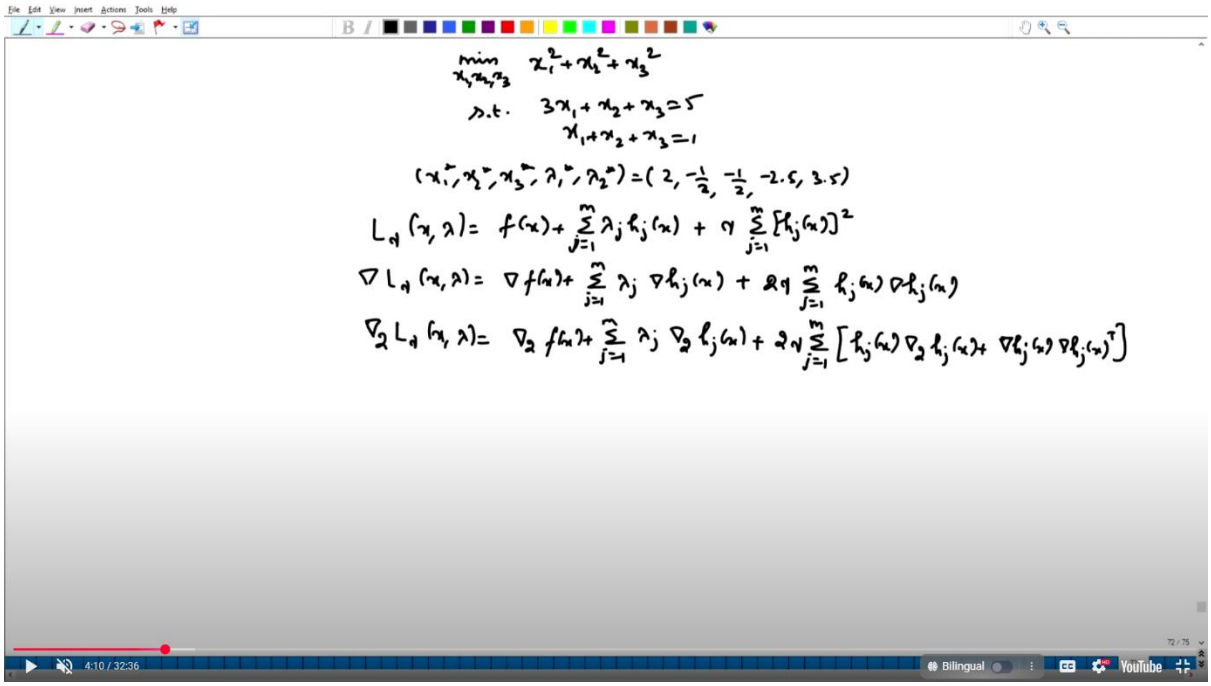
The gradient is:

$\nabla L_\gamma(x, \lambda) = \nabla f(x) + \sum_j \lambda_j \nabla h_j(x) + 2\gamma \sum_j h_j(x) \nabla h_j(x)$.

The Hessian is:

$\nabla^2 L_\gamma(x, \lambda) = \nabla^2 f(x) + \sum_j \lambda_j \nabla^2 h_j(x) + 2\gamma \sum_j [h_j(x) \nabla^2 h_j(x) + \nabla h_j(x) \nabla h_j(x)^T]$.

(Refer Slide Time 4:10)



We can implement this in Python by modifying the code for the quadratic penalty method. The key changes are:

1. Define $L_\gamma(x, \lambda) = f(x) + \lambda \cdot h(x) + \gamma (h \cdot h)$

2. Define $\nabla L_\gamma(x, \lambda) = \nabla f(x) + \lambda @ \nabla h(x) + 2\gamma\, h(x) @ \nabla h(x)$

3. Define the Hessian as $2I + 2\gamma\, \nabla h(x)^T \nabla h(x)$

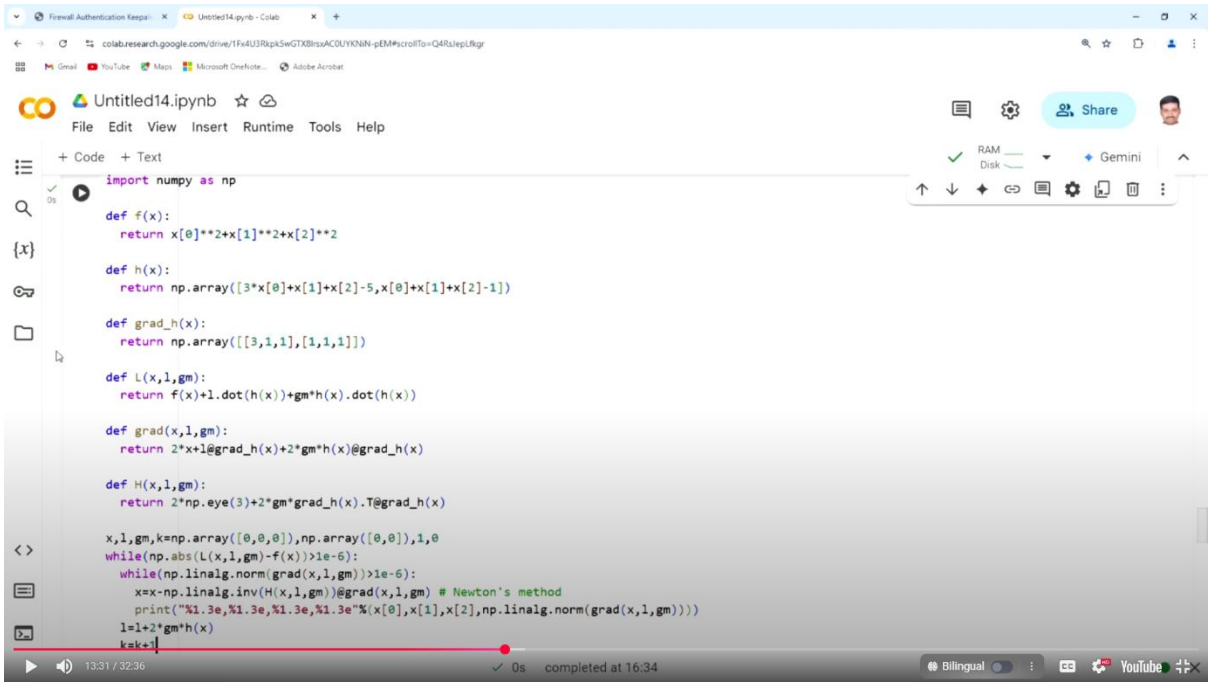
We initialize x_0, set $\gamma = 1$, and initialize $\lambda_0 = [0, 0]$.

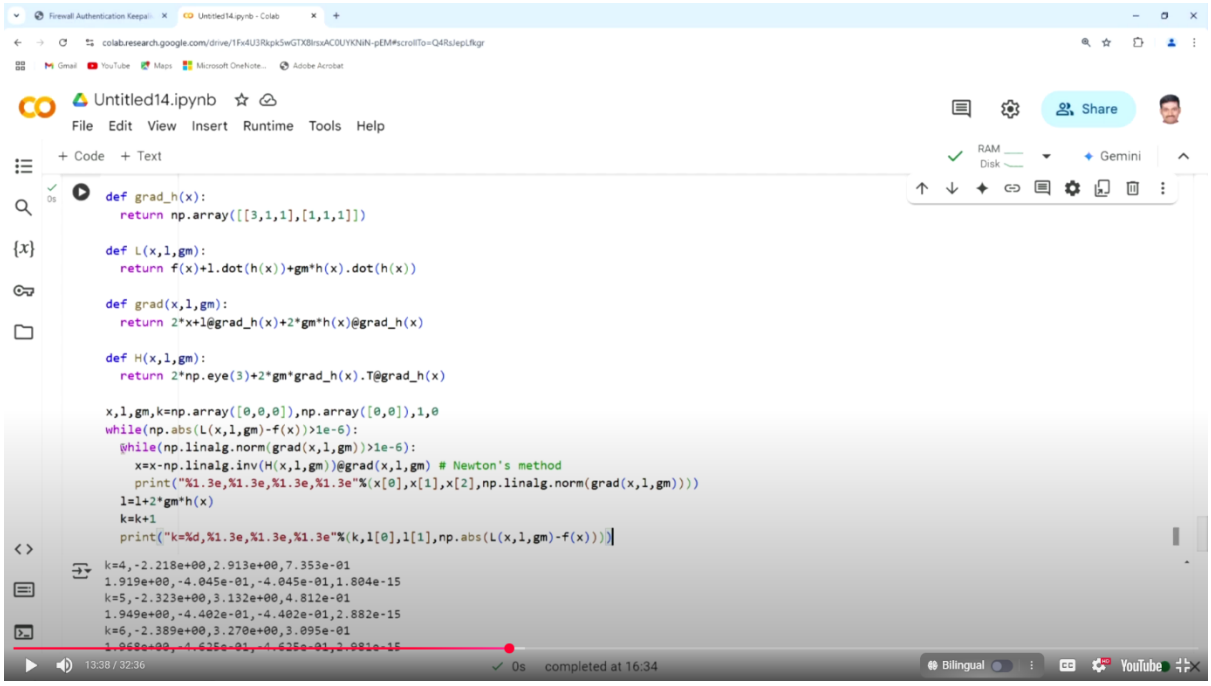
We then iterate until $|L_\gamma(x, \lambda) - f(x)| > 1e\text{-}6$ and $||\nabla L_\gamma(x, \lambda)|| > 1e\text{-}6$.

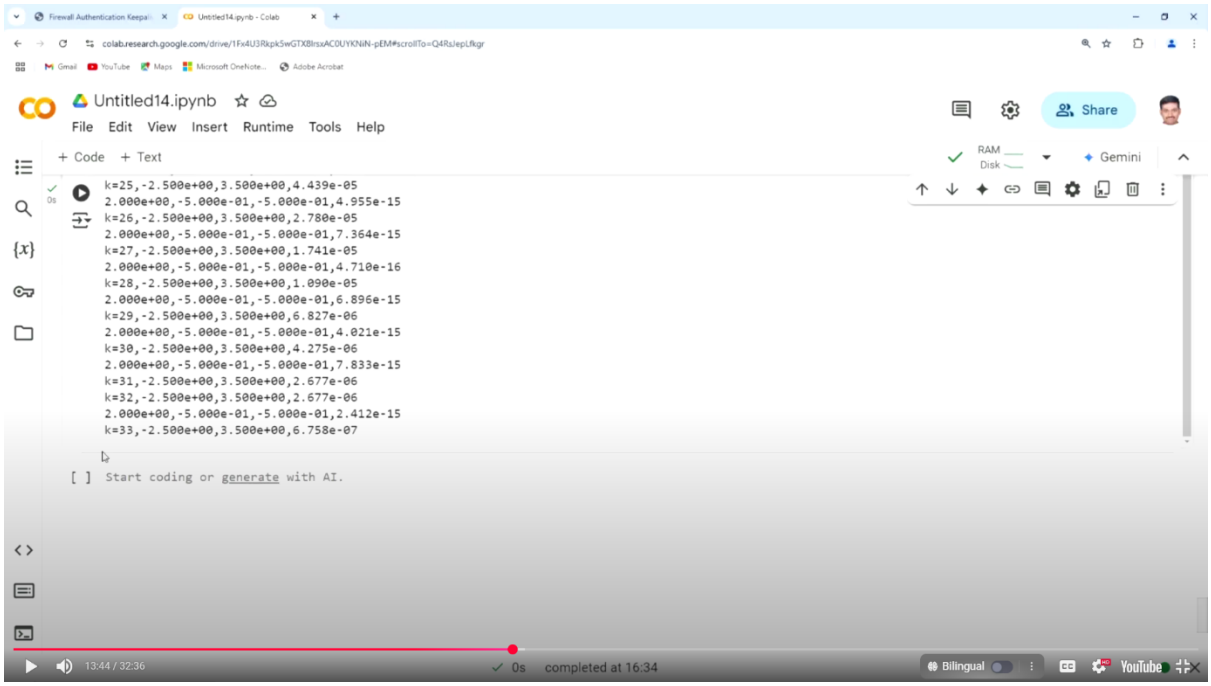
In each iteration, we update x using Newton's method and update λ as $\lambda_{k+1} = \lambda_k + 2\gamma\, h(x)$.

After running the code, we obtain $x^* = (2, -0.5, -0.5)$ and $\lambda^* = (-2.5, 3.5)$, which matches the expected solution.

(Refer Slides Time 13:30-13:45)


Untitled14.ipynb
File Edit View Insert Runtime Tools Help
Share
+ Code + Text
Gemini
import numpy as np
def f(x):
return x[0]**2+x[1]**2+x[2]**2
def h(x):
return np.array([3*x[0]+x[1]+x[2]-5,x[0]+x[1]+x[2]-1])
def grad_h(x):
return np.array([[3,1,1],[1,1,1]])
def L(x,l,gm):
return f(x)+l.dot(h(x))+gm*h(x).dot(h(x))
def grad(x,l,gm):
return 2*x+l@grad_h(x)+2*gm*h(x)@grad_h(x)
def H(x,l,gm):
return 2*np.eye(3)+2*gm*grad_h(x).T@grad_h(x)
x,l,gm,k=np.array([0,0,0]),np.array([0,0]),1,0
while(np.abs(L(x,l,gm)-f(x))>1e-6):
while(np.linalg.norm(grad(x,l,gm))>1e-6):
x=x-np.linalg.inv(H(x,l,gm))@grad(x,l,gm) # Newton's method
print("%1.3e,%1.3e,%1.3e,%1.3e"%(x[0],x[1],x[2],np.linalg.norm(grad(x,l,gm))))
l=l+2*gm*h(x)
k=k+1
13:31 / 32:36
0s completed at 16:34
Bilingual
YouTube


Untitled14.ipynb
File Edit View Insert Runtime Tools Help
Share
+ Code + Text
Gemini
def grad_h(x):
return np.array([[3,1,1],[1,1,1]])
def L(x,l,gm):
return f(x)+l.dot(h(x))+gm*h(x).dot(h(x))
def grad(x,l,gm):
return 2*x+l@grad_h(x)+2*gm*h(x)@grad_h(x)
def H(x,l,gm):
return 2*np.eye(3)+2*gm*grad_h(x).T@grad_h(x)
x,l,gm,k=np.array([0,0,0]),np.array([0,0]),1,0
while(np.abs(L(x,l,gm)-f(x))>1e-6):
while(np.linalg.norm(grad(x,l,gm))>1e-6):
x=x-np.linalg.inv(H(x,l,gm))@grad(x,l,gm) # Newton's method
print("%1.3e,%1.3e,%1.3e,%1.3e"%(x[0],x[1],x[2],np.linalg.norm(grad(x,l,gm))))
l=l+2*gm*h(x)
k=k+1
print("k=%d,%1.3e,%1.3e,%1.3e"%(k,l[0],l[1],np.abs(L(x,l,gm)-f(x))))
k=4,-2.218e+00,2.913e+00,7.353e-01
1.919e+00,-4.045e-01,-4.045e-01,1.804e-15
k=5,-2.323e+00,3.132e+00,4.812e-01
1.949e+00,-4.402e-01,-4.402e-01,2.882e-15
k=6,-2.389e+00,3.270e+00,3.095e-01
1.968e+00,-4.625e-01,-4.625e-01,2.081e-15
13:38 / 32:36
0s completed at 16:34
Bilingual
YouTube



Now, we extend the method to problems with inequality constraints. Consider the problem:

min f(x)

subject to

$g_i(x) \leq 0$ for i = 1 to p,

and

$h_j(x) = 0$ for j = 1 to m.

We convert inequality constraints to equality constraints by introducing slack variables s_i:

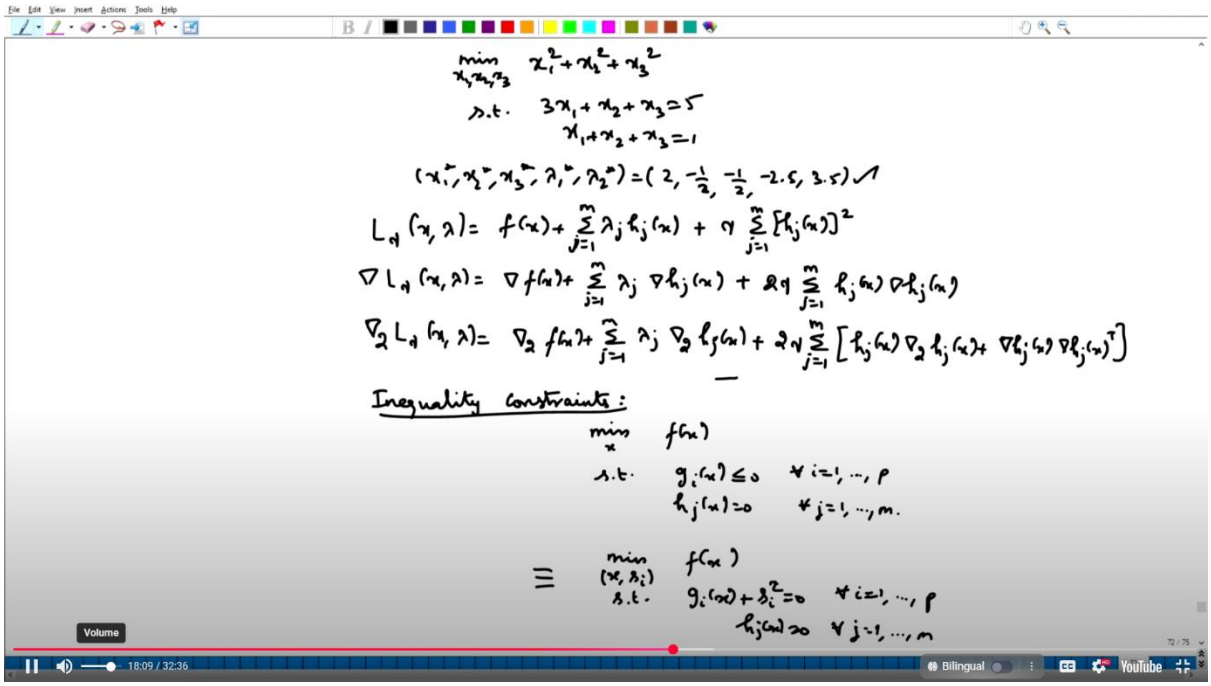
Which is equivalent to

min_{x,s_i} f(x)

s.t.

$g_i(x) + s_i^2 = 0$ for all i=1,2,…p and $h_j(x)=0$ for j=1,2,…m.

(Refer Slide Time 18:10)



The augmented Lagrangian becomes:

$L_\gamma(x, \lambda, \mu) = f(x) + \sum_{j=1}^{m} \lambda_j h_j(x) + \sum_{i=1}^{p} \mu_i (g_i(x) + s_i^2) + \gamma [\sum_{j=1}^{m} (h_j(x))^2 + \sum_{i=1}^{p} (g_i(x) + s_i^2)^2]$.

We minimize over s_i analytically. The optimal s_i* is given by:

$L_\gamma(x, \lambda, \mu) = f(x) + \sum_{j=1}^{m} \lambda_j h_j(x) + \gamma \sum_{j=1}^{m} (h_j(x))^2 + \gamma [\sum_{i=1}^{p} (g_i(x) + s_i^2 + \mu_i/(2\gamma))^2 - \mu_i^2/(4\gamma^2)]$.

This implies that

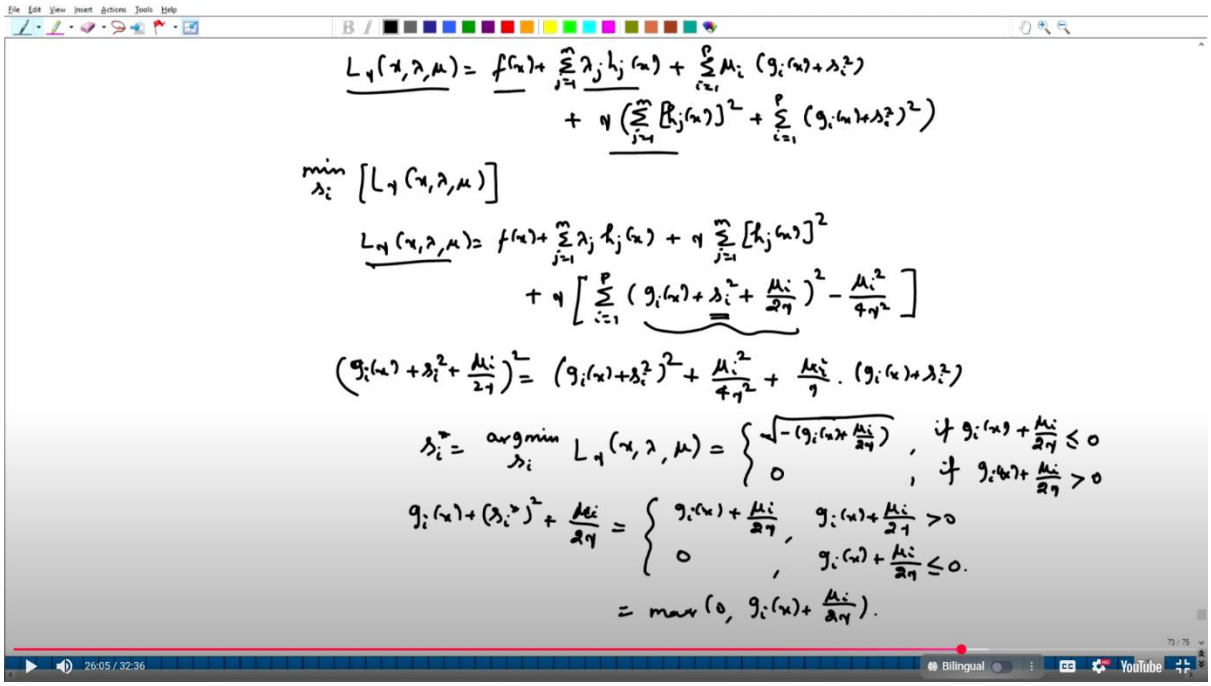
$(g_i(x) + s_i^2 + \mu_i/(2\gamma))^2 = (g_i(x) + s_i^2)^2 + \mu_i^2/(4\gamma^2) + (g_i(x) + s_i^2).\mu_i/\gamma$

$s_i^* = \text{argmin}_{s_i} L_\gamma(x, \lambda, \mu) = \sqrt{(g_i(x) + \mu_i/(2\gamma))}$ if $g_i(x) + \mu_i/(2\gamma) \leq 0$ and 0 if $g_i(x) + \mu_i/(2\gamma) > 0$

$g_i(x) + s_i^{*2} + \mu_i/(2\gamma) = (g_i(x) + \mu_i/(2\gamma))$ if $g_i(x) + \mu_i/(2\gamma) \leq 0$, and 0 otherwise.

$= \max(0, g_i(x) + \mu_i/(2\gamma))$

(Refer Slide Time)



Substituting back, we obtain:

$L_\gamma(x, \lambda, \mu) = f(x) + \sum_j \lambda_j h_j(x) + \gamma \sum_j (h_j(x))^2 + \gamma \sum_i [(\max(0, g_i(x) + \mu_i/(2\gamma)))^2 - \mu_i^2/(4\gamma^2)]$.

The update for μ is:

$\mu_{k+1} = \max(0, \mu_k + 2\gamma\, g(x))$.

The algorithm is:

Initialize x_0, $k = 0$, tolerance, $\gamma > 0$, $\lambda_0 = (\lambda_{1,0}, \lambda_{2,0}, \ldots \lambda_{m,0})$(size m), μ_0 (size p)$(\mu_{1,0}, \mu_{2,0} \ldots \mu_{p,0})$.

While $|L_\gamma(x_k, \lambda_k, \mu_k) - f(x_k)| >$ tolerance:

$x_{k+1} = \text{argmin}_x\, L_\gamma(x, \lambda_k, \mu_k)$

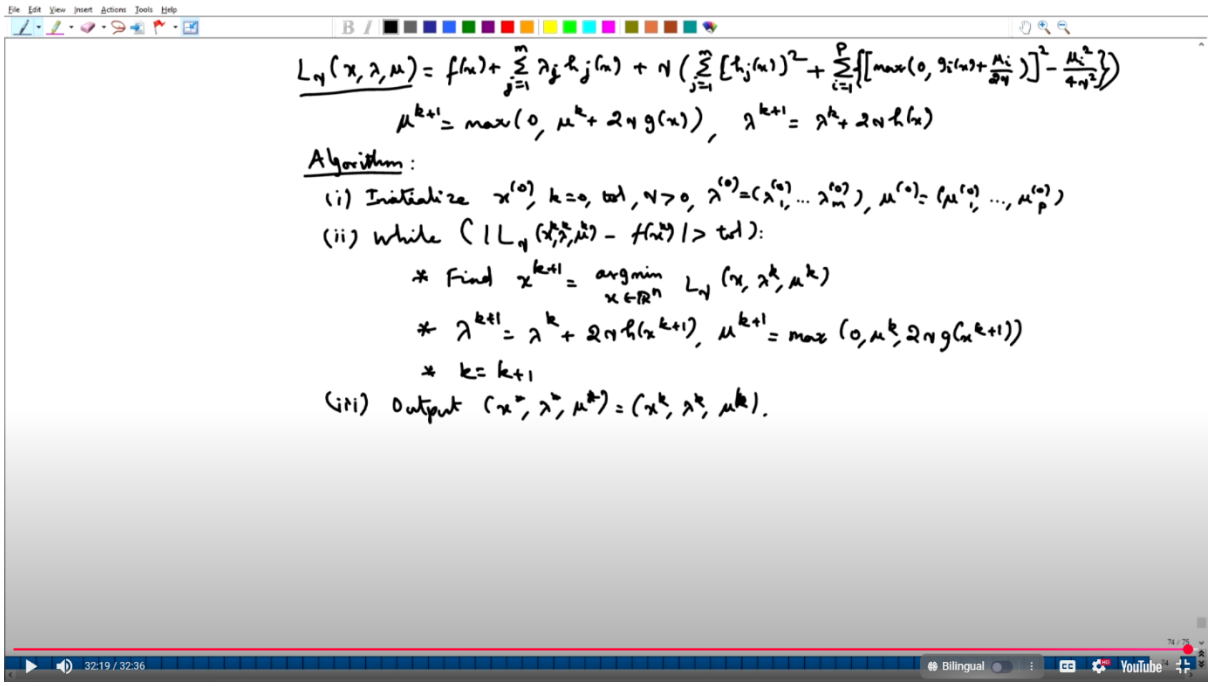
$\lambda_{k+1} = \lambda_k + 2\gamma\, h(x_{k+1})$

$\mu_{k+1} = \max(0, \mu_k + 2\gamma\, g(x_{k+1}))$

$k = k + 1$

Output $x^* = x_k$, $\lambda^* = \lambda_k$, $\mu^* = \mu_k$.

(Refer Slide Time 32:20)



This approach handles both equality and inequality constraints effectively.

We will continue with more details in the next lecture. Thank you.