Convex Optimization

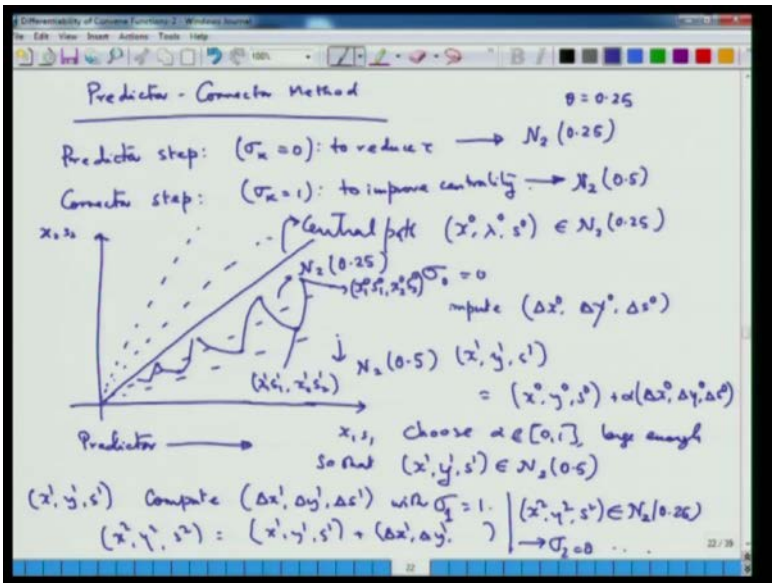
Prof. Joydeep Dutta

Department of Mathematics and Statistics

Indian Institute of Technology, Kanpur

Lecture No. # 33

(Refer Slide Time: 00:31)



So, today we will continue our discussion with the Predictor-Corrector method, which we ended which we ended yesterday's discussion, it is important to realize that you are not going to keep on writing down algorithms and doing the proofs to validate that they are correct, we are going to give you a brief conceptual idea of what a Predictor-Corrector method is, and then what are long step path following method is; and then of course, you can go ahead and read more the more references, and the real research papers or books, I I have already showed you this particular book, which is very, very important for your study here; and once, this book will have huge material, which you can study and enjoy.

So, what is a Predictor-Corrector step? I am I am actually telling it from this particular book is that you start with the… you have you have two neighborhoods here, the first neighborhood is linked to this particular choice of the centering parameter, and this is the N 2 (0.25) that is theta is equal to 0.25; the next neighborhood where sigma is equal to 1

is associated with the neighborhood 0.5 that is slightly larger neighborhood. So, what is happening is that I have two neighborhoods, so I start from a point, my if I start from a point (x naught,lambda naught,s naught) say in the N 2 (0.25) neighborhood, what I do is I then, I put sigma 0 is equal to 0, and then I compute (grad x naught, grad y naught grad)… That is the Newton steps of the for the relaxed intern case with this parameter - the centering parameter equal to 0.

So, now, what you do? Your next x 1 s 1 s 1 (x 1, y 1, s 1), how do you compute this? This is (x 0, y 0, s 0) plus alpha times you are doing this line search in the x s y space. Now observe that you now have to choose you what you do, you choose alpha, so large alpha element of 0, 1 large is large enough, so that x 1 (x 1, y 1, s 1) is in the larger neighborhood. So, you have straight a bit away from the central path; now you must be observing in these definitions, in this drawing, why I am giving a curved from; so, this is my starting point, this is this is my starting point (x naught,y naught,s naught) basically; this is my corresponding point x naught (x 1,y 1,s 1), but these are not x these are actually not those points.

So, this point here corresponds to the point x naught s naught in this particular case, it will correspond to the point (x naught 1,s naught 1) x naught 2 into s naught 2, maybe I will just write it in a better way. So, here this point is corresponding to (x naught,y naught,s naught) in the x y s space. So, when I am projecting on the x, s space, this point is actually corresponding to if there are two variables (x naught 1,s naught 1,x naught 2,s naught 2). And then this point which is (x 1,y 1,s 1) in the original space is corresponding to (x 1, s 1 1,x 1 2,s 1 2); now you might tell why this is curve, because why I am talking about moving along a particular direction from (x naught,y naught,s naught). So, I we should go in a straight line, but you have to remember that this space is not the x y s space, but the x s space. So, it is a projection of the thing in to the x s space and x 1 into s 1 by itself is a non-linear thing.

So, here I am look tracking the movement of x s, I am not tracking the I am the product x s, I am not tracking the movement in this space, I am not tracking the movement of this vector along the straight line. So, the projection of that gives the curve path in the x s space. So, what I have done? This step, what this is called the predictor step; this Predictor-Collector method actually follows closely in a that the idea is very close to the Predictor-Corrector method used for solving ordinary differential equations. So

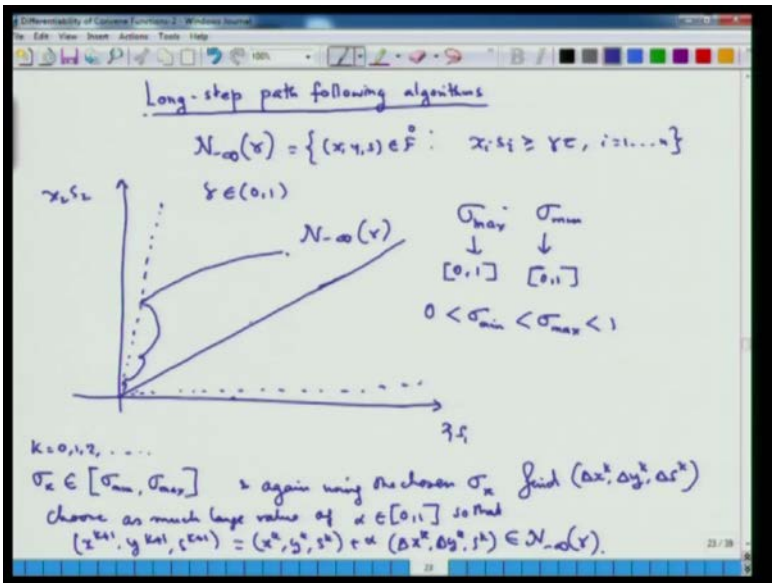
predictor, these are predictor part; now, once I have that (x 1,s 1 s 1,y 1,s 1), I would now compute (delta x 1,delta y 1,delta s 1) with sigma k equal to 1, where basically I will solve star - the star equation, the Newton equation, which you know, which I am not repeating with sigma k equal to 1.

Now, what you do is you do not choose… So, take if sigma equal to 1, so my centering parameter is strong, it is force on forcefully taking it towards the central path, the projection of the central path, these are central path in the x s space; interesting part is that here, I am looking at the central path at a straight line in x s space, when I had just two variables, so that is much more easier to explain things. So, I will now compute (x 2,y 2,s 2) I will take a unit step my alpha would be equal to 1; and I can actually mathematically prove, which will not do here, because of time constant, we cannot give so much detail.

This thing, this new (x 2,y 2,s 2), this (x 2,y 2,s 2) I can now show that it lies in a smaller neighborhood, so it is more near the central path. So, started from a point near the central path, come down to this and come here. So, I made a much more larger movement, then I can do possibly in the case of the short set path following method. Now you can mathematically prove that if I take a unit step from here to unique step along this direction, I will get a point, which will be there. Now, again in this step, when I am once I am in the smaller neighborhood, I will again choose k equal to sigma 2 equal to 0, and then again I will again keep on starting here sorry here my sigma k, k is 1. So, here I have done started with sigma 1 equal to 1.

Now, I will choose sigma 2 equal to 0, and generate the point (x 3,y 3,s 3), and then with the when that point is in the bigger higher higher neighborhood, here I have come to x 2 here I am in the (x 2,y 2,s 2) point, I come to x 3 s 3 point in the outside, and then I will choose again sigma you will start by choosing now, from here you will choose sigma 2 equal to 0, and proceed; you come to this point choose sigma 3 equal to 1 and proceed. So, in this way back and forth, back and forth, you will try, try keep on computing; obviously, at every step you will compute the tau, and the tau will basically, you will see its coming down. So, these are certain things, which you will not prove, but now we will go and discuss about the long step path following algorithm; obviously, it is these things are all can be proved to be following the polynomial time procedure.

(Refer Slide Time: 09:10)



Long step path following algorithm; so, here we will not go in to details, but try to explain to you.

(No audio from 09:17 to 09:34)

In the long step path following algorithms, I want to explain to you, how things and what we do; here as we have told before, we use the infinity neighborhood. And just to recall once again, the infinity neighborhood is given as follows that it consists of all elements of the form x y s in the strict neighborhood, in a strict feasible set; such that x i into s i is bigger than gamma times tau, where gamma is a quantity, where i is obviously from 1 to n, and gamma is a quantity, which is lying between 0 and 1, without 0 and 1, it does not take 0 or does not take the value 0 or 1. So, N minus infinity gamma is actually, if you look at again the x s space is much more easier to look at the x s space, because here I am only talking about the products. So, let me look at x 1 s 1 and x 2 s 2, standard two-dimensional weight would describe things; there are there are hardly in the once I look in to three dimension, it will become very difficult for me to describe this thing.

So, here again the central path is this, now what is the neighborhood in this case; how is is it a smaller neighborhood or bigger neighborhood? The answer is yes, it is a bigger neighborhood; why; what what is this x i s i is greater than equal to gamma into tau. So, if there is a particular tau x i s i, so suppose if you take x 1 s 1, x 2 s 2. So, x 1 s 1, so which is this value x 1 s 1, it has to be bigger than something; and x 2 s 2, it has to be

also bigger than something. So, basically both of the values have to be bigger than certain amount, certain quantity. So, this neighborhood so if i put x i s i is equal to gamma tau, look if I look at that line right. So, every tau is changing, the x i s i is taking a different value gamma into tau.

So, x 1 s 1 is equal to gamma tau, x 2 s 2 is equal to gamma tau, for some tau and some gamma which is fixed. So, you will get a point here, and a point here, gamma is very small, and you know tau basically, you will get a point here, because gamma is small, and whatever be the value of tau gamma is pulling the tau down. So, x 1 s 1 is we will get a point here, corresponding x 2 s 2, a similar point here. Similarly, you will create points like this, which are actually, coming down to 0. So basically, it will move, this is this is the neighborhood, so you observe this is a bigger neighborhood, because you take one point here, which is x 1 s 1 equal to gamma tau; you will take the same x 2 s 2 equal to gamma tau here. So, this is the value of so this is x 1 s 1, so when tau is bigger, this is say gamma tau. So, this will be a value of x 1 s 1. So, you will take all x 1 s 1 on the top, so, x 1 s 1 which is bigger than gamma tau, you will take this part.

Here this is your gamma tau, so, x 2 s 2 equal to gamma tau, you would take everything bigger than x 2 s 2 equal to gamma tau, so this is my neighborhood. So, this this is my N… So, you start with the neighborhood point say x 0, and we take as much long step as possible, even I am, if it is possible to come to the boundary of this thing, then take another step here, again take another step here, and come here. So, you can take quite a bit of long steps to come towards the solution.

So, what you do in this particular case, how do you take this centering parameter sigma; you basically fix to take take, you fix two values of sigma, sigma max and sigma mean; So, this sigma max and sigma mean both are in the interval 0 1. Now at every step k, but you have to remember that sigma max and sigma mean has to follow this inequality, they cannot be sigmas can be choose from 0 and 1, sigma mean and sigma max has to be has to be bigger than 0 and less than 1, strictly less than 1. Now, what I will do is that for any k that you choose for any k, so I am I am making a slightly more general now in the way I am telling.

So, what you do here is that you choose your sigma k, sigma k is now adaptive, it can change as you change the iteration; it can simply change as you change the iteration;

earlier, when we had a short step path following method, sigma k was fixed; it was 0.4 possibly, so sigma k was fixed, sigma k was sigma for everything; the Predictor-Corrector method in one step, sigma was 0, other step sigma was 1; here the Corrector step the sigma was… A predictor step the sigma was 0, corrector step, which was the sigma was 1, but here sigma k becomes adaptive, so this long step path following algorithm, takes you more and more closer to algorithms, which are more of practical use, as the one of the most important such algorithm is the Sanjay Mehrotra's Predictor-Corrector algorithm.

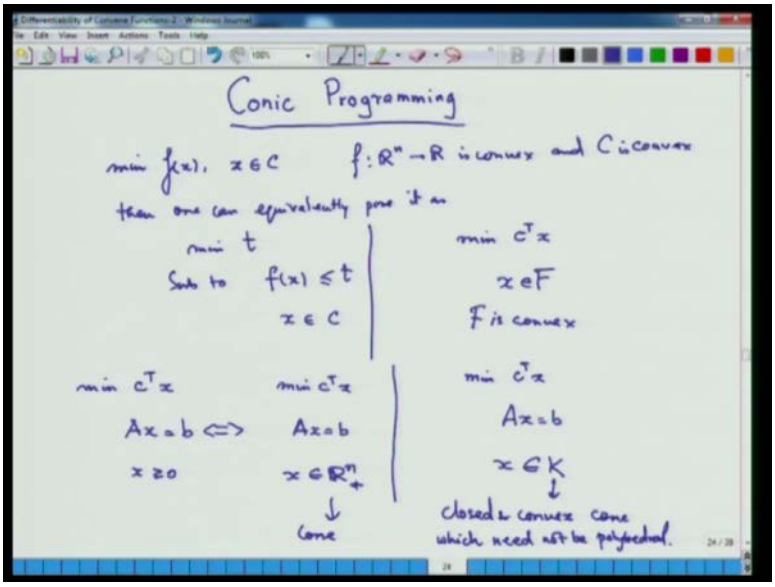
So, here sigma k is chosen from this set of values, so you have sigma max, sigma mean and sigma max, and what you do is again using the chosen sigma k, find… Now, choose the as much the large value, you can choose of alpha. So, choose as much large value of alpha in 0 1, so that (x k plus 1,y k plus 1,s k plus 1), which is given as (x k,y k,s k) plus alpha (delta x k,delta y k,delta s k), this whole thing is actually in… So, it continuous to remain in the neighborhood; so find the largest alpha for which this this value would be this, and that then this particular vector is called (x k plus 1,y k plus 1,s k plus 1).

So, choose as much large alpha you can means, if you take the 0.1, if you choose the large alpha, you can go as much quite far off from the central path actually, but the interesting part is that you are still bounded, you cannot just go towards the boundary, where one of the x i(s) or s i(s) can become 0, which is really not, we cannot do it, because x i into s i has to be equal to nu and nu plus something. So, it has to x i s i both has to be positive. So, we are stopping x i s i by this neighborhood, which is large, we are stopping x i and s i to go to the boundary, anyone of them where x 1 x x 1 s 1 here, x 2 s 2 here, to go the boundary; it cannot just drop to 0, one of them cannot drop to 0, both of them has to be positive.

But giving this large space, I am allowing a lot of movement, so if I can take lot of movement in much shorter number of steps, I can come towards the solution; and that is the whole idea behind the long step path following method; with this, we are ending the our discussion of the pleasures of linear programming, you see how effective we can do a lot of things with linear programming, and which is a very, very important part of convex programming, but this long step path following methods etcetera, also applied also applied in convex programming, where my functions are non-linear.

So, we are not getting in to those details, but now we are going to study another important class of convex programming problem, which arises out of the generalization of linear programming problems, and those are conic programming problems, and a particular part of it, which is a special part called semi definite programming problem is something, which we are going to concentrate on at this moment.

(Refer Slide Time: 19:21)



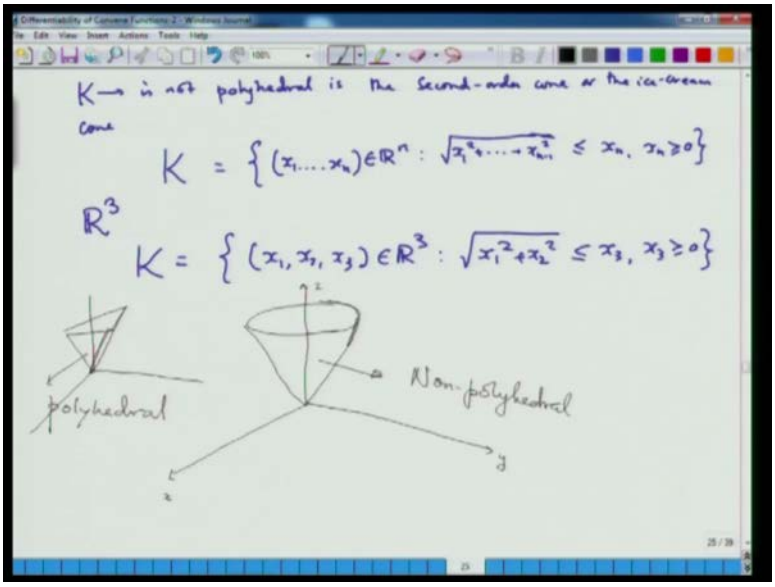
Now, we are coming to a very, very important class of problems called conic programming problems. In fact, if you give me any linear programming problem, I can always write it as a minimization of a linear function over a convex set. So, in general every convex programming problem can be posed as a minimization of a linear programming problem over a convex set. So, linear studies of linear programming problem over convex sets plays a extremely central role in convex optimization; now for example, if you take any linear problem, any convex optimization problem, minimize f (x) x element to of C, subject to this, where f is convex, and C is convex, then one can equivalently pose it as minimize t; now this problem is a problem, where I have two variables x and t, but this function, this function t, this a linear function in x and t.

So, we are basically minimizing any convex, this convex programming problem can be viewed as a minimization of a linear function over a convex set. So this, because f is convex, so this is a convex set, and C is a anyway convex set. So, in general, we want to study minimization of a linear programming problem, in where x is in R n, C is in R n

over some convex set A, because here I have just put say A is convex. So, this is the class of problems that you will interest us; one of the important problems that we have just finished studying here is the linear programming problem in the standard form, where the set C or A or in fact, I should write here, I should put F, instead of confusing things, as here I will use a matrix notation, A will come.

Now, if you take minimize this x greater than equal to 0, if I can now equivalently write this problem as subject to A x equal to b, and x element of R n plus; if you observe that this convex set is a polyhedral set, because they can be expressed as an intersection of half spaces and i proplanes, so a linear programming problem is minimization of a linear function over a polyhedral convex set; now how can I generalize this idea? One of the direct generalizations that can come is to write this problem as follows, where R n plus is a cone, and a polyhedral cone, I can take K to be any closed convex cone, which need not be polyhedral. I can take this as a thing, which need not be at all polyhedral.

(Refer Slide Time: 23:44)



One such example is where K is not polyhedral, the second order cone or the orange cone, cone or I would get in to this orange name and all or the ice cream cone. So, let me describe this cone K in R n, it would be of of this form.

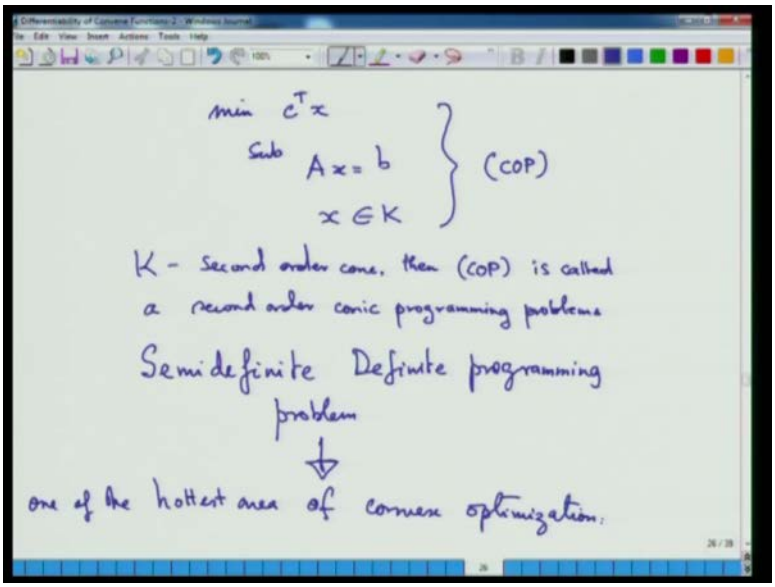
(No audio from 24:31 to 24:43)

This is less than x n, where x n is greater than equal to 0. So, how do you view it geometrically? If you view it geometrically, let us take the case of R 3; in R 3 K would look like following root over x 1 square plus x 2 square is less than equal to x 3 with x 3 bigger than 0. So, this cone can be drawn as follows; x-axis, y-axis, z-axis, this cone looks like this; not that the drawing is not so good, but not very bad either I would say.

(No audio from 25:59 to 26:12)

You look at the cone, this looks like an ice cream cone; like the one, you get in shops, but this is not polyhedral, because the the sides are rounded, a polyhedral cone cannot have rounded sides, polyhedral cones have to have flat sides, if you if you want to have a polyhedral cone, a polyhedral cone would be like this; for example, this would be the example of polyhedral cone. So, this is polyhedral cone; and this is non polyhedral cone. So, these are the two drawing.

(Refer Slide Time: 27:08)



So, when I put k is equal to this, so when I have a conic programming problem, where I put minimize C transpose x, subject to A x equal to b, and x is element of K, where K is the second order cone; then, this problem if I call it conic problem or COP, then COP is called a second order conic programming problem; a second order conic programming problem SOCP - Second Order Conic Programming Problem. So, I should conic programming problem. So, this is not a linear programming problem in general, but a convex programming problem, because my final feasible set is not polyhedral. So, if my

final feasible set is not polyhedral, I cannot call such a problem as a linear programming problem.

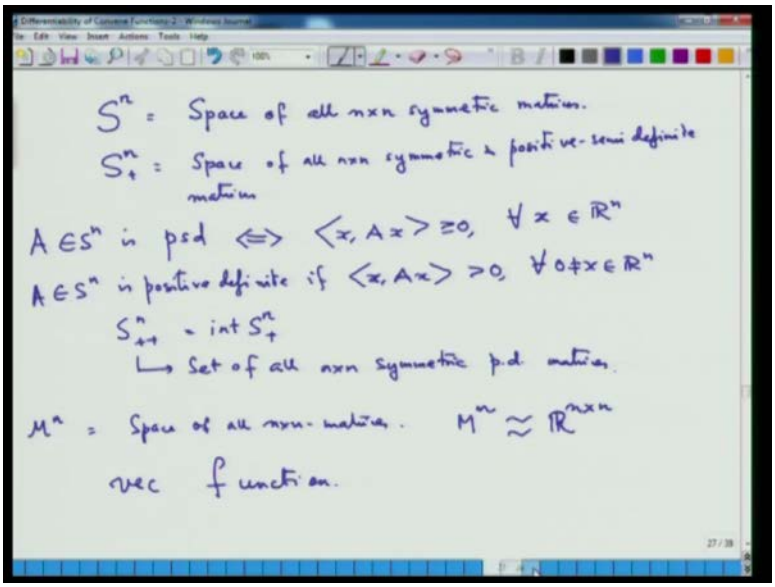
Today, I am not giving examples, but I will soon give you examples of how these classes or problems are important; a lot of important and interesting application problems can be modeled as a second order conic programming problem; now here we have been consent essentially with x is in R n and all these stuff, but R n is not the only finite dimensional space, though any finite dimensional space can be of dimension n, can be I have a have a has a bisection with R n naturally, but it does not mean that I cannot look into any other framework of finite dimension.

For example a set of matrices we have spoken about semi definite programming; now in this case basically if we look at finite dimensional spaces from a basis free point of view, then we are essentially talking about any finite dimensional space. So, we are now going to talk about semi definite programming; a semi definite programming at this moment, one of the hottest areas of convex optimization.

(No audio from 29:31 to 29:51)

One of the hottest area of … So, how do you, we have already described the beta what semi definite programming trying to find its dual; so, let me first today introduce what a semi definite programming problem is, duality etcetera will come later on.

(Refer Slide Time: 30:24)

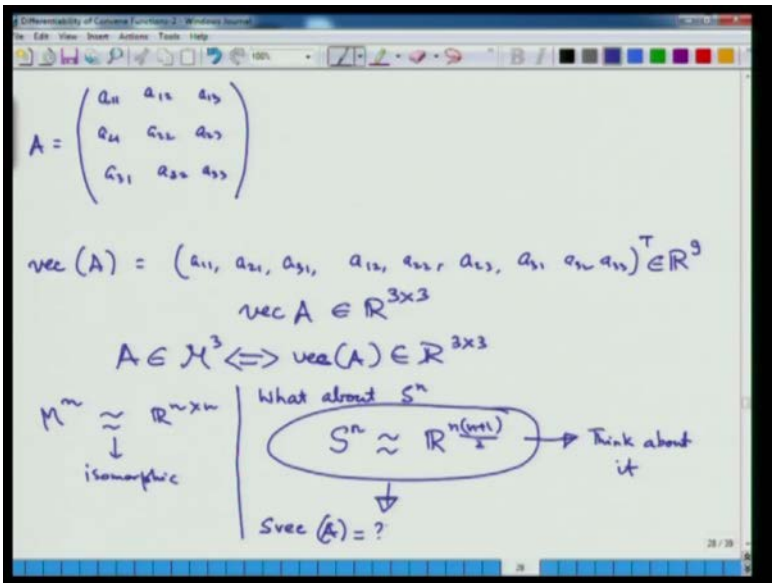


Now, in our space, we will now consider the space S n, it is a space of all n cross n symmetric matrices; now I will also consider like if this was R n, then I will consider the cone R n plus. So, what is the natural cone here? In this case, the natural cone is the space of all n cross n symmetric and positive-semi definite matrices. Now by A is positive semi definite, which will now start writing short as psd, a symmetric matrix A. So, if A is element of S n is psd, I will just write it in a more clear way; symmetric n cross n matrix is psd if and only if x, A x is greater than equal to 0 for all x in R n.

It is there is also a concept of positive definite, so A element of S n is positive definite, if x, A x is strictly bigger than 0 for all non-zero x in R n; now what I want to tell you is that if you collect all the positive definite matrices, the interior of S n plus, which we call S n plus plus. So, this is nothing the set of all pd matrices - positive definite n cross n symmetric pd matrices - positive definite matrices. Now, how do you relate a matrix say an n, suppose I have a collection of n cross n matrices, suppose M n just is a collection of space of all n cross n matrices, space of all n cross n matrices.

So, how do I say that this space of matrices has an one to one mapping with the space R n and how do I say M n is almost like R n; R n, where n n has to be decided, actually M n will will soon tell you is isomorphic to n cross n. So, any finite dimensional space is actually related to the euclidian space of this type. So, how do you do it? So, this is done by something called a vec operation. So, what is this vec operation? The vec function; so, these are some things, which you have to know about matrices in the beginning to truly appreciate the beauty of semi definite programming.

(Refer Slide Time: 34:07)

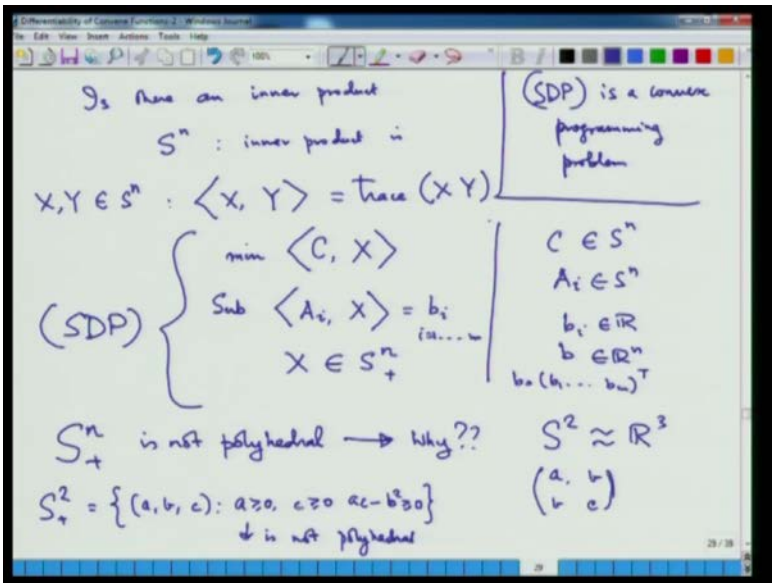


So, let us look at this one; so what is a vec operation? Let me just take the 3 cross 3, n cross n matrix need not be symmetric; a 11 a 12 a 13 a 21 a 22 a 23 a 31 a 32 a 33; now if you look at this, my vec operation if this is my A, what I do? I take this one, I stack it, take this, put it here, as in one column; take the next column, stack it, below it; third column stack it, below it. So, that will give me a vector, so my vec of A, the vector associated with the matrix A is nothing but (a 11, a 21, a 31, a 12, a 22, a 23, a 31, a 32, a 33) transpose, because we write things as column vector. So at 3 3 3 - 9; so, this is R 9, which is same as… So I can write that vec of A is element of R 9, so vec of A belongs to R 3 cross 3. So M 3, to which A belongs to, is actually M 3 is similar to if when A belongs to M 3, if and only if vec of A belongs to R 9 - R 3 cross 3. So, M of n is isomorphic, this symbolizes in mathematic is isomorphic means structurally the same that is as the bijection between them to R n cross n, where M n is a set of n cross n matrices.

Now, what about S n plus sorry what about S n? S n also a space of n cross n matrices, so, is it same as R n cross n? No, there is a little bit of clacks, there is little bit of (( )), small thing that one has to observe, because S n is a symmetric matrix, because a 12 is a 2 1, a 1 3 is equal to a 3 1, a 2 3 is equal to a 3 2. So, we have to take those things in to account, and once we take this things in to account, you will see at S n is isomorphic to the space; how does it happen? Think about it and we will discuss it tomorrow; this would associated with this is something called an S vec operation; for example, in if you

have the S vec operation, S vec of A, you do not repeat the vectors that is the whole thing, you do not repeat the vectors that is the whole thing, you do not repeat the vectors. So my so you do not the repeat the elements here, so you see a dimension is getting cut down.

(Refer Slide Time: 37:53)



So, we will discuss tomorrow, what is S vec (A), but now if I am in the space S n. So, is there an inner product in S n? In fact, if you take the inner space M n also I can define an inner product, but for S n, the inner product is given as… So, you took take 2 X and Y in S n, an inner product is given as trace of X actually, X transpose Y in general, but X transpose is same as X here, because it is symmetric. So, X is trace of X Y. So, you know that trace is nothing but the sum of the diagonal elements of a matrix.

So, now once you have this, then it allows you to frame an optimization problem; now I will frame, as if I am framing a linear programming problem in the spacer matrices, so I will take an element C from S n, and consider minimizing of this, this is nothing but trace of C x, which is this subject to… So, everything is just like R n plus this R n plus; A i here what would happen C is of course, in S n; A i is in S n, x is of an b i is, so I will take i from 1 to m. So, the vector is b i is in R, so the vector b is in R n. So, which is b is nothing but b 1, b 2, b n. Now, this is called a semi definite programming problem or more colloquially as SDP is very important to know that this problem is not a linear programming problem in the spacer matrices, you cannot do a simplex method type of

thing and solve this problem; this is in general a convex programming problem, because S n plus is not polyhedral; the question is why? I will try to explain to you in two different ways.

Today before I end this talk, I will try to just explain to you one simple fact is that if you take say n equal to 2, a two-dimensional scenario that is S 2 plus, then what would happen is that S 2 plus would consist of all positive semi definite matrices; 2 cross 2 positive semi definite matrices, but S n itself is how much, is in is isomorphic to which space, it is isomorphic to n in R, n into n plus 1 by 2, if i put n equal to 3, it will be 3 into 3 plus 1 by S n equal to 2, then it will be 2 into 2 plus 1 by 2. So, it will be R 3. So, S 2 is actually isomorphic to R 3, it is a three-dimensional space. So, S 2 plus… So, if you have a symmetric matrix, you will have a b b d b c sorry. So, this is the matrix. So, this is (a,b,c), which is giving you the matrix, so (a,b,c) is in R 3.

Now, if you look at it; so what do I mean by this matrix, when positive semi definite; it simply means that I should have a bigger than 0, all the principle minus are determinant of the principle minus the greater than equal to 0; c is greater than equal to 0 and ac minus b square is greater than equal to 0; now if I look at the whole thing in R 3, is it a polyhedral set in R 3? The answer is no, because polyhedral set is nothing but a set described through linear equations and inequalities; here, the first two equations inequalities, which are linear, but the third equation, third inequality is quadratic, it is not linear. So, this set is not polyhedral.

So, what you have here is a convex programming problem. So, we have to understand that SDP is a convex programming problem and not a linear programming problem in general. So, if you talk about duality theory and all those things, strong duality always holding for linear programming problem, such things do not happen for conic programming SOCP or SDP problem. So, in tomorrow's class, I will start discussing the S vec and give you an another very general way of telling, why this is not polyhedral, we will show that you can always map this S n plus to the second order cone, which is not polyhedral.

Secondly, tomorrow we will start giving you some interesting examples of problems, which can be cast as SDP, because SDP in convex programming problem and there are interior point methods to solve it SDP is a tractable problem, is the tractability issue

which is the most important thing in optimization. So, we show that a lot of problems come under the category SDP, and then we will discuss its optimality conditions and duality, and a brief outline of how interior point methods can be used to solve it, that would be possibly be the last but one lecture in this course, and the last lecture would possibly take in to some more general view of sub gradient optimization. So, with this I would end today's lecture, thank you, and looking forward to talk to you on SDP in the next class, because SDP is a very, very important part of current modern convex optimization, thank you very much.