Numerical Optimization

Prof. Shirish K. Shevade
Department of Computer Science and Automation
Indian Institute of Science, Bangalore

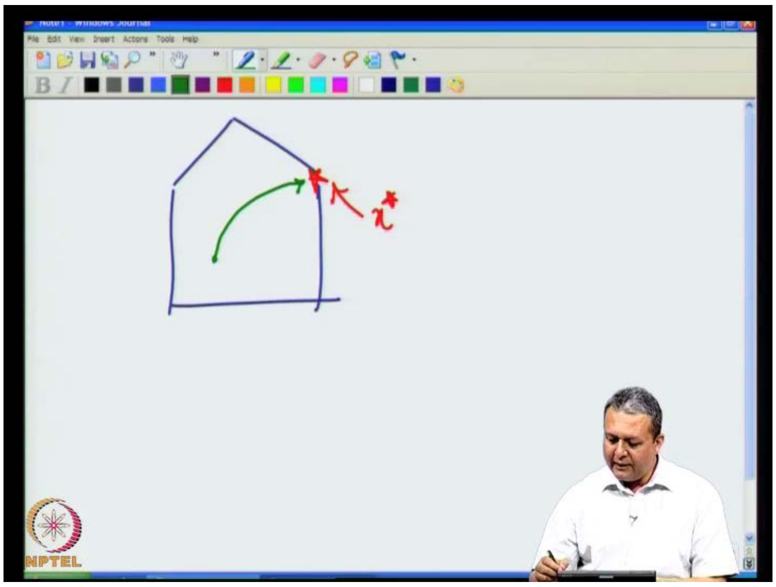
Lecture - 37
Karmarkar's Method

Hello welcome back. In the last class we saw that simplex algorithm is not a polynomial time algorithm. In particular we saw an example given by Klee and Minty, we showed that if we start from a particular point, then we may end up in visiting all the vertices of the feasible region before reaching the solution and the number of vertices could be exponential in number, and therefore simplex algorithm is not a polynomial time algorithm. So therefore, there was a need to devise algorithms which are polynomial time algorithms for linear programs, and which could work well for large dimensional problems.

So, in 70's Kutchian developed and ellipsoid method which was the first polynomial time algorithm for linear programming, and in 1984 Karmarkar developed a polynomial time algorithm which has complexity better than that of Kutchian and which was performing as well as the simplex algorithm. So the ideas Karmarkar used where based on the interior point methods.

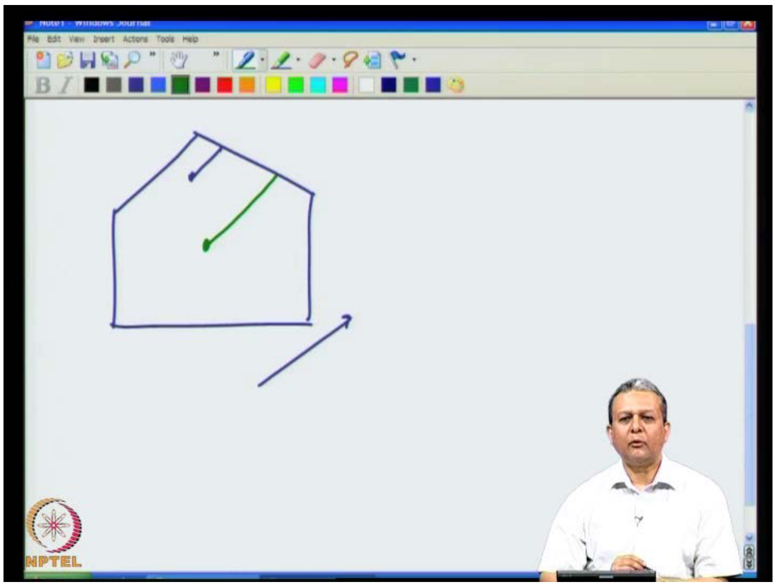
So, in as I mentioned in the last class the interior point methods generate a sequence of points in the interior of the feasible region, and these sequence of points is generated till the final solution is reached. Of course, for a linear program as we know the solution optimal solution exists at an extreme point, so this interior point methods generate a sequence of points which are in the interior of the feasible region, but they converge to a point which would be very close to the optimal solution which is a vertex, which is a boundary point.

(Refer Slide Time: 02:46)



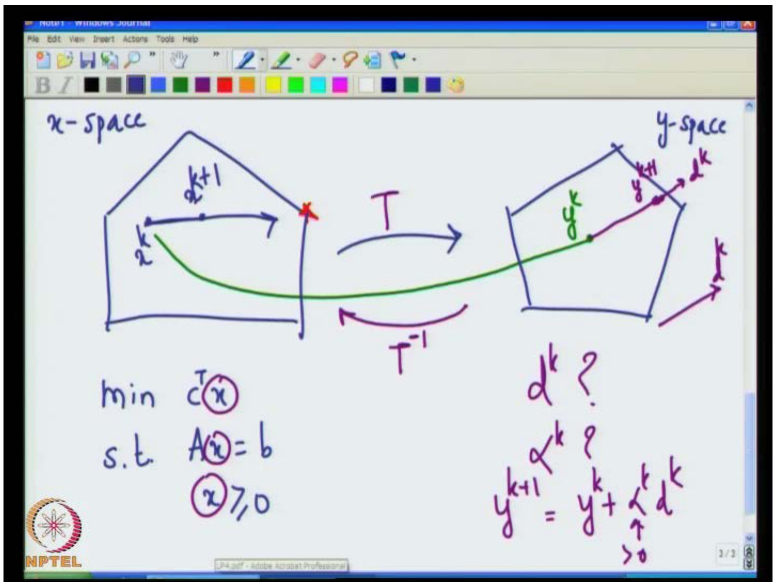
So, as we discussed in the last class, so the idea of the interior point method is that suppose the feasible region is as shown here, and suppose that the extreme point the solution is at this point. So this is x star, so a typical interior point method would start from an interior point and would follow a path which is in the interior and finally, it will reach a point which is very close to the solution and the Karmarkar's method was based on some of the novel ideas.

(Refer Slide Time: 03:36)



So, one of the ideas was that if we have a point so, if this is the direction along which we want to make a movement or we want to move, so, that the there is improvement in the objective function. Now if we have a point which is here then if we make a movement along this direction we would reach a point which is somewhere close to the boundary on the other hand we are close to the center of this feasible region. So; that means, if we are at point which is somewhere here and if we make a movement then you will see that starting from a point which is close to the center we can make a considerable improvement in the objective function compared to any point which is not close to the center of the feasible region. So, it is better to have a point which is close to the center of the feasible region so, that we can make a reasonable improvement in the objective function.

(Refer Slide Time: 05:10)



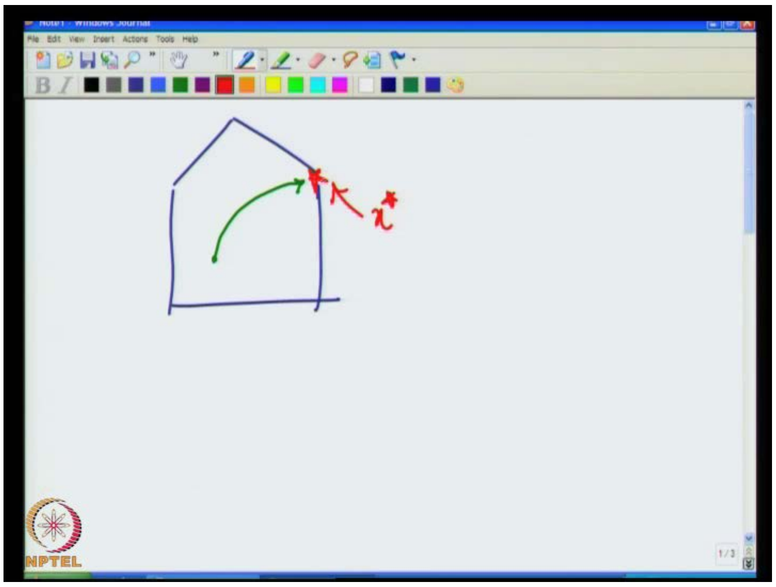
So Karmarkar suggested the following idea that let us take a simple feasible region and let us take any point and let us call it x k. So Karmarkar suggested to use a transformation to transform this feasible region into some other feasible region in different space. So let us call this as the x space and this as y space. Now this transformation should be done such that the point x k that we have here get transformed to a point which is y k in the y space such that this point y k is close to the center of the feasible region and this is done to make sure that the improvement in the objective function is considerable.

Now in the y space will need a direction to move because in the original space the problem was to minimize c transpose x subject to A x equal to b, x greater or equal to 0. Now when we use the transformation T to project this points in the y space the variables x get changed to y and this will become a problem in the y space and in the y space we need to find out what is the direction d k along which we need to move. Now if we look at the x space as I mentioned in the last class that the idea is to use non-linear programming algorithms to solve this problem and therefore, since the objective function is continuously differentiable steepest descent direction is the best choice or the preferred choice. So if we move along the steepest descent direction then we may leave the feasible region.

So what we need to do is that we need to project the steepest descent direction on to this set. Now when we transform the original problem into a new problem which need not be a linear program, how do we find the direction or the steepest descent direction in the new space which is projected on to the constraint set? So that is the first question that we would like to answer. Now having found the direction to move then the next step is to find alpha k. So that we would have y k plus 1 is equal to y k plus alpha k d k like what we had in our discussion on the unconstraint optimization problem, that the new point is formed using the current point by moving by moving along the direction d k with a step length of alpha k remember that, this quantity alpha k is greater than 0. So from y k so suppose we get a direction d k to be this, so we make a movement of so we make a movement of alpha k along the direction d k.

So remember that, this is the direction d k and we need to move along this direction. So that we do not cross the boundary, so we would reach a point which is going to be y k plus 1. So this point y k plus 1 is obtained based on the current value of alpha k and the direction d k. Now having the reached the point which is close to the boundary we would need to see what is the corresponding point in the x space. So, the corresponding point to find the corresponding point we need a map which is from the y space to the x space let us call that map as T inverse. So the map T that we get should be invertible so; that means, that from x k we go to point y k from y k we make a movement along the direction d k with a step length of alpha k and from y k plus 1 we come back to a point which could be x k plus 1. So we have made a movement in this direction now, suppose that this point was optimal. So; that means, you will see that we have made a movement towards the direction of the optimal point.

(Refer Slide Time: 11:37)

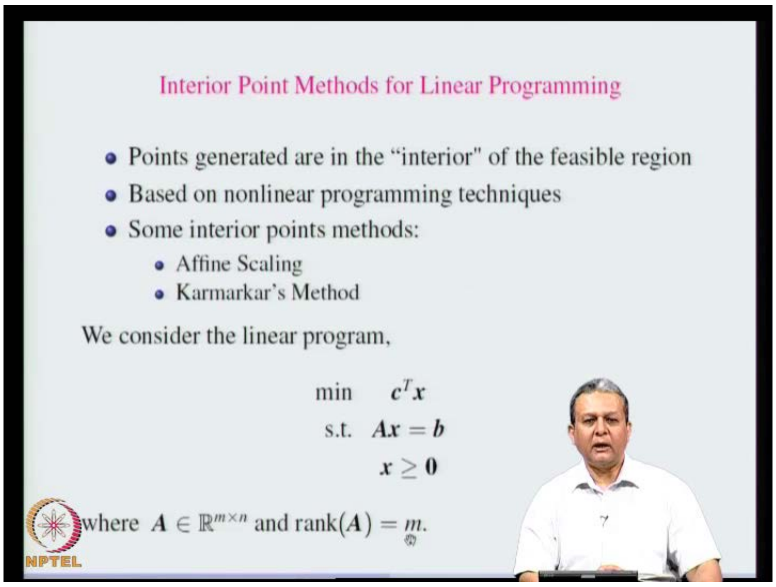


So note that this movement need not be in the straight line, as I showed earlier one could have a movement which could be represented by an arbitrary curve towards the final point.

Once we find x k plus 1 then again we use the transformation T. Now this time a new feasible region would be formed in the y space and from that feasible region we find out that point y k plus 1. So that point will be different from this because now we have used a different T based on the current value of x k plus 1 and this procedure is repeated so finally, we may get the path in the input space which could be something like this and finally, we will go close to the solution.

So this was the idea proposed by Karmarkar in his projective scaling algorithm. Now his algorithm was modified to devise and affine scaling algorithm which is a which uses a very simple transformation T. So in today’s lecture we will first see the affine scaling algorithm which was an extension of Karmarkar’s algorithm, note that kutchian’s algorithm was a polynomial time algorithm, Karmarkar’s algorithm was also polynomial time algorithm but, the affine scaling algorithm is not a polynomial time algorithm for solving linear programs but, nevertheless it is useful to understand some of the concepts that were used in Karmarkar’s algorithm.

(Refer Slide Time: 13:37)



Interior Point Methods for Linear Programming

- Points generated are in the "interior" of the feasible region
- Based on nonlinear programming techniques
- Some interior points methods:
  - Affine Scaling
  - Karmarkar's Method

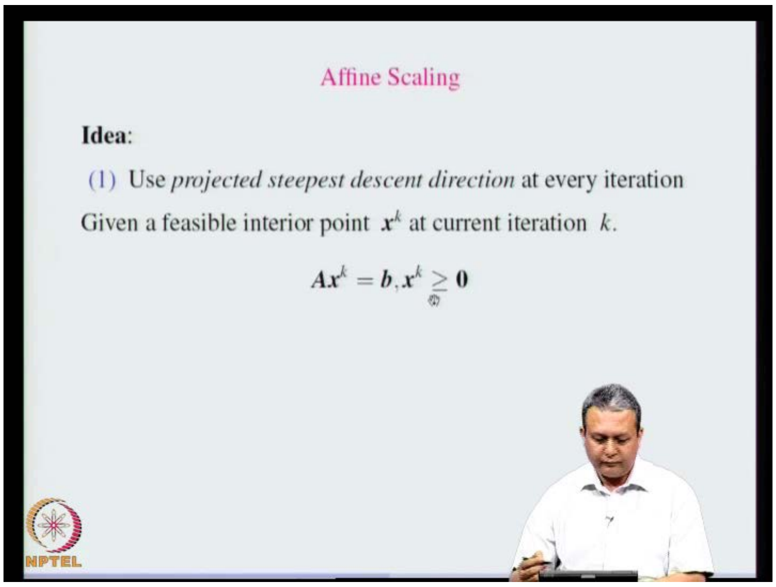
We consider the linear program,

$$\begin{aligned} \min \quad & \boldsymbol{c}^T \boldsymbol{x} \\ \text{s.t.} \quad & \boldsymbol{A}\boldsymbol{x} = \boldsymbol{b} \\ & \boldsymbol{x} \geq \boldsymbol{0} \end{aligned}$$

where $\boldsymbol{A} \in \mathbb{R}^{m \times n}$ and $\text{rank}(\boldsymbol{A}) = m$.

So let us look at the affine scaling algorithm to solve linear programs. So as I mentioned that in the interior point methods, the points are generated in the interior of the feasible region and they are based on non-linear programming techniques like steepest descent method and chronologically Karmarkar's method was developed first and then the affine scaling algorithm but, we will study affine scaling method first before moving on to Karmarkar's method. So we will continue to consider the standard linear program, minimize c transpose x subject A x equal to b and x non-negative with the assumption that A is full row rank matrix A is a matrix of size m by n.

(Refer Slide Time: 14:27)



Affine Scaling

Idea:

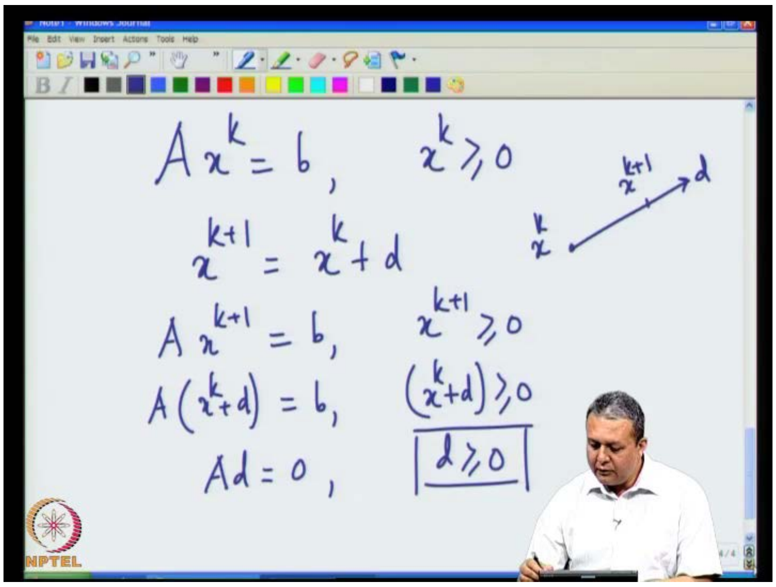
(1) Use *projected steepest descent direction* at every iteration

Given a feasible interior point $\boldsymbol{x}^k$ at current iteration k.

$$\boldsymbol{A}\boldsymbol{x}^k = \boldsymbol{b}, \boldsymbol{x}^k \geq \boldsymbol{0}$$

So affine scaling method uses the idea of projected the steepest descent direction at every iteration. So suppose we are given a feasible point which is in the interior of the feasible region and the current iteration k, since the point is feasible it satisfies the constraints A x equal to b and x k greater than or equal to 0.

(Refer Slide Time: 15:05)



$$A x^k = b, \quad x^k \geq 0$$

$$x^{k+1} = x^k + d$$

$$A x^{k+1} = b, \quad x^{k+1} \geq 0$$

$$A(x^k + d) = b, \quad (x^k + d) \geq 0$$

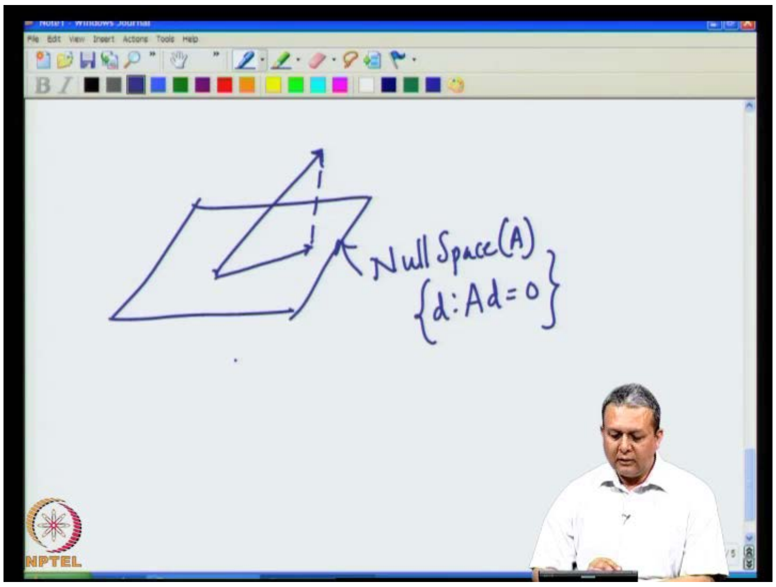
$$Ad = 0, \quad \boxed{d \geq 0}$$

So we have A x k equal to b and x k greater than or equal to 0. Now suppose we want to move along the direction d, so x k plus 1 let us assume that x k plus 1 is x k plus d, now the ideas which I am going to talk about they also hold in the y space, although I am discussing everything related to the x space these ideas also work for the y space.

So lets us assume that the step length is of size 1 so the alpha is 1. So x k is so x k plus 1 is equal to x k plus d. So if we have a point x k and we have direction d we take a suppose a unit step along this direction to move to this point x k plus 1. Now the new point also should satisfy this constraints, so we also need to have x k plus 1 is equal to b and x k plus 1 greater than or equal to 0. So therefore, A x k plus d should be equal to b and x k plus d should be greater than or equal to 0. So since we already have z x k equal to b so what we get a d is equal to 0 and x k is already greater than or equal to 0. So we get d greater than or equal to zero.

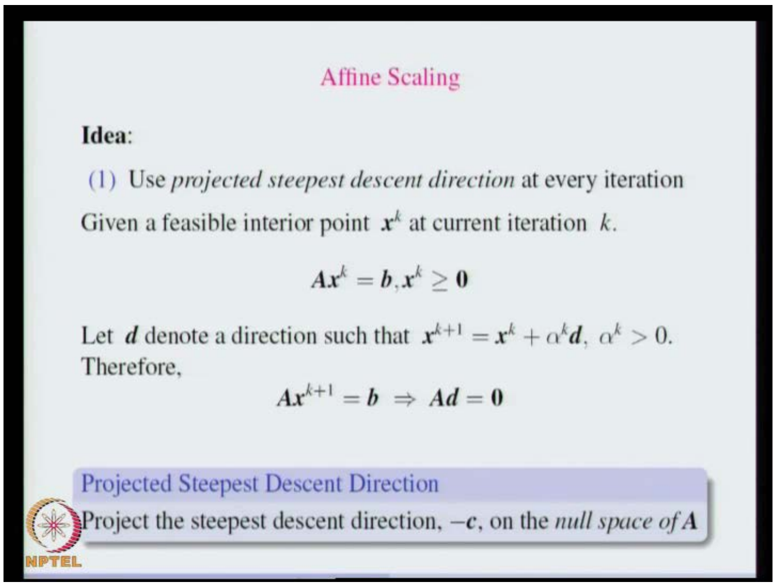
So the direction that we are going to choose should be such that should be in the null space of the matrix A and moreover all its components have to be non-negative. Now this requirement can be typically handled using the algorithm, so let us not worry about this at this movement. Let us mainly concentrate on the fact that the direction d along which we should move should be in the null space of the matrix A.

(Refer Slide Time: 17:37)



So suppose we have some direction along which we need to move, for example, it could be a steepest descent direction. Now the steepest descent direction may not ensure the feasibility, so what we need to do is that we need to find the null space of the matrix A and project this steepest descent direction on to. So this space is the null space of the matrix A.

(Refer Slide Time: 18:47)



Affine Scaling

Idea:

(1) Use *projected steepest descent direction* at every iteration

Given a feasible interior point x^k at current iteration k.

$$Ax^k = b, x^k \geq 0$$

Let d denote a direction such that $x^{k+1} = x^k + \alpha^k d$, $\alpha^k > 0$.
Therefore,

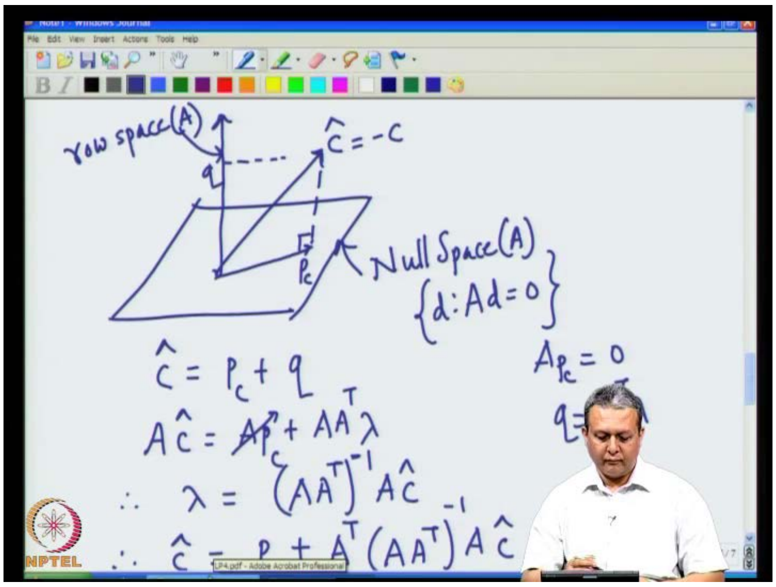
$$Ax^{k+1} = b \Rightarrow Ad = 0$$

Projected Steepest Descent Direction

Project the steepest descent direction, $-c$, on the *null space of A*

So null space of the matrix is the set of all directions d such that A d is equal to 0. So if we have any direction along which we need to make a movement, first we need to project that on to the null space of the matrix A. So how to do that that we will see now. So if we choose a step length alpha k where alpha k is positive then x k plus 1 will be denoted by x k plus alpha k d. And therefore, A x plus 1 equal to b that is x k plus 1 is a feasible point implies that A d equal to 0. So in the affine scaling method it is suppose to use the projected steepest descent direction, so which is the projection of steepest descent direction on the null space of the matrix A. So let us see how to do that.

(Refer Slide Time: 19:29)



$$\hat{c} = p_c + q$$

$$A_{p_c} = 0$$

$$A\hat{c} = \cancel{Ap_c} + AA^T\lambda$$

$$\therefore \quad \lambda = (AA^T)^{-1} A\hat{c}$$

$$\therefore \quad \hat{c} = p + A^T (AA^T)^{-1} A\hat{c}$$

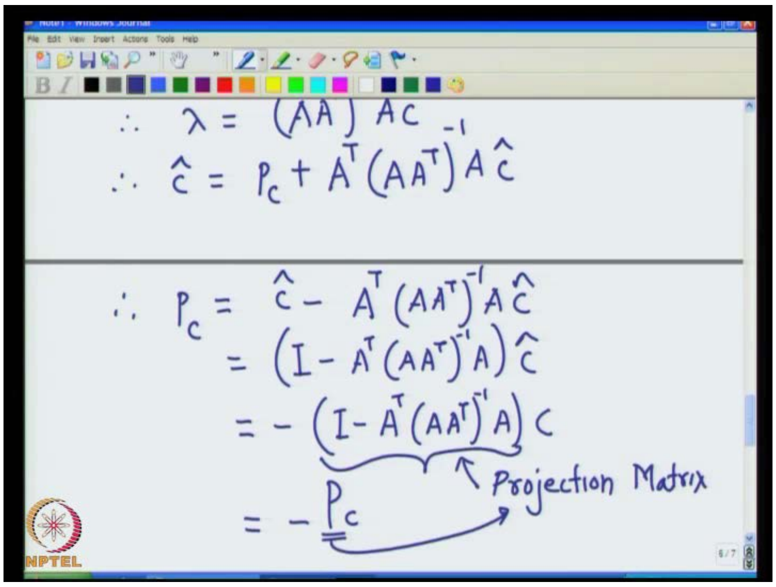
So let us call this as c hat and c hat is nothing but, suppose minus c. So this is the steepest descent direction, because the objective function is c transpose x so the gradient of the objective function is c and the negative of the gradient is minus c. So this is the steepest descent minus c and which we want to project it to the null space. So lets us denote this component by p c projection of the steepest descent direction.

Now we know that there is a space which is orthogonal component of null space of A and that is this space is called the row space of A. So any vector c, c hat which is in the n dimensional space can be split into 2 parts, 1 is the projection so this is the orthogonal projection of c hat onto the null space of A and then the other component will be the projection in the projection on the row space of A. So let us call this as q.

So c hat can be written as pc plus q. Note that the 2 space s which are not mentioned here row space of A and null space of A they are orthogonal components of A. So c hat can be uniquely represented as pc plus q where p c denotes the component of c hat on the null space of A and q denotes the row space of A. Now since p c belongs to the null space of A what we have is A p c equal to 0 because it satisfies this property and q belongs to the row space of A, so q can be written as A transpose lambda. Now given c hat we want to find out what is p c? Now for that purpose we first need to find out what is q, and q is nothing but, A transpose lambda but, for finding q we need what is the value of lambda.

So let us use this fact to find out the value of lambda. So if you multiply throughout by a so what we get A c hat is equal to A p c plus A q is nothing but, A transpose lambda. Now this quantity is 0. So therefore, since A is a full row rank matrix we can write lambda to be A A transpose inverse A c hat. So this is the value of lambda we get. Now we can c as therefore, c hat can be written as p c plus q, q is A transpose lambda, so A transpose a AA transpose inverse A c hat. So this is the vector c hat obtained as combination of p c plus q. So we know A the matrix A, we know c hat, so it is easy to calculate this quantity and therefore.

(Refer Slide Time: 24:11)



$$\therefore \; \lambda = (AA)\; Ac \quad {}^{-1}$$

$$\therefore \; \hat{c} = P_c + A^T (AA^T) A\hat{c}$$

$$\therefore \; P_c = \hat{c} - A^T (AA^T)^{-1} A\hat{c}$$

$$= \left(I - A^T (AA^T)^{-1} A\right)\hat{c}$$

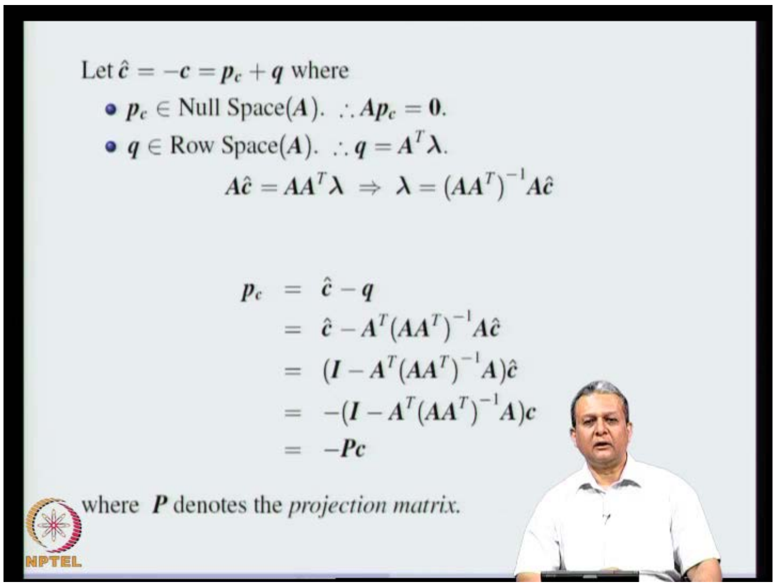
$$= -\underbrace{\left(I - A^T (AA^T)^{-1} A\right)}\, c \quad \nwarrow \text{Projection Matrix}$$

$$= -\underline{\underline{P_c}}$$

And therefore p c which is the projection of the vector c hat on the null space of A is nothing but, c hat minus A transpose A A transpose inverse A c hat and this is nothing but, identity matrix minus A transpose A A transpose inverse A into c hat.

Now c hat is nothing but, negative of c. So this can be written as minus identity minus A transpose A A transpose inverse A c. So this matrix what we get here this is called the projection matrix. So this matrix projects c on to the null space. So with the projection of minus c will be as shown here. So we can write this is as minus p c now this capital p this is a projection matrix and c is a vector. Remember that this small p lower case p is a component of c hat on the null space of matrix A. So given a direction d, which could be a steepest descent direction, we can use projection matrix to project it on to the null space of any matrix using this formula.

(Refer Slide Time: 26:45)



Let $\hat{c} = -c = p_c + q$ where

- $p_c \in$ Null Space(A). $\therefore Ap_c = 0$.
- $q \in$ Row Space(A). $\therefore q = A^T\lambda$.

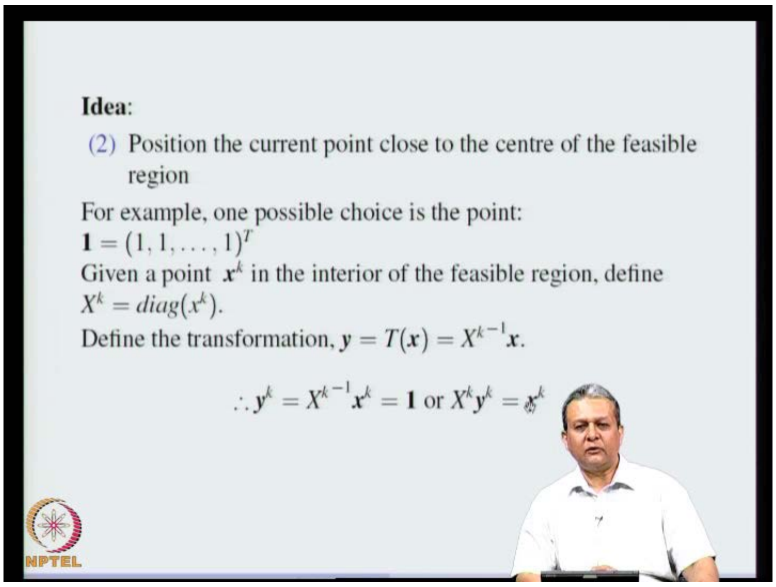
$$A\hat{c} = AA^T\lambda \Rightarrow \lambda = (AA^T)^{-1}A\hat{c}$$

$$\begin{aligned} p_c &= \hat{c} - q \\ &= \hat{c} - A^T(AA^T)^{-1}A\hat{c} \\ &= (I - A^T(AA^T)^{-1}A)\hat{c} \\ &= -(I - A^T(AA^T)^{-1}A)c \\ &= -Pc \end{aligned}$$

where P denotes the *projection matrix.*

So as I mentioned that if we denote c hat to be minus c, minus c is the negative descent direction and it has 2 components p c and q where p c belongs to the null space of A and q belong to row space A. And therefore, A p c is equal to 0 and A transpose lambda q and the using the fact that A c hat is equal to A p c plus a q is 0 so A q is nothing but, a transpose lambda and since A is full row rank matrix A A transpose is invertible and therefore, we get a value of lambda which can be used to find out q and therefore, using p c equal to c hat minus q what we can do is that we can write it as c hat minus A transpose A A transpose inverse A c hat. And this is nothing but, minus p c where the matrix p denotes the projection matrix. So any direction can be projected on to the constraint using the projection matrix.

(Refer Slide Time: 27:56)



Idea:

(2) Position the current point close to the centre of the feasible region

For example, one possible choice is the point:
$\mathbf{1} = (1, 1, \dots, 1)^T$
Given a point x^k in the interior of the feasible region, define
$X^k = diag(x^k)$.
Define the transformation, $y = T(x) = {X^k}^{-1}x$.

$$\therefore y^k = {X^k}^{-1}x^k = \mathbf{1} \text{ or } X^k y^k = x^k$$

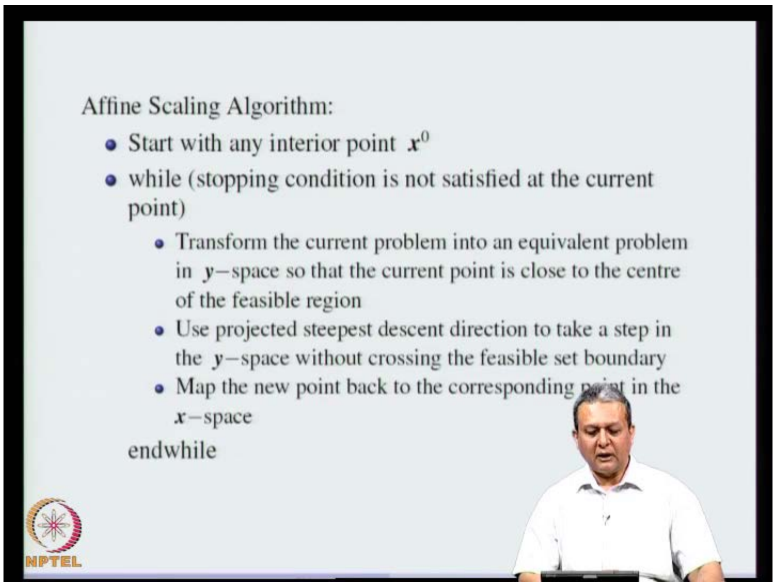
Now the second idea that is used in affine scaling method is the positioning of the current point to the center of the feasible region. So if the current escapes S in the x space can be projected using some transformation p to the point or to a point which is close to the center of the feasible region.

So in affine scaling of preferred choice for this point which is close to the center of feasible point is the point where all the coordinated are 1. So all the coordinated have equal values and that is a preferred choice in affine scaling to be point close to the center of the feasible region. So if you are given a point x k in the interior of the feasible region in the x space, so lets us define the matrix x k to be having a diagonal structure, where each entry in diagonal represents one of the components of vector x k. Now having defined the matrix, let us define the transformation y transformation t as y equal to t x where t x is nothing but, x k inverse x.

So you will see that since x k was obtained by placing the diagonal entries of the vector x by placing the entries of x along the diagonal of matrix x k inverse x will be a unit vector or the vector of all one’s. And therefore, y k will be x k inverse x k will be a vector of all one’s. So by defining this transformation we were able to get for any x k by defining a the matrix x k to be diagonal of x k, the point in the y space for the feasible region will have all the entries equal to 1. And note that x k's are all greater than 0, because the given point x k is in the interior of the feasible region so x k's are all greater than 0 so this inversion is also possible.

And because of which we can write x y k as x k x k y k as x k. So given x we can use this transformation t to convert it to the y space. So y is nothing but, x k inverse x and given y it is easy to transform it back to the original space using x k y equal to x transformation.

(Refer Slide Time: 31:14)



Affine Scaling Algorithm:

- Start with any interior point $\boldsymbol{x}^0$
- while (stopping condition is not satisfied at the current point)
    - Transform the current problem into an equivalent problem in $\boldsymbol{y}-$space so that the current point is close to the centre of the feasible region
    - Use projected steepest descent direction to take a step in the $\boldsymbol{y}-$space without crossing the feasible set boundary
    - Map the new point back to the corresponding point in the $\boldsymbol{x}-$space

  endwhile

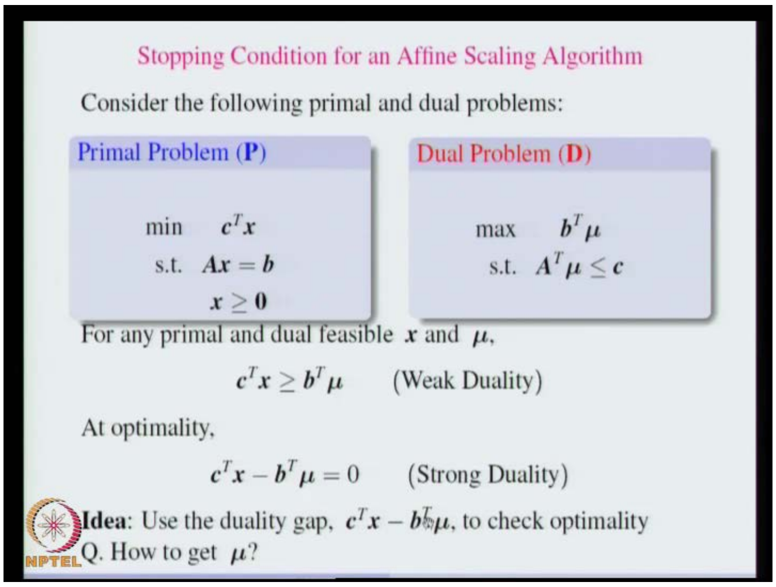
So now lets us look at a general idea of affine scaling algorithm. So start with any interior point x naught. Now while some stopping condition is not satisfied at the current point, the algorithm first transforms the current problem into an equivalent problem in y space. So note that this is very important that we should have an equivalent problem in the y space. We may not always have a linear program in the y space but, we should have a program in the y space or the problem in the y space which is in equivalent to the original problem. And this transformation should be such that the current point is close to the center of the feasible region.

We then use projected steepest descent direction in the y space to take a step in the y space, so that we do not cross the feasible set boundary. So this is very important that the y k that we get, y k plus 1 that we get from the current y k should be feasible, so therefore, the step should be taken such that they do not cross the feasible set boundary.

Now once we find y k plus 1, we need to find out what is x k plus 1, so we need to map that point y k plus 1 back to that original space. So map the third step is to map the new point back to the corresponding point in the x space. Now this procedure is repeated till stopping condition is satisfied.

So there are few important points that needs to be discussed. The first 1 is that what is the stopping condition for a for the program? Now in the simplex algorithm we saw that since the solution always lies at the extreme point, for a linear program we saw that if at a point all the neighboring vertices have an objective function value, which is not less than the current point then we stop. So that stopping criterion was easy to check. Now here since we are not talking about the vertices in the algorithm we need to find out some condition which could be satisfied at optimality. So that is the first point so we have already seen what is what could be a possible transformation to move the current point to a point in the y space which is close to the center of the feasible region.

(Refer Slide Time: 34:36)



Stopping Condition for an Affine Scaling Algorithm

Consider the following primal and dual problems:

Primal Problem (P)

$$\begin{aligned} \min \quad & \boldsymbol{c}^T\boldsymbol{x} \\ \text{s.t.} \quad & \boldsymbol{A}\boldsymbol{x} = \boldsymbol{b} \\ & \boldsymbol{x} \geq \boldsymbol{0} \end{aligned}$$

Dual Problem (D)

$$\begin{aligned} \max \quad & \boldsymbol{b}^T\boldsymbol{\mu} \\ \text{s.t.} \quad & \boldsymbol{A}^T\boldsymbol{\mu} \leq \boldsymbol{c} \end{aligned}$$

For any primal and dual feasible $\boldsymbol{x}$ and $\boldsymbol{\mu}$,

$$\boldsymbol{c}^T\boldsymbol{x} \geq \boldsymbol{b}^T\boldsymbol{\mu} \qquad \text{(Weak Duality)}$$

At optimality,

$$\boldsymbol{c}^T\boldsymbol{x} - \boldsymbol{b}^T\boldsymbol{\mu} = 0 \qquad \text{(Strong Duality)}$$

Idea: Use the duality gap, $\boldsymbol{c}^T\boldsymbol{x} - \boldsymbol{b}^T\boldsymbol{\mu}$, to check optimality

Q. How to get $\boldsymbol{\mu}$?

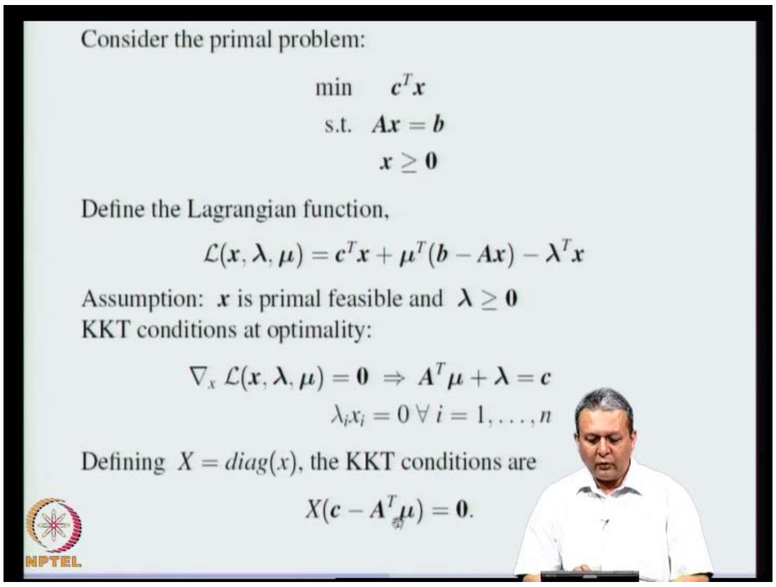
We also saw how to project steepest descent direction in the given space. So let us first see what is the good stopping condition for this algorithm. Now the duality ideas for the linear programming that we discussed earlier, they become very useful while designing many of the algorithms. So we will see 1 such condition which could be used as a stopping condition for an affine scaling algorithm.

So let us consider a linear standard programming form and the corresponding dual. Now we have seen that because of the weak duality, if x is primal feasible and mu is dual feasible c transpose x is greater than or equal to b transpose mu. So given x and mu which are primal and dual feasible we know that the optimal objective function value is at least the dual objective function value.

And moreover at optimality, the there exists x and mu such that c transpose x is equal to b transpose mu at optimality. As I mentioned earlier that this is also called gap the difference between the primal objective function value and dual objective function value where the primal is of the form minimize the objective function and dual is the form maximize corresponding objective function. That duality gap represents the difference between the primal and dual objective function value at feasible x and mu. So at optimality the dual gap is 0 and that could be used as a stopping criteria for our algorithm.

Now, that means that we need to find the value of mu and as we will see that the k k t conditions can be used to find the value of mu corresponding to the value of x. So let us see how to do that. So the weak duality ensures that the primal objective function value at any feasible primal point is at least the dual objective function value at corresponding dual point mu. And at optimality the 2 are equal and that is called strong duality. So the question so the idea is to the duality gap c transpose x minus b transpose mu, to check optimality and the question is how do we get mu. And for that purpose let us make use of the k k t conditions.

(Refer Slide Time: 37:11)



Consider the primal problem:

$$\begin{aligned} \min \quad & c^T x \\ \text{s.t.} \quad & Ax = b \\ & x \geq 0 \end{aligned}$$

Define the Lagrangian function,

$$\mathcal{L}(x, \lambda, \mu) = c^T x + \mu^T (b - Ax) - \lambda^T x$$

Assumption: x is primal feasible and $\lambda \geq 0$
KKT conditions at optimality:

$$\nabla_x \mathcal{L}(x, \lambda, \mu) = 0 \Rightarrow A^T \mu + \lambda = c$$
$$\lambda_i x_i = 0 \ \forall\, i = 1, \ldots, n$$

Defining $X = diag(x)$, the KKT conditions are

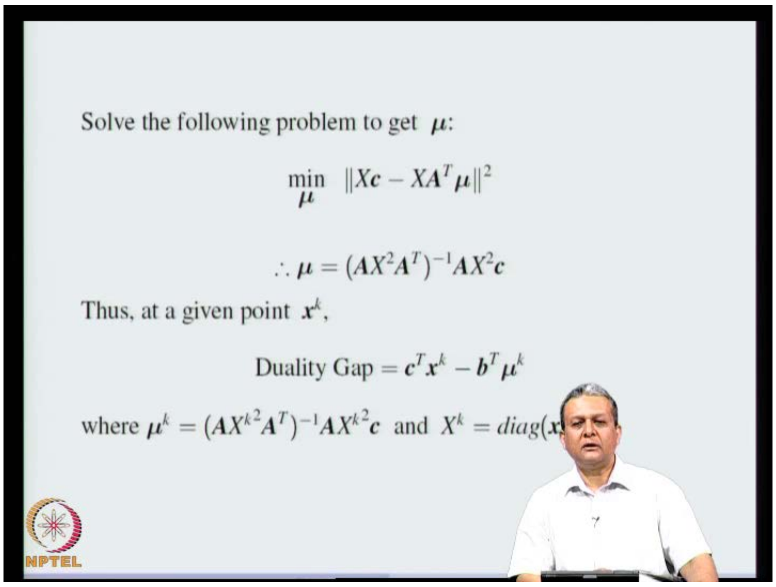
$$X(c - A^T \mu) = 0.$$

So let us consider this standard linear program and define the Lagrangian to be c transpose x plus mu transpose b minus A x minus lambda transpose A x. Now let us assume that x is primal feasible and lambda is non-negative. Now if we write down the k k t conditions at optimality 1 of the conditions is the gradient of the Lagrangian with respect to the primal variables is 0. So gradient of the Lagrangian with respect to x is 0.

So if you take the gradient of this Lagrangian function it will be c minus A transpose mu minus lambda. So that will be equal to 0, so therefore, A transpose mu plus lambda equal to c. So this condition needs to be satisfied at optimality. Now we have 2 variables mu and lambda which need to be determined. But, the complimentary slackness conditions becomes very useful at this moment. So the complimentary slackness conditions say that lambda i xi should be equal to 0 for all i equal one to n. So which means that if a particular xi is greater than 0 then the corresponding lambda has to be equal to 0. And therefore, we can combine this 2 conditions together to write that by defining a matrix x which is a diagonal of x.

So we take a vector x and put its components along the diagonal matrix x and form a matrix x then the k k t conditions can be written as x into c minus A transpose mu should b e equal to 0. So if x is greater than 0 then the corresponding lambda has to be 0 because of the complimentary slackness condition and that lambda is nothing but, the component of c minus A transpose mu. And if x equal to 0 we have xi lambda i to be 0 so x into xi into the corresponding component of lambda is 0. So this condition needs to be satisfied at optimality. Now that condition can be used to determine the value of mu. So what we need to do is that we need to minimize remember that the minimum quantity of this expression is on the left side is 0. So we need to minimize x c minus x A transpose mu subject to mu so as to get the current value of mu.

(Refer Slide Time: 40:23)



Solve the following problem to get μ:

$$\min_{\mu} \quad \|Xc - XA^T\mu\|^2$$

$$\therefore \mu = (AX^2A^T)^{-1}AX^2c$$

Thus, at a given point x^k,

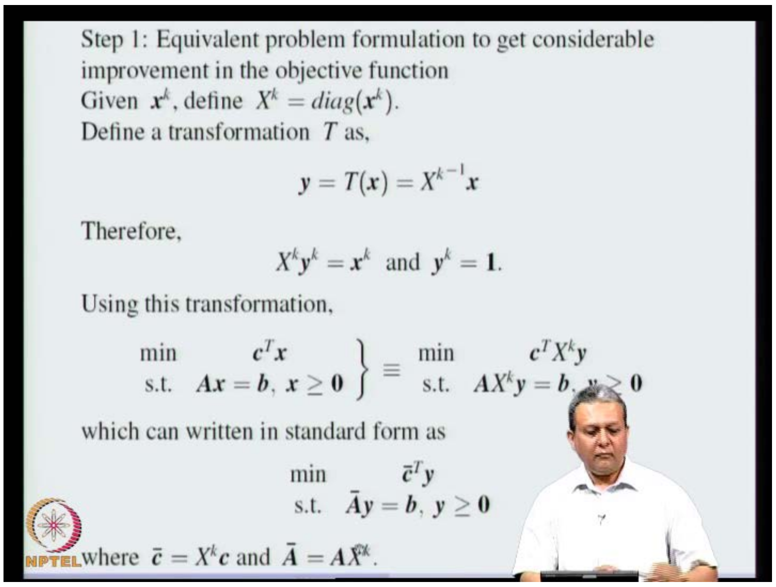
$$\text{Duality Gap} = c^Tx^k - b^T\mu^k$$

where $\mu^k = (AX^{k^2}A^T)^{-1}AX^{k^2}c$ and $X^k = diag(x$

So we minimize the norm of norm square of x c minus x A transpose mu. Now if we differentiate so before differentiating let us note that mu is unrestricted in sign. So this becomes an unconstraint optimization problem. So we have a convex function which is differentiable and differentiating with respect to mu what we get is A x transpose inverse A x square equal to x square c. So this is the value of mu, note that the current point x is part of matrix because the matrix x was found using the current variable x. So given x we first form a matrix x and then use this formula to find out mu. And therefore, the duality gap at x square will be c transpose x k minus b transpose mu k where mu k is obtained using x k and x k is obtained using the current iteration vector x k.

So the matrix x k is obtained using the current iteration vector x k. So mu k is nothing but, A x k square inverse A x square c where x square is the diagonal of the matrix x k. So x k can be used to find out mu k and then c transpose x k minus b transpose mu k gives us the duality gap. And therefore, the stopping condition for an algorithm could be that c transpose x k minus b transpose mu k is greater than 0. Because duality gap is always greater than or equal to 0 and the algorithm would terminate when the duality gap is equal to 0.

(Refer Slide Time: 42:29)



Step 1: Equivalent problem formulation to get considerable improvement in the objective function

Given $\boldsymbol{x}^k$, define $X^k = diag(\boldsymbol{x}^k)$.

Define a transformation T as,

$$\boldsymbol{y} = T(\boldsymbol{x}) = X^{k^{-1}}\boldsymbol{x}$$

Therefore,

$$X^k\boldsymbol{y}^k = \boldsymbol{x}^k \text{ and } \boldsymbol{y}^k = \boldsymbol{1}.$$

Using this transformation,

$$\left.\begin{array}{ll} \min & \boldsymbol{c}^T\boldsymbol{x} \\ \text{s.t.} & A\boldsymbol{x} = \boldsymbol{b},\ \boldsymbol{x} \geq \boldsymbol{0} \end{array}\right\} \equiv \begin{array}{ll} \min & \boldsymbol{c}^T X^k \boldsymbol{y} \\ \text{s.t.} & AX^k\boldsymbol{y} = \boldsymbol{b},\ \boldsymbol{y} \geq \boldsymbol{0} \end{array}$$

which can written in standard form as

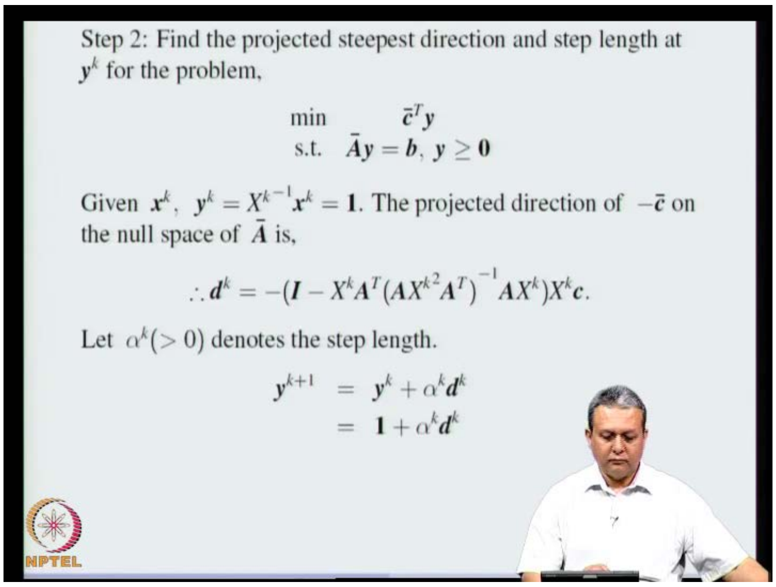
$$\begin{array}{ll} \min & \bar{\boldsymbol{c}}^T\boldsymbol{y} \\ \text{s.t.} & \bar{A}\boldsymbol{y} = \boldsymbol{b},\ \boldsymbol{y} \geq \boldsymbol{0} \end{array}$$

where $\bar{\boldsymbol{c}} = X^k\boldsymbol{c}$ and $\bar{A} = AX^k$.

Now the next step is after studying the stopping condition the first step that we saw in the algorithm was to get an equivalent problem formulation to consider to get considerable improvement in the objective function. So given x k we define the matrix x k to be the diagonal matrix having elements x k on its diagonal and we define the transformation y equal to x as x k inverse x. And therefore, if we use this transformation then what happens to the linear program that we get. Or that we wanted to solve.

So in the y space this program becomes minimize c transpose x k y because x is nothing but, x k y. So minimize c transpose x k y subject to A x k y equal to b and y greater than or equal to 0. So this is out new linear program. Remember this is a linear program because the objective function the constraints are all linear in terms of the variable y. So the variable is now y and this is still in the standard from because we can combine c transpose x k to c bar transpose and A x k to be A bar. And therefore, it still remains a program in the standard form where c bar is x k c and A bar is equal to x k.

(Refer Slide Time: 44:56)


Step 2: Find the projected steepest direction and step length at y^k for the problem,

$$\begin{array}{ll} \min & \bar{c}^T y \\ \text{s.t.} & \bar{A}y = b,\ y \geq 0 \end{array}$$

Given x^k, $y^k = X^{k^{-1}} x^k = \mathbf{1}$. The projected direction of $-\bar{c}$ on the null space of $\bar{A}$ is,

$$\therefore d^k = -(I - X^k A^T (A X^{k^2} A^T)^{-1} A X^k) X^k c.$$

Let $\alpha^k (> 0)$ denotes the step length.

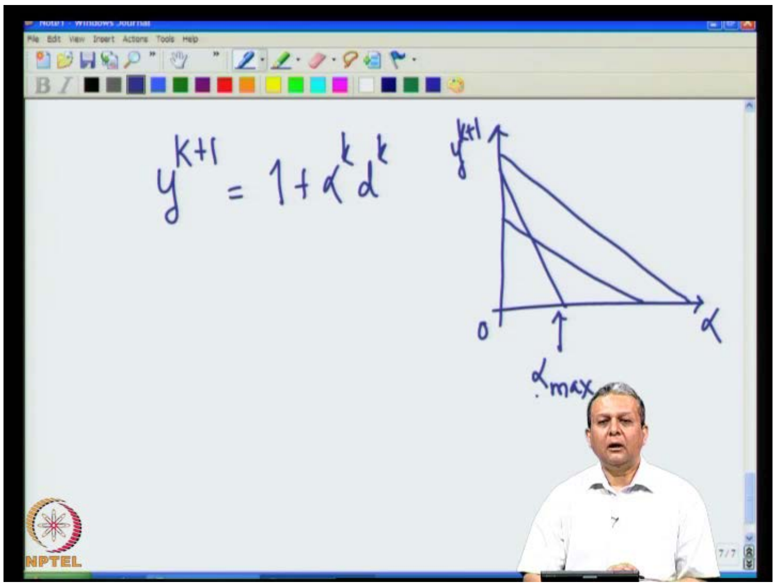
$$\begin{aligned} y^{k+1} &= y^k + \alpha^k d^k \\ &= \mathbf{1} + \alpha^k d^k \end{aligned}$$

So by using this transformation we still retain the program as a linear program. But, now the descent direction will be negative x k c bar, which is nothing but, minus x k c. So that will be the descent direction that will be used and the matrix which was in the original space will now correspond to a matrix A x k in the y space. So what we would be interested in finding out the negative of x k c direction on the null space of matrix A bar where A bar is nothing but, A x k. So the projected steepest descent direction and the determination of step length is the next step of the algorithm. And let us recall that in the y space we work with variables we work with the variable y with the constant vector c bar and A bar and b which are given to us.

So we have this transformation given x k y k is nothing but, x k inverse x k and that is nothing but, the unit vector in the y space. And the projected direction of the negative of the c bar on the null space of A bar, by using the same ideas that we discussed earlier. We can write it as A d k to be the negative of the projection matrix into x k c. So this will be the direction along which one can make a movement by retaining feasibility. But, then how much progress can be made or how much movement can be made along the direction d k so that we do not cross the feasible set boundary. So that is the next question that we would like to answer. Now if alpha k which is the positive quantity denotes the step length then y k plus 1 should be equal to y k plus alpha k d k.

Now this note that we want y k plus 1 to be in the interior of the feasible region. So that means that y k plus 1 should be strictly greater than 0. Now y k which is the current point is greater than 0, so what we want is that y k plus alpha k d k should be greater than 0 or 1 plus alpha k d k should be greater than 0.

(Refer Slide Time: 47:21)

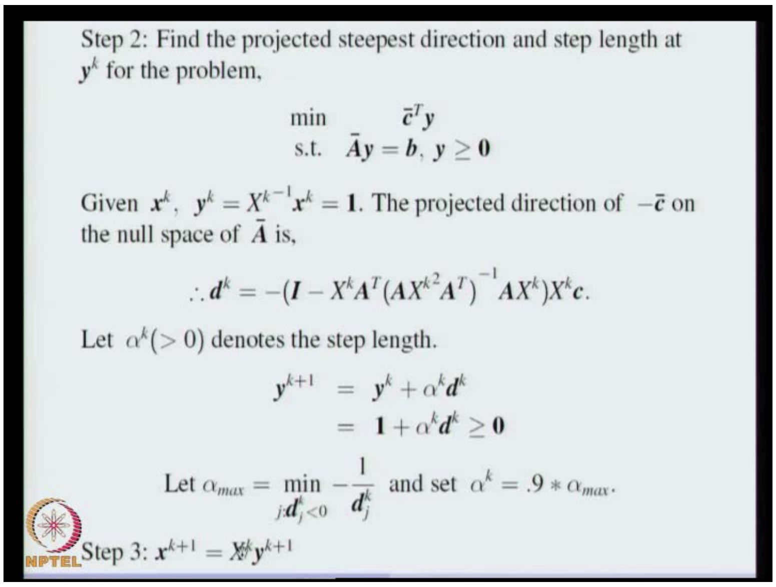


$$y^{k+1} = 1 + \alpha^k d^k$$

Now we have y k plus is equal to 1 plus alpha k d k. So on the horizontal axis we have alpha and on the vertical axis we have y k plus 1. Now y k plus 1 is of the same dimension x which is m. Now y k plus 1 will have different n components and the different components vary in a different way based on d k. Now some components could vary like this some components could vary like this and some component could vary like this.

So we will see that at this point a particular component of y will becomes 0. And then it will go to a negative quantity as alpha is increased further. So we check the different values of alpha’s at which different components of y k plus 1 go to 0. And among them we choose the least 1 and this we will call it as alpha max. So the maximum step length could be alpha max and that will correspond to taking 1 of the components to 0. And we do not want to take a point which is on the boundary so alpha k will be chosen as some multiple of alpha max. So that all the components of y will remain positive at that value of alpha. So typically 0.9 is a good choice for that.

(Refer Slide Time: 49:56)



Step 2: Find the projected steepest direction and step length at y^k for the problem,

$$\begin{array}{ll} \min & \bar{c}^T y \\ \text{s.t.} & \bar{A}y = b,\ y \geq 0 \end{array}$$

Given x^k, $y^k = X^{k^{-1}} x^k = 1$. The projected direction of $-\bar{c}$ on the null space of $\bar{A}$ is,

$$\therefore d^k = -(I - X^k A^T (A X^{k^2} A^T)^{-1} A X^k) X^k c.$$

Let $\alpha^k (> 0)$ denotes the step length.

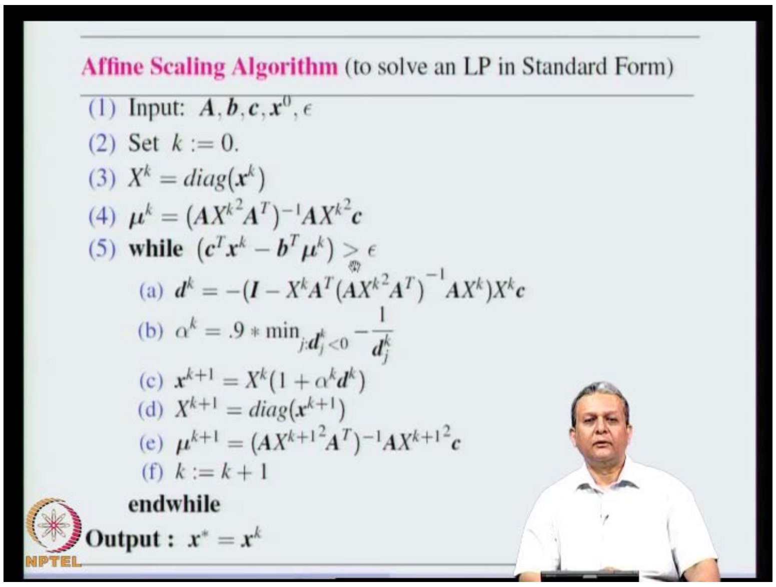
$$\begin{aligned} y^{k+1} &= y^k + \alpha^k d^k \\ &= 1 + \alpha^k d^k \geq 0 \end{aligned}$$

Let $\alpha_{max} = \min_{j: d^k_j < 0} -\frac{1}{d^k_j}$ and set $\alpha^k = .9 * \alpha_{max}$.

Step 3: $x^{k+1} = X^k y^{k+1}$

So 0.9 into alpha max could be a value of alpha k. So we want this to be greater than or equal to 0. And therefore, alpha max is chosen as minimum of minus over d k, d k j where d k j is less than 0. And this alpha k is set to 0.9 into alpha max. And the step 3 of the algorithm is to transform y k plus 1 which is obtained here back to the x space that is x k plus 1 is nothing but, x k y k plus 1.

(Refer Slide Time: 50:28)



Affine Scaling Algorithm (to solve an LP in Standard Form)

```
(1) Input: $A, b, c, x^0, \epsilon$
(2) Set $k := 0$.
(3) $X^k = diag(x^k)$
(4) $\mu^k = (A X^{k^2} A^T)^{-1} A X^{k^2} c$
(5) while $(c^T x^k - b^T \mu^k) > \epsilon$
      (a) $d^k = -(I - X^k A^T (A X^{k^2} A^T)^{-1} A X^k) X^k c$
      (b) $\alpha^k = .9 * \min_{j: d^k_j < 0} -\frac{1}{d^k_j}$
      (c) $x^{k+1} = X^k (1 + \alpha^k d^k)$
      (d) $X^{k+1} = diag(x^{k+1})$
      (e) $\mu^{k+1} = (A X^{{k+1}^2} A^T)^{-1} A X^{{k+1}^2} c$
      (f) $k := k + 1$
    endwhile
Output : $x^* = x^k$
```

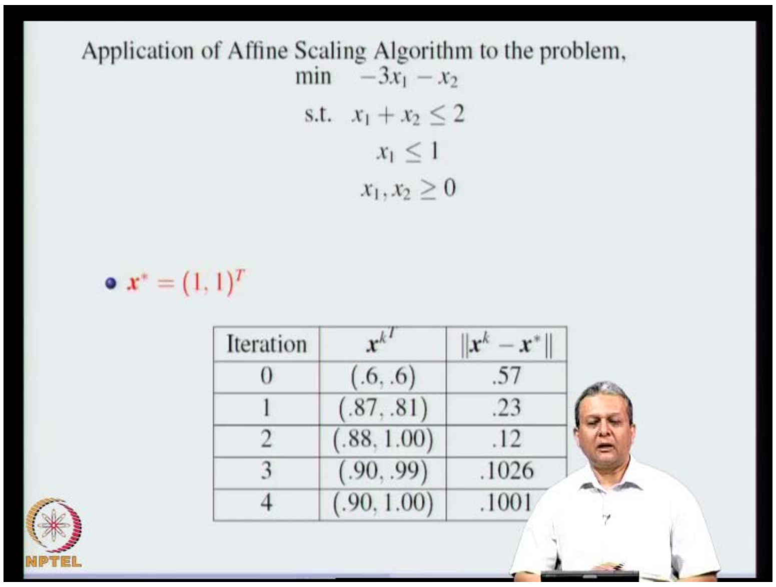
So now we have an affine scaling algorithm to solve an l p in standard form. I reiterate here this is not a polynomial time algorithm, but, it gives us an easy way of implementing some of the ideas that Karmarkar proposed in his projective scaling algorithm. Of course, the transformation that Karmarkar's algorithm uses is different and we will see that some time later but, right now we have used the simple transformation and that resulted in simple linear programming space.

And therefore, we studied the affine scaling algorithm first. So we are given linear program in standard form so the values of a b c are known, x 0 is initial point which is arbitrary in the interior of the feasible region, epsilon will be used for finding out the stopping condition. So the iteration counter is set to 0 the matrix x k is set using the diagonal the diagonal matrix x k is set based on the values of the corresponding components in the vector x k. The dual variable mu k is found using the formula that we saw earlier now the first step is that while stopping condition is not satisfied, so which means that while duality gap is greater than some epsilon where epsilon is a given parameter to project the negative steepest descent direction.

So project minus c bar on the null space of A bar and this is the expression that one gets and we saw this earlier. Now having done this projection we need to find out the step length that could be used. So the step length which one can use is 0.9 times the minimum at which one of the components becomes 0 or 0.9 into alpha max. So this is the alpha max that we have. And then transform back to the original space, so this space this step first does y k plus 1 is equal to y k plus alpha k d k and then x k y k or x k plus 1 is equal to the matrix x k into y k. So this is nothing but, y k 1 here indicates that it is y k is equal to 1 in the y space and that is added to alpha k d k,, d k is the negative of the steepest descent direction projected on the A bar matrix the null space of the A bar matrix.

And then we get ready for the next iteration, so the matrix x k plus 1 is a set to this quantity and mu k plus 1 found because that will be used to check the stopping condition the iteration counter is set to 1 and then we go back and check whether the stopping condition is satisfied. Finally, when the algorithm terminates we get a x star which is equal to the current value of x k and this point will be typically close to the vertex which is a solution point of the linear program. Because every at time we are ensuring that we do not cross the boundary, so we may not the algorithm path may not a generate a point which is vertex. But, we will get a point which is close to the vertex.

(Refer Slide Time: 54:43)



Application of Affine Scaling Algorithm to the problem,

$$\begin{aligned} \min \quad & -3x_1 - x_2 \\ \text{s.t.} \quad & x_1 + x_2 \leq 2 \\ & x_1 \leq 1 \\ & x_1, x_2 \geq 0 \end{aligned}$$

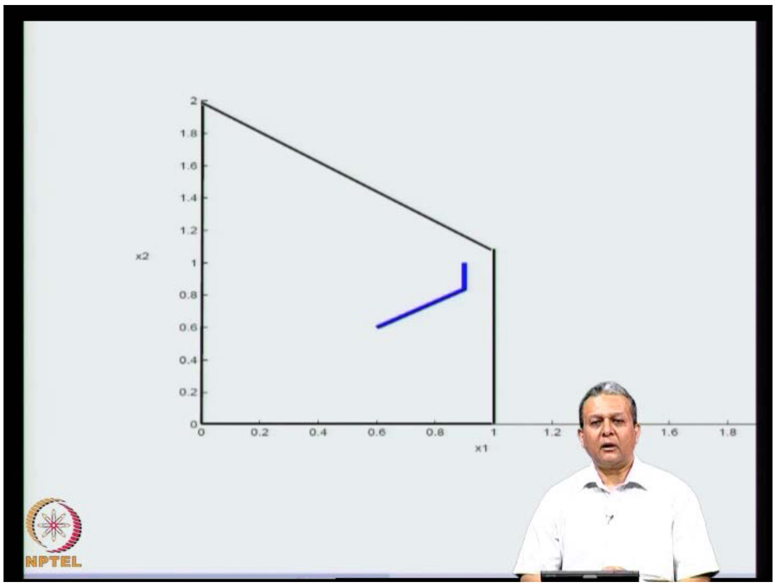
- $x^* = (1,1)^T$

Iteration	x^{k^T}	$\|x^k - x^*\|$
0	(.6, .6)	.57
1	(.87, .81)	.23
2	(.88, 1.00)	.12
3	(.90, .99)	.1026
4	(.90, 1.00)	.1001

So if we apply this affine scaling algorithm to this problem, we have already seen this problem when we discuss simplex method and we know the solution is 1 comma 1. So if we start with a 6.6, then here are the iterates generated using the affine scaling algorithm that we saw. And the last column denotes the distance between the current point and x star.

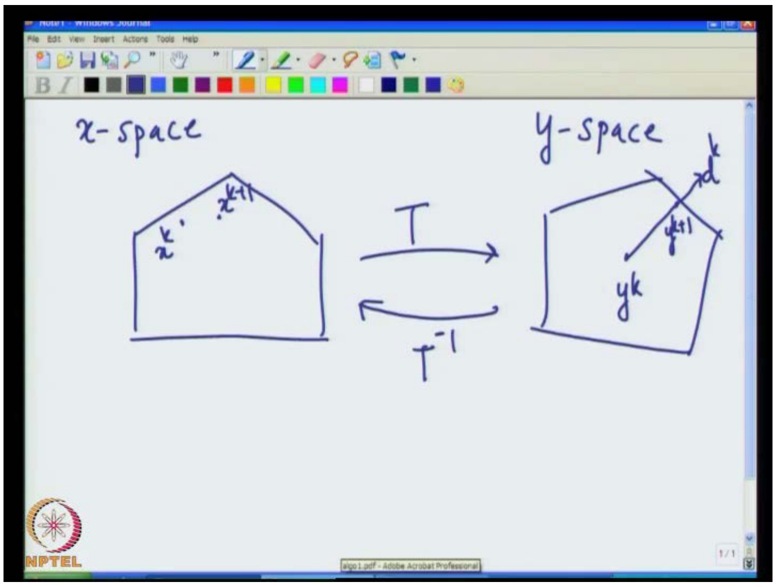
The utility and distance and you will see that the distance and current point from x star keeps coming down as the algorithm next progress. So after four iteration you will see that the distance between the 2 is only 0.10 or 0.10. So this gives us some idea about how the affine scaling algorithm works.

(Refer Slide Time: 55:44)



Now here is the plot, so this quadrilateral denotes the feasible region and we start from a point which is x 1 and x 2 to be 0.6 and this is the solution point, which is 1 comma 1. And this is how the algorithm makes a progress after four iterations. So this gives us some idea about how an affine scaling algorithm works.

(Refer Slide Time: 56:38)

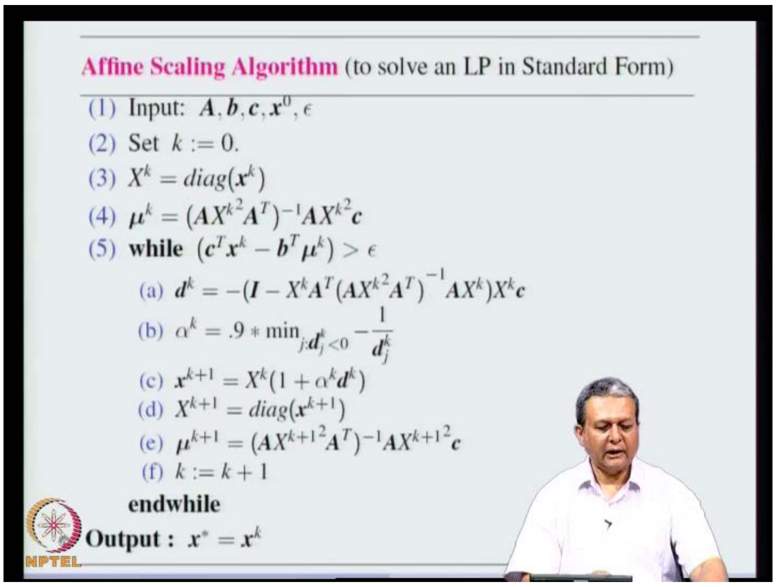


Now we will see the different transformation we used by Karmarkar in his algorithm to solve linear program. And also the form of the linear program which is needed for Karmarkar’s method. In the last class we discussed about affine scaling algorithm. So the idea was that in the x space we had this constraint set and the idea is to use some transformation t to transform this constraint set into y space.

So that the point gets mapped to a point y k which is y k which is close to the center of the feasible set. So the reason for this is that if a current point is close to the center of the feasible set then a considerable improvement in the objective function is possible. So that is the idea behind this transformation. So when moves along the steepest direction so this is along the projected steepest direction, so 1 moves till 1 hits the boundary of the feasible region.

Now once this point so this is the direction d k and the step length is chosen such that one does not cross the boundary, so before hitting the boundary of the feasible set 1 finds out an appropriate step length so that a new point y k plus 1 is found. And then that point is mapped back to using the transformation t inverse. So we need a invertible transformation and that new point could be a corresponding point x k plus 1 in the input space. And again once x k plus defines a new transformation t so this transformation t does depend upon the current point. So it is not a constant transformation it is not a transformation which is independent of x. So this was the main idea of affine scaling algorithm and the reason why we studied this algorithm first was that it is simple and easy to understand.

(Refer Slide Time: 59:08)



Affine Scaling Algorithm (to solve an LP in Standard Form)

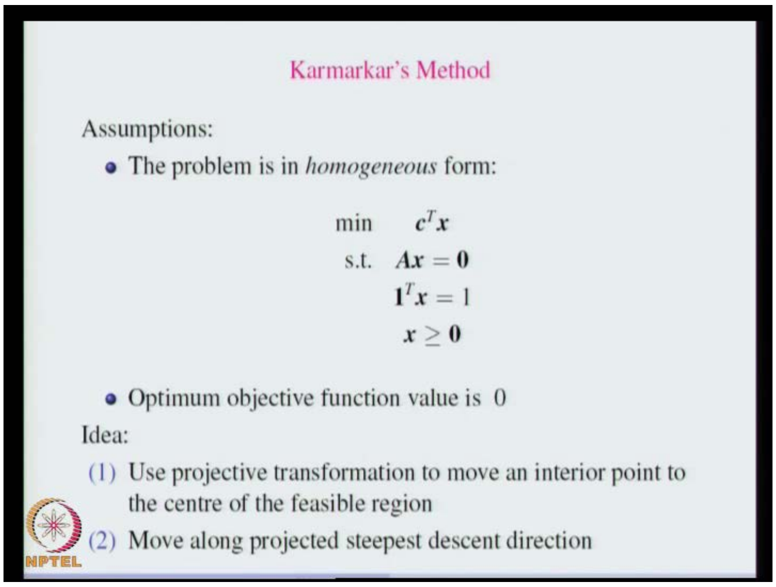
```
(1) Input: $\boldsymbol{A}, \boldsymbol{b}, \boldsymbol{c}, \boldsymbol{x}^0, \epsilon$
(2) Set $k := 0$.
(3) $X^k = diag(\boldsymbol{x}^k)$
(4) $\boldsymbol{\mu}^k = (\boldsymbol{A}{X^k}^2\boldsymbol{A}^T)^{-1}\boldsymbol{A}{X^k}^2\boldsymbol{c}$
(5) while $(\boldsymbol{c}^T\boldsymbol{x}^k - \boldsymbol{b}^T\boldsymbol{\mu}^k) > \epsilon$
      (a) $\boldsymbol{d}^k = -(\boldsymbol{I} - X^k\boldsymbol{A}^T(\boldsymbol{A}{X^k}^2\boldsymbol{A}^T)^{-1}\boldsymbol{A}X^k)X^k\boldsymbol{c}$
      (b) $\alpha^k = .9 * \min_{j:\boldsymbol{d}_j^k<0} -\frac{1}{\boldsymbol{d}_j^k}$
      (c) $\boldsymbol{x}^{k+1} = X^k(1 + \alpha^k\boldsymbol{d}^k)$
      (d) $X^{k+1} = diag(\boldsymbol{x}^{k+1})$
      (e) $\boldsymbol{\mu}^{k+1} = (\boldsymbol{A}{X^{k+1}}^2\boldsymbol{A}^T)^{-1}\boldsymbol{A}{X^{k+1}}^2\boldsymbol{c}$
      (f) $k := k + 1$
    endwhile
Output : $\boldsymbol{x}^* = \boldsymbol{x}^k$
```

So the main idea was that while the duality gap is not small to find the direction d k which is the projected steepest descent direction projection of steepest descent direction of null space of the matrix A. So since the problem is transformed to y space 1 has to project the x k c on the nu space of the appropriate matrix. Find the step length so that 1 does not cross the boundary then get x k plus 1 using the value of alpha k d k and current value of x k.

And find x k plus 1 the matrix x k plus 1 using the value x k plus 1 by putting the entries of the vector in the diagonal of the matrix. Find mu k plus 1 to find out the duality gap for the next iteration setting k equal to k plus 1. And going back to the step previous step where we check whether the duality gap is small enough. So this is the idea of affine scaling algorithm and we saw in the last class that the point generated are in the interior and this was the plot for the four iterations of affine scaling algorithm.

Now with this background we move on to Karmarkar's algorithm. Now Karmarkar's algorithm or Karmarkar's method makes certain assumptions about linear program. So let us first see those assumptions and then study some of the important steps that were used by Karmarkar's method.

(Refer Slide Time: 01:01:36)



Karmarkar's Method

Assumptions:

- The problem is in *homogeneous* form:

$$\begin{aligned} \min \quad & \boldsymbol{c}^T\boldsymbol{x} \\ \text{s.t.} \quad & \boldsymbol{Ax} = \boldsymbol{0} \\ & \boldsymbol{1}^T\boldsymbol{x} = 1 \\ & \boldsymbol{x} \geq \boldsymbol{0} \end{aligned}$$

- Optimum objective function value is 0

Idea:

(1) Use projective transformation to move an interior point to the centre of the feasible region

(2) Move along projected steepest descent direction

Again I repeat that Karmarkar's algorithm was developed first and affine scale algorithm was developed as a modification of Karmarkar's algorithm. Karmarkar's algorithm is a polynomial time algorithm to solve a linear program while affine scaling algorithm is not a polynomial time algorithm to solve a linear program.

But nevertheless it was simple to understand affine scaling algorithm before moving on to Karmarkar's method. So the first assumption that Karmarkar's method makes is that the linear program is in homogenous form. So you will see that in the constraints there is no b here involved in the right side of system of equations. So we have the linear objective function to be minimized subject to A x equal to 0. And this 1 is a vector of all one's. So the sum of all elements in x is 1 and x is non-negative.

So this is not a serious this is not a assumption which will be violated at every point of time. On the other hand there are ways to convert linear programs in standard form to the homogenous form. So we will continue to use this homogenous form when we talk about Karmarkar's method. It is not a serious drawback of Karmarkar's algorithm. Now the second assumption is that the optimum objective function value is 0.

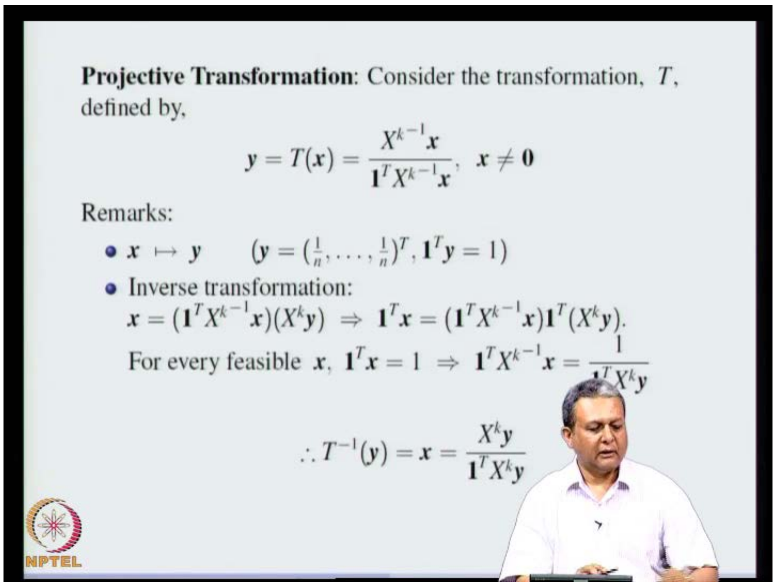
Now if one has some knowledge about duality and dual functions then one can appropriately choose the dual function so that the duality so that the optimum objective function value can be assumed to be 0. So these are not serious drawbacks of Karmarkar's algorithm but, these assumptions make the algorithm simple. Now as is the case in the case of affine scaling method the idea here is to use some transformation to move an interior point of the feasible set to the center of the feasible region..

So as was done in affine scaling here also one needs to use the projective transformation to move a point. Now in affine scaling we moved the input point to the point which contain all one's in the y space. In Karmarkar's method the point is moved to a point where in the n dimensional space every coordinate has a value m by n. And the then as was in the case of affine scaling we move along the steepest descent direction.

So finding a transformation to move an interior point to the center of feasible region and moving along the steepest descent direction are two key aspects of Karmarkar's method. And all these are done with respect to this problem formulation which is the linear program in homogenous form and under the assumption that optimum objective function value of this program is 0.

Now as I said that there are different ways to convert a linear program in standard form to this homogenous form, so this is not a serious drawback of Karmarkar's method.

(Refer Slide Time: 65:07)



Projective Transformation: Consider the transformation, T, defined by,

$$y = T(x) = \frac{X^{k^{-1}}x}{\mathbf{1}^T X^{k^{-1}}x}, \quad x \neq \mathbf{0}$$

Remarks:

- $x \mapsto y \qquad (y = (\frac{1}{n}, \ldots, \frac{1}{n})^T, \mathbf{1}^T y = 1)$
- Inverse transformation:
  $x = (\mathbf{1}^T X^{k^{-1}}x)(X^k y) \Rightarrow \mathbf{1}^T x = (\mathbf{1}^T X^{k^{-1}}x)\mathbf{1}^T(X^k y).$
  For every feasible x, $\mathbf{1}^T x = 1 \Rightarrow \mathbf{1}^T X^{k^{-1}}x = \frac{1}{\mathbf{1}^T X^k y}$

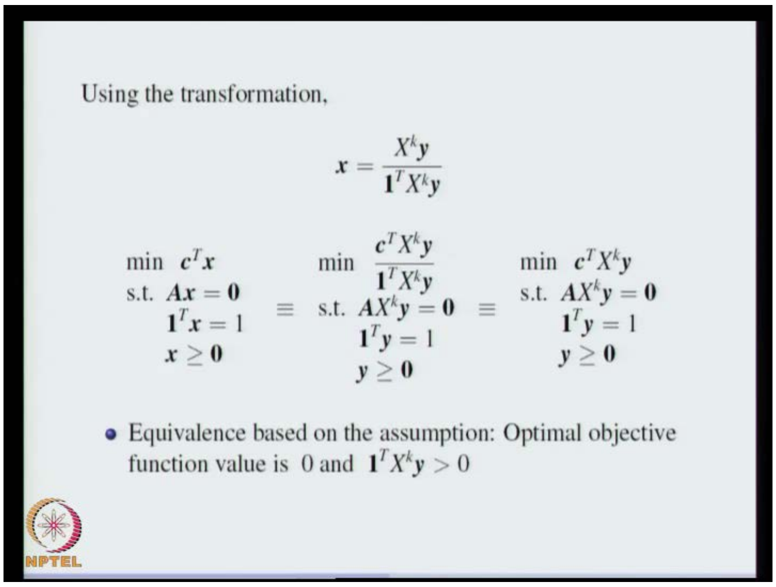
$$\therefore T^{-1}(y) = x = \frac{X^k y}{\mathbf{1}^T X^k y}$$

Now let us look at the projective transformation used to in Karmarkar's method. Now let us consider the transformation t which is defined by t x equal to x k inverse x by 1 transpose x k inverse x. So x k is matrix is a diagonal matrix obtained by putting elements of the vector x k along the diagonal of matrix x k. And of course, we are assuming that x is not equal to 0. So suppose we consider this transformation then you will see that 1 transpose y where 1 is a vector of all one's. So 1 transpose y will be nothing but, 1 transpose x k inverse x by 1 by 1 transpose x k inverse x and that is equal to 1. So any point x gets mapped to the center of the feasible region in the y space. So we consider a mapping from x to y, we will see that y has a coordinates 1 over n for all n dimensions and 1 transpose y equal to one.

So this transformation lets us to map x to y such that y is the center of the feasible region. Now it is easy to derive a inverse transformation for this, now from this expression we can write x to b. So if you multiply throughout by x k and multiply this equation by 1 transpose x k inverse x so what we get is x equal to 1 transpose x k inverse into x k y. Now we want to write x in terms of y, so we have to get rid of this x from this expression.

And write it in terms of y so let us see how to do that now 1 transpose x will be nothing but, 1 transpose x k inverse x into x k y. Now 1 transpose x we have assumed that 1 transpose x equal to 1 because that is 1 of the requirements of homogenous linear program. So 1 transpose x equal to 1 means that 1 transpose x k inverse x is equal to 1 over x transpose y. And therefore, we are able to write 1 transpose x inverse x in terms of 1 transpose x k y, so substitute this value here and what we get is t inverse y will be x equal to x k y this quantity and 1 transpose x k inverse x will b 1 over 1 transpose x k y, so that quantity is in the denominator. So we are able to get a inverse transformation.

(Refer Slide Time: 68:23)



Using the transformation,

$$x = \frac{X^k y}{\mathbf{1}^T X^k y}$$

$$\begin{array}{l} \min\ c^T x \\ \text{s.t. } Ax = 0 \\ \quad \mathbf{1}^T x = 1 \\ \quad x \geq 0 \end{array} \equiv \begin{array}{l} \min\ \dfrac{c^T X^k y}{\mathbf{1}^T X^k y} \\ \text{s.t. } AX^k y = 0 \\ \quad \mathbf{1}^T y = 1 \\ \quad y \geq 0 \end{array} \equiv \begin{array}{l} \min\ c^T X^k y \\ \text{s.t. } AX^k y = 0 \\ \quad \mathbf{1}^T y = 1 \\ \quad y \geq 0 \end{array}$$

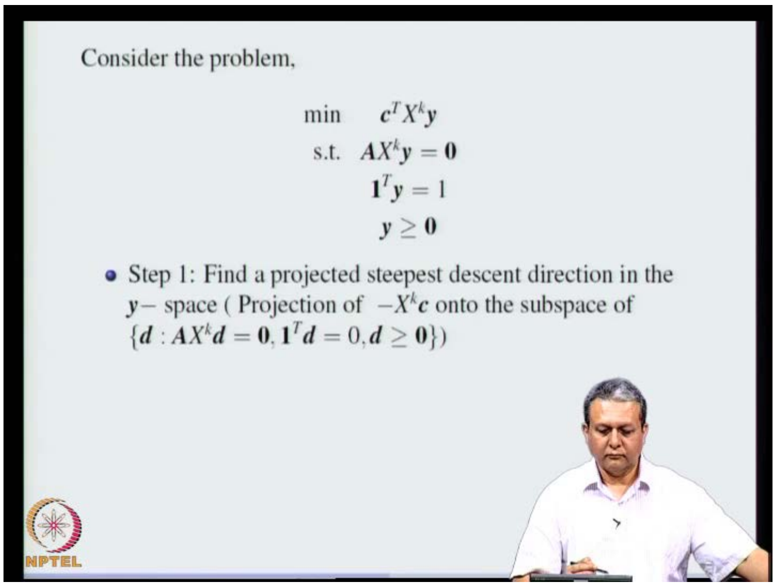
- Equivalence based on the assumption: Optimal objective function value is 0 and $\mathbf{1}^T X^k y > 0$

So one can check that this transformation t is 1 invertible transformation. Now if we use this transformation then the original linear program which was in homogenous form now substitute x here x as x k y by 1 transpose y in this homogenous linear program. So we get a linear program in y space and that program would look like this where x is replaced by x k y by 1 transpose x k y. So the objective function this value is replaced a x equal to 0 will become converted to A x k y equal to 0.

And as we saw any point x is converted to y such that 1 transpose y is equal to 1 and y's are greater than or equal to 0. Now for a movement it may appear that a simple linear program has been converted to a program which is not linear for example, the objective function itself is a fraction of 2 linear terms. So it is not a linear program and therefore, it will be difficult to solve this but, if you look at the denominator, note that x k was greater than 0. So all the entries in this matrix are positive, y is also point which is in the center so y is also positive.

So 1 transpose x k y is a positive quantity and not only that it is also bounded. So this program can be equivalently written in the form minimize c transpose x k y subject to the same conditions. So we have gotten rid of the denominator here because the denominator here is a positive quantity and bounded quantity and therefore, minimization of this fractional form of objective function is equivalent to the numerator since the denominator is positive and bounded. The other constraints remain the same So the original linear program which is in homogenous form is now converted to another linear program in the y space which is also in homogenous form. So now we have to solve this program using the steepest descent using the projected steepest descent method.

(Refer Slide Time: 71:41)



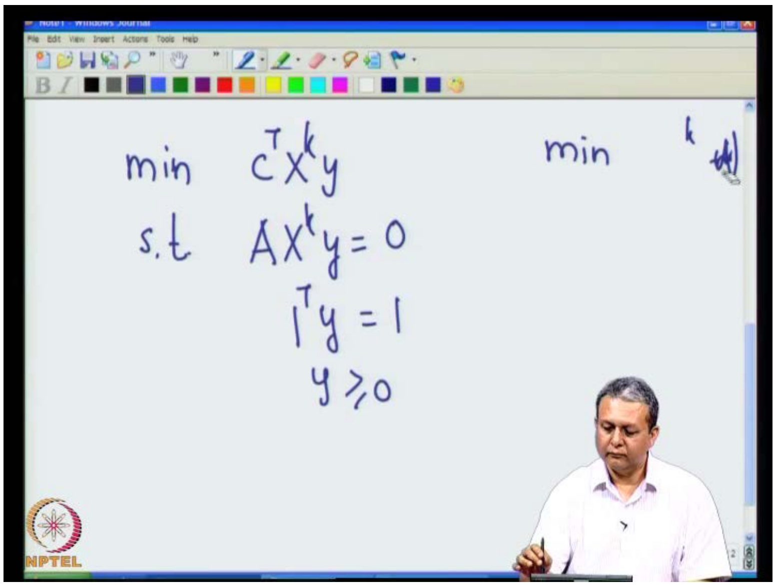
Consider the problem,

$$\begin{aligned} \min \quad & \mathbf{c}^T X^k \mathbf{y} \\ \text{s.t.} \quad & A X^k \mathbf{y} = \mathbf{0} \\ & \mathbf{1}^T \mathbf{y} = 1 \\ & \mathbf{y} \geq \mathbf{0} \end{aligned}$$

- Step 1: Find a projected steepest descent direction in the $\mathbf{y}-$ space (Projection of $-X^k\mathbf{c}$ onto the subspace of $\{\mathbf{d} : AX^k\mathbf{d} = \mathbf{0}, \mathbf{1}^T\mathbf{d} = 0, \mathbf{d} \geq \mathbf{0}\}$)

Now as I said that earlier that the equivalence of these 2 is possible because of the assumption that optimal objective function value is 0 and 1 transpose x k y is greater than 0. So we now see how to get a descent direction for this homogenous linear program in the y space. Now the gradient this objective function is x k c and the negative of the gradient is minus x k c’s,.

(Refer Slide Time: 72:52)

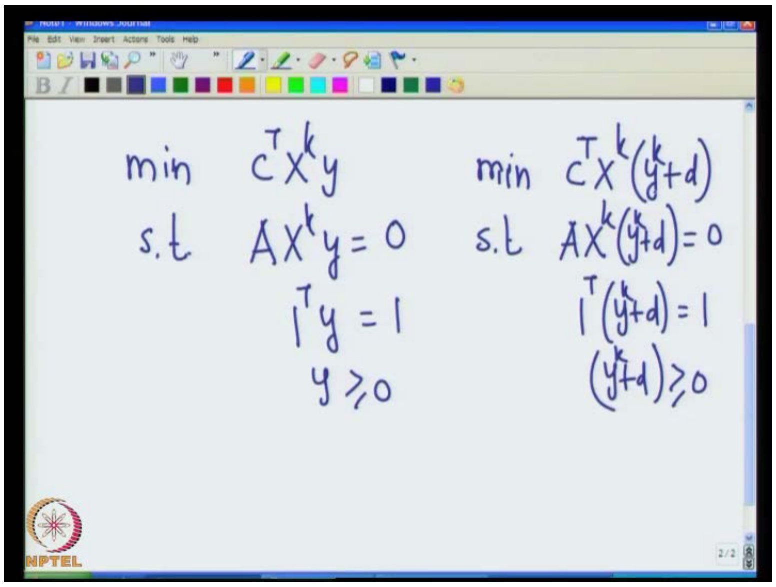


$$\begin{aligned} \min \quad & c^T X^k y \\ \text{s.t} \quad & A X^k y = 0 \\ & 1^T y = 1 \\ & y \geq 0 \end{aligned}$$

$\min$ [illegible]

So that is the steepest descent direction but, then we have to project that direction on to the null space of the matrix a x k and 1 transpose put together and we also have to make sure that d is greater than or equal to 0. So the first step is to get a projection of the steepest descent direction minus x k c on to the subspace where d satisfies a x k d equal to 0, 1 transpose d equal to 0 and d is non-negative. So it is easy to see this because… So suppose d is the direction that we are looking for along which we make a movement.

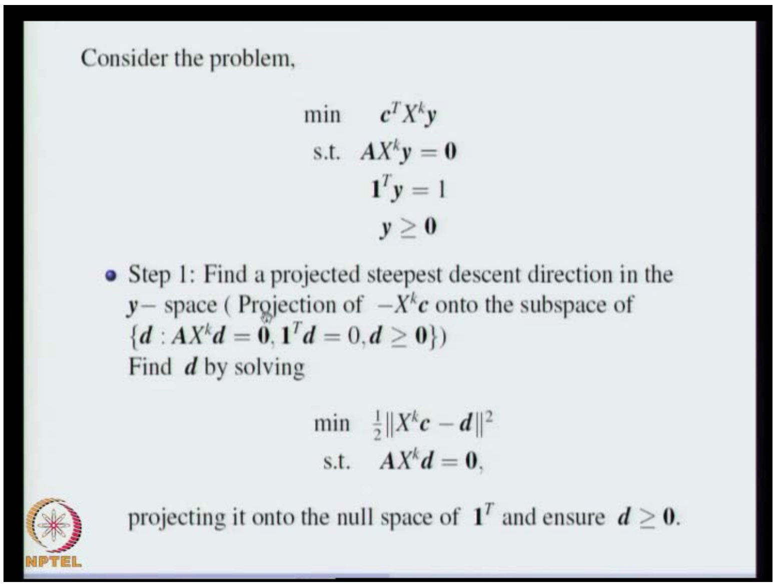
(Refer Slide Time: 73:54)



$$\begin{aligned} \min \quad & c^T X^k y \\ \text{s.t} \quad & A X^k y = 0 \\ & 1^T y = 1 \\ & y \geq 0 \end{aligned} \qquad \begin{aligned} \min \quad & c^T X^k (y^k + d) \\ \text{s.t} \quad & A X^k (y^k + d) = 0 \\ & 1^T (y^k + d) = 1 \\ & (y^k + d) \geq 0 \end{aligned}$$

So we write the problem in terms of d, so it will be minimize c transpose x k y plus d minimize c transpose x k y plus d subject to a x k y plus d equal to 0 1 transpose y plus d equal to 1 and y plus d greater than or equal to 0. Now c transpose x k y this quantity is already this should be y k, because y k is the current point. So c transpose x k y k is a constant quantity so we will be interested in minimizing c transpose x k d subject to A x d equal to 0 1 transpose d equal to 0 and d greater than or equal to 0.

(Refer Slide Time: 75:00)



Consider the problem,

$$\begin{aligned} \min \quad & \boldsymbol{c}^T X^k \boldsymbol{y} \\ \text{s.t.} \quad & \boldsymbol{A} X^k \boldsymbol{y} = \boldsymbol{0} \\ & \boldsymbol{1}^T \boldsymbol{y} = 1 \\ & \boldsymbol{y} \geq \boldsymbol{0} \end{aligned}$$

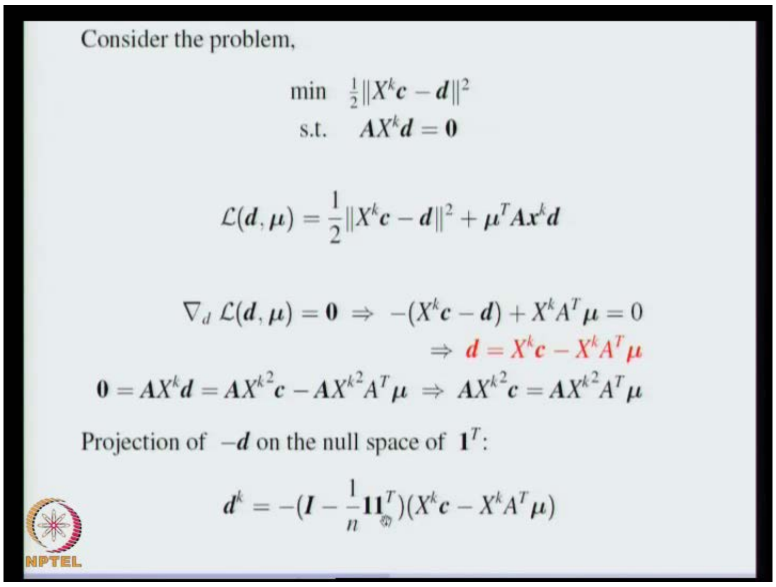
- Step 1: Find a projected steepest descent direction in the $\boldsymbol{y}-$ space (Projection of $-X^k\boldsymbol{c}$ onto the subspace of $\{\boldsymbol{d} : \boldsymbol{A}X^k\boldsymbol{d} = \boldsymbol{0}, \boldsymbol{1}^T\boldsymbol{d} = 0, \boldsymbol{d} \geq \boldsymbol{0}\}$)
  Find $\boldsymbol{d}$ by solving

$$\begin{aligned} \min \quad & \tfrac{1}{2}\|X^k\boldsymbol{c} - \boldsymbol{d}\|^2 \\ \text{s.t.} \quad & \boldsymbol{A}X^k\boldsymbol{d} = \boldsymbol{0}, \end{aligned}$$

projecting it onto the null space of $\boldsymbol{1}^T$ and ensure $\boldsymbol{d} \geq \boldsymbol{0}$.

Now so the way it is done is the following that we first solve d by minimizing or getting the projection of X k c on the null space of A X k and the projection of X k c on the null space of A X k is obtained by minimizing norm of X k c minus d square subject to the constraint that X k d equal to 0.

(Refer Slide Time: 75:59)



Consider the problem,

$$\begin{aligned} \min \quad & \tfrac{1}{2}\|X^k c - d\|^2 \\ \text{s.t.} \quad & AX^k d = \mathbf{0} \end{aligned}$$

$$\mathcal{L}(d, \mu) = \frac{1}{2}\|X^k c - d\|^2 + \mu^T A x^k d$$

$$\nabla_d \mathcal{L}(d, \mu) = \mathbf{0} \Rightarrow -(X^k c - d) + X^k A^T \mu = 0$$
$$\Rightarrow d = X^k c - X^k A^T \mu$$
$$\mathbf{0} = AX^k d = AX^{k^2} c - AX^{k^2} A^T \mu \Rightarrow AX^{k^2} c = AX^{k^2} A^T \mu$$

Projection of $-d$ on the null space of $\mathbf{1}^T$:

$$d^k = -(I - \frac{1}{n}\mathbf{1}\mathbf{1}^T)(X^k c - X^k A^T \mu)$$

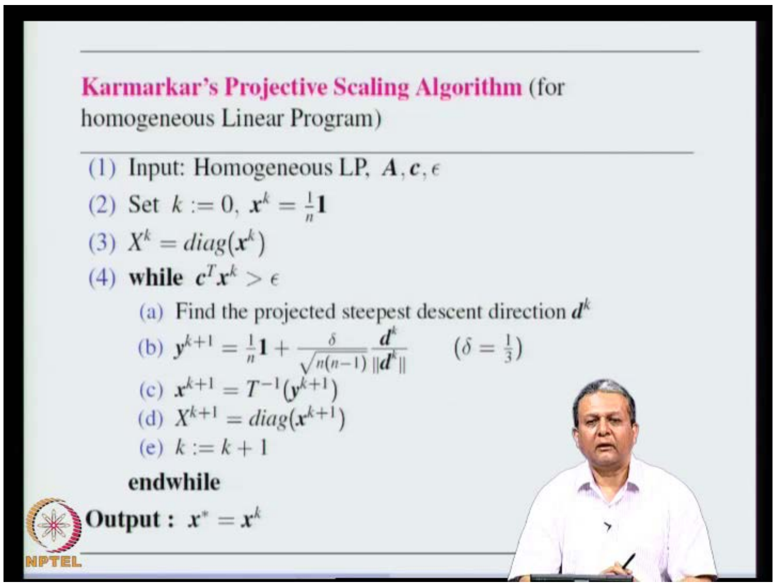
So we have taken care of the first part. Then after having obtained that d 1 projects it on the null space of the matrix 1 transpose and then using the algorithm we make sure that d is greater than or equal to 0. So if you do this then we will have found the projected steepest descent direction of minus X k c in the y space. So let us do this first..

Now if you consider the problem to minimize this norm of X k c minus t square to get the projection of X k c on the null space of A X k, so we write the Lagrangian, so the Lagrangian variables are the d the variables of the original problem as well as the Lagrangian multipliers associated with the equality constraint. So the objective function plus mu transpose A X k d that will be the Lagrangian. So getting the gradient of the Lagrangian with respect to d and setting it to 0 what we get is minus X k c minus d plus X k a transpose mu equal to 0. And this gives us d to be x k c minus x k a transpose mu.

Now remember that d and mu are the variables and we have d written in terms of mu. Now this d also should satisfy feasibility A X k d equal to 0. Now if you multiply this equation throughout by A X k then the left side becomes 0 and we will get mu in terms of A X k and c so multiplying throughout by A X k what we get is A X k square c minus A X square a transpose mu equal to 0. And this equation can be used to find mu so since A is a full rank full row rank matrix we can invert A X k square a transpose and multiply that post multiply by A X square c to get mu. And once we have this value mu we plug-in that value of mu here and we get the expression for d.

So finding d which solves this problem is easy. And then one has to get the projection of minus d on the null space of 1 transpose. So if you take the projection of minus on the null space of 1 transpose, so this is the projection matrix I minus 1 by n transpose where 1 is a vector of all one's into the direction X k c minus X k transpose mu where mu is obtained using this. So first we obtained mu plug-in that value of mu here and we get the projection of minus d on the null space of 1 transpose. Now here we will have to use those entries which are not negative. So this will be the direction along which we move so the entries which are not negative can be obtained using a simple algorithm.

(Refer Slide Time: 78:55)



Karmarkar's Projective Scaling Algorithm (for homogeneous Linear Program)

```
(1) Input: Homogeneous LP, $\boldsymbol{A}, \boldsymbol{c}, \epsilon$
(2) Set $k := 0$, $\boldsymbol{x}^k = \frac{1}{n}\mathbf{1}$
(3) $X^k = diag(\boldsymbol{x}^k)$
(4) while $\boldsymbol{c}^T\boldsymbol{x}^k > \epsilon$
        (a) Find the projected steepest descent direction $\boldsymbol{d}^k$
        (b) $\boldsymbol{y}^{k+1} = \frac{1}{n}\mathbf{1} + \frac{\delta}{\sqrt{n(n-1)}}\frac{\boldsymbol{d}^k}{\|\boldsymbol{d}^k\|}$    $(\delta = \frac{1}{3})$
        (c) $\boldsymbol{x}^{k+1} = T^{-1}(\boldsymbol{y}^{k+1})$
        (d) $X^{k+1} = diag(\boldsymbol{x}^{k+1})$
        (e) $k := k + 1$
    endwhile
Output : $\boldsymbol{x}^* = \boldsymbol{x}^k$
```

So, here is Karmarkar's projective scaling algorithm, where the input to the algorithm is a homogenous program and we have the matrix A which is a full row rank matrix vector c in the objective function and some epsilon used for stopping criteria. Remember that one of the assumptions that Karmarkar's algorithm used was that the minimal objective function value is 0. So we can use epsilon instead of 0 to check whether the optimality has been reached.

Of course, in the original algorithm the stopping criteria use was different but, just for understanding we have put c transpose x k greater than epsilon to be the stopping condition for Karmarkar's algorithm. One can look at original paper of Karmarkar to see the stopping criteria used by that algorithm. So the iteration counter is set to 0, initial point is taken to be the center of that feasible region the matrix x k is set. So while the optimum objective function value is not close to epsilon and finds the projected steepest direction d k.

And then makes the movement in the y space along that direction so this is the current point which is also 1 by n and then, so current point plus alpha k d k. So this is the normalized projected steepest descent direction d k by non d k and that is multiplied by alpha k. So karmarkar found that good choice for alpha k is 1 by 3 into root of n n into n minus 1. So one finds a new point y k plus 1 and then the next step is to project it back to the input space x space and that is done using the inverse transformation t inverse. And the new matrix x k plus 1 is found iteration counter is increased and 1 goes back to the new point and check whether optimality is satisfied. So this procedure is repeated till optimum value is satisfied or the procedure is repeated till the current point x k is very close to the optimal value. Because with this condition we cannot truly reach the exact optimal point but, we will be or the algorithm will be very close to the optimal point.

So when the algorithm terminates, we stay get the x star to be the current iteration value of x. So this is the rough idea about Karmarkar’s algorithm, which was the first algorithm which was polynomial time and also efficient than Kutchian’s polynomial time algorithm based on ellipsoid method. And you will see that the algorithm is simple and easy to implement. And more details about this algorithm like how to find the exact step size can be found in Karmarkar’s paper.

So with this we complete our discussion on interior point methods for linear programs. So in particular we saw affine scaling method and Karmarkar’s projective scaling method. The reason for studying interior point methods for linear program is that the simplex algorithm is not a exponential time algorithm is not a polynomial time algorithm.

It is a exponential time algorithm and therefore, there is a need to design algorithms for linear programs which are polynomial time algorithms. And interior point methods are one of the approaches which ensure that the algorithms for solving linear programs are polynomial time. In fact Karmarkar’s algorithm as a better complexity compared to Kutchian’s algorithm for solving linear programs.

Affine scaling algorithm which we studied is not a polynomial time algorithm but, nevertheless it is a simple algorithm and can be used for solving linear programs. So this completes our discussion on linear programs. We now move on to constraint optimization problems and study some of the algorithms which could be useful for solving constraint optimization problems.