Dynamic Data Assimilation

Prof. S Lakshmivarahan
School of Computer Science
Indian Institute of Technology, Madras

Lecture – 35
Covariance Square Root Filter

In the previous module 8.1 we had given a complete derivation of the Kalman filter, where the dynamics is linear and the observations are linear functions of state. This is the classical LQG problem. We gave a complete derivation of the Kalman filter equations, and we illustrated profusely with a scalar example. We talked about the evolution of model forecast without observation, then we brought an observation.

We did the Kalman filter derivation. We also analyzed the properties of the growth rate of forecast errors and examined under what condition the forecast errors will converge. And then we also talked about stability the filter in terms of what happens to the errors. In the stability of the filter you can readily see this analysis, provide a complete characterization of the behavior of Kalman filter equation; the classical Kalman filter.

So, when we say Kalman filter, we essentially mean application of the filtering equations in the case of linear dynamics and linear observations linear in this state. So, that is the classical LQG. Now in the early days when Kalman filter was implemented, especially one of the earliest application was in space travel. The onboard computers had only 8 bit. They had only 8 bit processors.

So, when you do arithmetic in 8 bit processors that could be errors, due to round off. One way to be able to improve the quality of computation is to be able to reduce the conditioning numbers for matrices that are involved. One way to reduce the conditioning condition number for matrix matrices is to consider square root versions of the matrices, instead of the matrix themselves.

So, what kind of matrices we are in, involve in. We are involved in forecast covariance analysis covariance, these matrices. If it happens to be a large condition number that could be difficulties coming from numerical instabilities, in order to be able to induce numerical stability in the early to mid 60s.

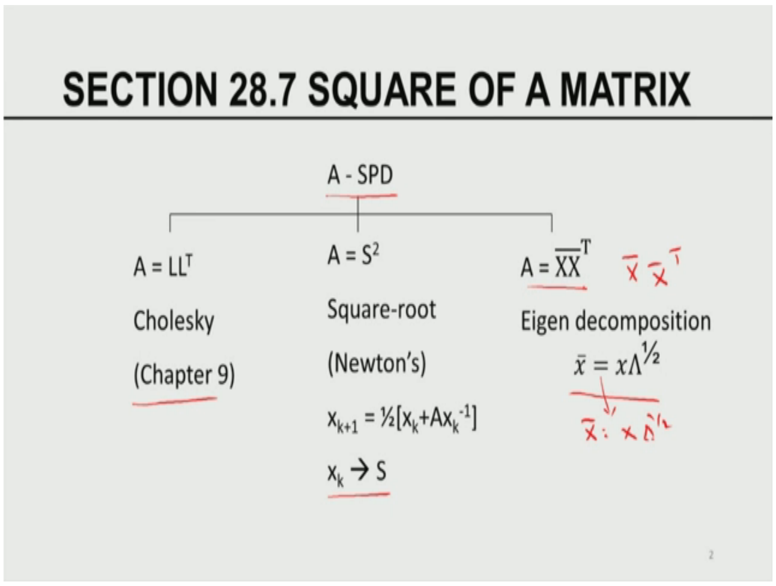
It was felt that one needs to be able to improve the conditioning of the calculations in order to improve the overall stability and accuracy of the solutions calculated, that led to what is called square root filters. Since we have derived the Kalman filter equation within the covariance setup, its called covariance square root filter. I also want to remind you the inverse of the covariance is called information. So, one could readily derive the Kalman filter equation in the information form, are then that is called information filter once i .

In other words instead of working with the covariance, we will simply work with the inverse of the covariance, while the, if the covariance is symmetric positive definite the, its inverse is also symmetric positive definite. So, both the covariance and its inverse; namely the information matrix all have positive square root. So, we have covariance square root

Information square root these are some of the different versions in which Kalman filter equations can be readily and easily written the square root versions in general have better stability properties and we are going to give an argument how to derive the Kalman filter equations within the framework of covariance square root filters. So, that is the aim of this module

There is also another motivating aim sooner we will go into what is called non-linear filters non-linear filters are very difficult problems to solve we can only approximate in the case of non-linear filters we will also introduce an approximation one form of approximation using what is called ensemble filters there is a natural connection between ensemble realizations of filters with co covariance square root filter. So, it is also for that reason we would like to educate the reader with respect to the derivation the Kalman filter equations not in the covariance form, but in the square root of the covariance form. So, that leads to covariance square root filter.

(Refer Slide Time: 05:03)



SECTION 28.7 SQUARE OF A MATRIX

A - SPD

$A = LL^T$

Cholesky

(Chapter 9)

$A = S^2$

Square-root

(Newton's)

$X_{k+1} = \frac{1}{2}[X_k + AX_k^{-1}]$

$X_k \rightarrow S$

$A = \bar{X}\bar{X}^T$ $\bar{X}\bar{X}^T$

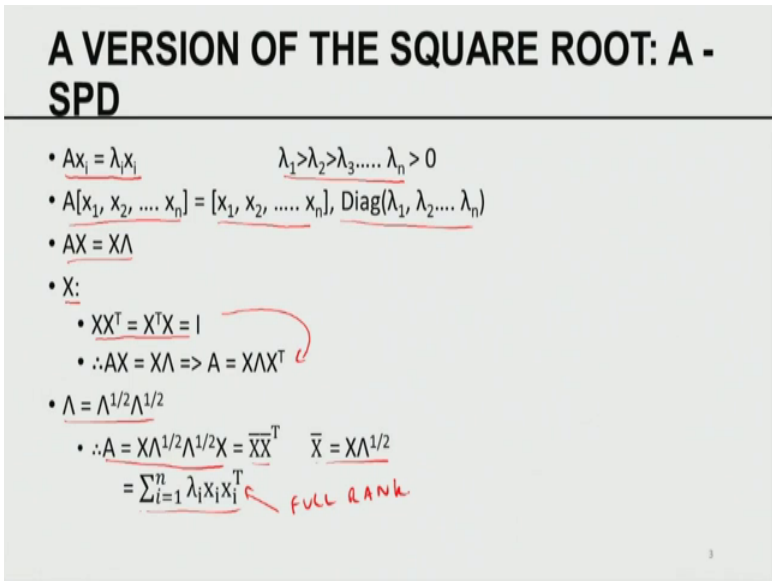
Eigen decomposition

$\bar{x} = x\Lambda^{1/2}$

$\bar{X} : X\Lambda^{1/2}$

Please recall square root of a symmetric positive definite matrix A can be given in one of three ways. One is in Cholesky, another is using yeah symmetric square root version, and solving the symmetric square root version using a Newton's method. Another one is called the eigenvalue decomposition for simplicity in calculation to illustrate basic ideas. We would either use Cholesky or Eigen decomposition, whichever is convenient for discussion.

(Refer Slide Time: 05:34)



A VERSION OF THE SQUARE ROOT: A - SPD

- $Ax_i = \lambda_i x_i$ $\lambda_1 > \lambda_2 > \lambda_3 \ldots\ldots \lambda_n > 0$
- $A[x_1, x_2, \ldots. x_n] = [x_1, x_2, \ldots.. x_n], \text{Diag}(\lambda_1, \lambda_2 \ldots. \lambda_n)$
- $AX = X\Lambda$
- X:
  - $XX^T = X^TX = I$
  - $\therefore AX = X\Lambda \Rightarrow A = X\Lambda X^T$
- $\Lambda = \Lambda^{1/2}\Lambda^{1/2}$
  - $\therefore A = X\Lambda^{1/2}\Lambda^{1/2}X = \bar{X}\bar{X}^T$ $\bar{X} = X\Lambda^{1/2}$

    $= \sum_{i=1}^{n} \lambda_i x_i x_i^T$ ← FULL RANK

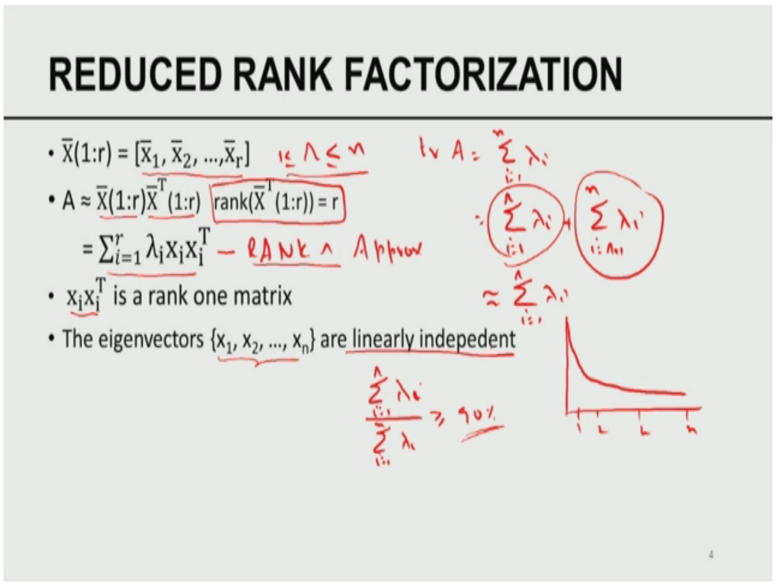
In this particular case, we are going to be dependent on the Eigen value, Eigen decomposition based square root version. Please go back to the previous case X bar is a matrix. So, this is X bar times this. So, this could be written as this. This is going to be written as X bar times X bar transpose, but X bar matrix X bar. So, this must be actually X bar is equal to X times lambda to the power half.

Lambda to the power half is the square root of the diagonal matrix, that consists of Eigen values. Since the matrix A is positive definite, all the Eigen values are positive square would exist. So, this is how the matrix square root is defined. So, we are going to use the matrix square root derived from Eigen decomposition in our discussion of the covariance square root filter derivation.

So, let me quickly talk about the covariance matrix in the properties if A is a matrix Ax i is equal to lambda i lambda A is SPD. So, all the Eigen values are positive. So, this can be written succinctly as A X 1 Xn X 1 Xn to diagonal. So, that becomes this, the matrix X is such that it is orthogonal. Therefore, by explaining this property, I can get a decom, a multiplicative decomposition lambda is lambda to the power half lambda to the power half. So, I can get a decomposition, which is X bar this is that, and that is also equal to in component form lambda i times Xi Xi transpose. So, this is the crux of the Eigen decomposition, this is the crux of the Eigen decomposition.

Now, what are we going to do? We are going to talk about two things; full rank as well as reduce rank formulation. What do you mean by full rank reduce rank formulation. If I took all the Eigen values, and Eigen vectors in the sum; that is called the full rank. So, this is the full rank factorization.

(Refer Slide Time: 08:05)



REDUCED RANK FACTORIZATION

- $\bar{X}(1{:}r) = [\bar{x}_1, \bar{x}_2, \ldots, \bar{x}_r]$ $1 \le r \le n$ $\quad tr\, A = \sum_{i=1}^{n} \lambda_i$
- $A \approx \bar{X}(1{:}r)\bar{X}^T(1{:}r)$ $\boxed{\text{rank}(\bar{X}^T(1{:}r)) = r}$ $\quad \sum_{i=1}^{r} \lambda_i + \sum_{i=r+1}^{n} \lambda_i$

  $= \sum_{i=1}^{r} \lambda_i x_i x_i^T$ – RANK r Approx $\quad \approx \sum_{i=1}^{r} \lambda_i$
- $x_i x_i^T$ is a rank one matrix
- The eigenvectors $\{x_1, x_2, \ldots, x_n\}$ are linearly indepedent

$$\frac{\sum_{i=1}^{r} \lambda_i}{\sum_{i=1}^{n} \lambda_i} \geq 90\%$$

4

Now, in here we are going to assume all the Eigen values 1 to r are large from r plus 1 to m are small. It can also be shown that the trace of the matrix A which is a covariance matrix. So, if A is the covariance matrix A is a speed trace of A, is equal to sum of all the Eigen values i is equal to 1 2 n.

So, I can divide this into two summation in there summation of two parts i 1 is equal to r lambda i plus summation lambda i is equal to i is equal to r plus 1 to n. If this is small compared to this proximate this by i is equal to 1 2 or lambda i. Therefore, I am going to take only the first r columns corresponding to the first r Eigen values. So, this is the matrix of first r eigenvalues, where r is f number which is less than or equal to 1 1 is less than or equal to r. So, r lies in this region, if r lies in this region. Now I can approximate A using the product of X bar, and x bar transpose. Please understand the rank of this matrix is r. So, this is what is called rank r approximation to a rank r approximation to A. why are we talking about rank r approximation, because it also comes from the computational demands.

If the matrix is very large, and if the eigenvalues are such that 1 2 k up to n if the eigenvalues falls very sharply like this, it can be shown that the sum of the eigenvalues, the sum of the first r eigenvalues; that means, the sum of the first r eigenvalues i is equal to r 1 to r divided by sum of i is equal to 1 to n lambda i which is the, this is the total sum versus partial sum. If this is greater than equal to 90 percent let us say; that means, 90 percent of the information are contained in the first r eigenvalues, or it could be 95 percent ,95 percent of the information are contained in the first r values.

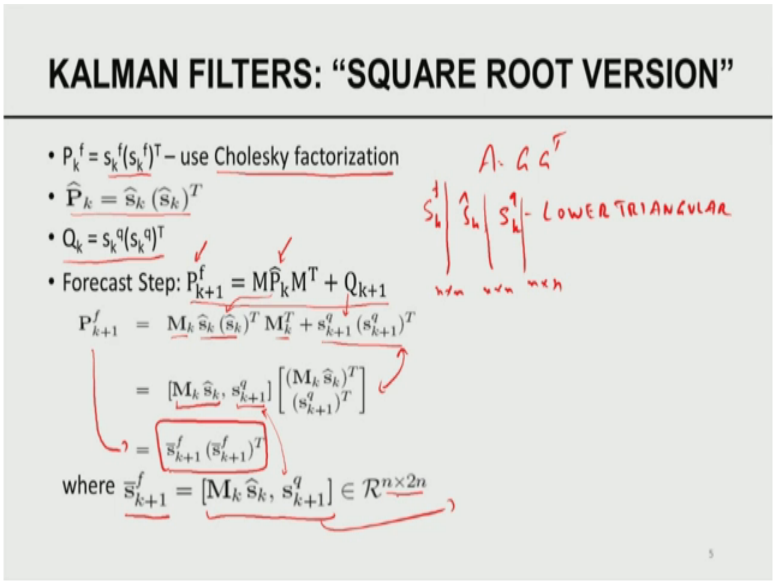
So, the value of r will depend on the kind of percentage, you are interested in the kind of approximations, you are willing to accept. So, if I consider the full rank, I will have 100 percent of the variance accounted for. If I am going to, go for a reduced rank, I am going to neglect Eigen vectors corresponding to very small values of Eigen, very small Eigen values. So, we can talk about rank or approximation. So, in this case Xi Xi transposes is a rank 1 matrix. A sum of r rank 1 matrices gives you a rank r approximation ah.

Now, please realize this Eigen value, Eigen vectors to start with their law linearly independent. So, in the first subset of r are also linearly independent. So, these are all the choices one have in terms of approximation. So, we talk about two things. Now i want to go back to, where we are we started with a positive definite matrix. We got an Eigen decomposition, we also got a square root version of A A is equal to X bar times X bar transpose; that is a square root.

Now, we are trying to interpose another idea, if in practice the eigenvalues falls off very sharply like this, the last Eigen values, the sum of these very small Eigen values do not add up much. So, we may be able to cut off at a decent value r. The r is decided by the amount of the ratio of the sum of the first r eigenvalues to the total sum of all the Eigen values. So, if i can explain 90 percent of the variance. If i can explain 95 percent of the variance, the odd will vary.

If it can 90 or 91 95 or 95 or 99 so, that is something you start with the level of approximation you are willing to accept, to start with . So, given that I can define. So, you decide the number of percentage you are comfortable with based on that I can fix the value of r, once I have the value of r. Then I can pick the first r eigenvalues, and consider an Eigen approximation to A. So, A can be approximated by a product of two rank r matrices. So, that is which is called reduced rank approximation, to the square root reduced rank approximation, obtained by appropriately selecting the columns that go into the definition of the square root matrix.

(Refer Slide Time: 13:49)



KALMAN FILTERS: "SQUARE ROOT VERSION"

- $P_k^f = s_k^f (s_k^f)^T$ – use Cholesky factorization
- $\widehat{P}_k = \widehat{s}_k (\widehat{s}_k)^T$
- $Q_k = s_k^q (s_k^q)^T$
- Forecast Step: $P_{k+1}^f = M\widehat{P}_k M^T + Q_{k+1}$

$$\begin{aligned} P_{k+1}^f &= M_k \widehat{s}_k (\widehat{s}_k)^T M_k^T + s_{k+1}^q (s_{k+1}^q)^T \\ &= [M_k \widehat{s}_k, s_{k+1}^q] \begin{bmatrix} (M_k \widehat{s}_k)^T \\ (s_{k+1}^q)^T \end{bmatrix} \\ &= \boxed{\overline{s}_{k+1}^f (\overline{s}_{k+1}^f)^T} \end{aligned}$$

where $\overline{s}_{k+1}^f = [M_k \widehat{s}_k, s_{k+1}^q] \in \mathcal{R}^{n \times 2n}$

$A = GG^T$

$s_k^f \,|\, \widehat{s}_k \,|\, s_k^q$ - LOWER TRIANGULAR

$n \times n \quad n \times n \quad n \times n$

Now, we are. So, that is one way to be able to reduce the rank. Why would you reduce the rank? If the computation of the full rank becomes too difficult, too expensive, we may settle for an approximation. The approximation could be r rank approximation. So, that is one thought. So, what is that we have done? We have simply explained one form of taking square roots, one form of taking reduce rank square roots .These are simply ideas by which I can deal with a full rank square root or reduced rank square root.

The advantage of Eigen value decomposition based square root allows for this flexibility of being able to consider the square root of full rank on. It is right that is the key message of the first four slides of this module. Now I am going to go back to using Cholesky factor as a square root. So, we are going to use Cholesky factorization. So, this means we are going to be, talking about full rank. So, let forecast covariance at time k. I am still concerned with the Kalman filter.

Please recall Kalman, we would like to be able to rewrite the Kalman filter equation in the covariance square root form. So, that is simply the product of the Cholesky factor Skf, and Skf transpose. Please understand the Cholesky factors A is equal to GG transpose. The G is lower triangular, G transpose is upper triangular. So, in this case Skf is a lower triangular square root of P kf, and its transpose is Skf transpose. So, this is simply a Cholesky factorization of the forecast covariance. Likewise I have the analysis covariance its square root.

So, we are going to rewrite the Kalman filter equation instead of Pk f using Skf. Instead of using Pk hat, I am going to rewrite and using Sk hat. I am also trying to get a square root factor for Qk which is equal to Xk Q and Sk Q transpose. So, in here Skf Sk hat, and Skq are all lower triangular, are all lower triangular matrices, and it is lower triangular matrix times its transpose. Now let us try to consider the forecast of the covariance matrix under the Kalman filter derivation that we already have done the forecast covariance at time k plus 1, depends on the analysis covariance at time k. Now replace the forecast covariance by its square root, replace the model noise covariance by its square root.

So, you can readily see Mk times the square root times Mk transpose plus the product of the two Cholesky factors, the lower triangular upper triangular part of Qk plus 1. This can be written, rewritten in the matrix form as Mk Sk hat plus Xk plus 1 Q, and the transpose of that. So, you can readily see that the product this, this equal to this product and that product is equal to that.

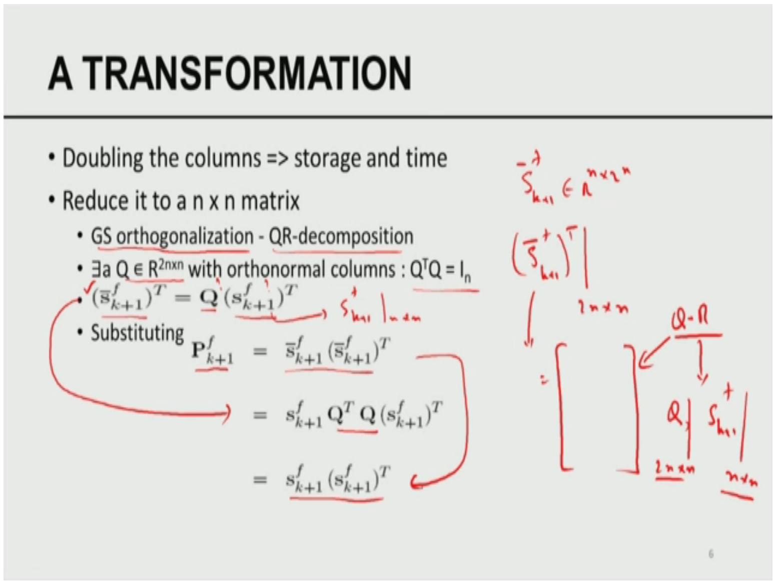
Now, I am going to concoct a new matrix, which is Xk plus 1 bar f which is given

By this matrix, this matrix account, this is Xk plus 1. Therefore, Pk plus 1 has this form of Cholesky factor. So, what does this tell you? This essentially tells you that if I have a Cholesky factor expression for Pk hat.

If I have a Cholesky X fact expression for Q k plus 1, I can get the corresponding Cholesky factor for the forecast covariance, which is given by this equation. Each of these equations, they are all n by n, they are all n by n, they are all n by n, but; however, this equation that describes the covariance of the square root of the covariance in the forecast is unfortunately is a matrix of size n by 2 n, n by 2 m is not a square matrix, n by 2 ns are tangled matrix. It has double the number of columns as the rows that is undesirable.

So, while we are interested in keeping the square root, we cannot allow the expansion for the size of the matrices here. The expansion for the size of the matrices essentially comes from the fact the forecast covariance sum of two terms; one coming from analysis covariance, another coming from the model error or model noise covariance.

(Refer Slide Time: 19:32)



A TRANSFORMATION

- Doubling the columns => storage and time
- Reduce it to a n x n matrix
    - GS orthogonalization - QR-decomposition
    - $\exists$a Q $\in$ $R^{2n\times n}$ with orthonormal columns : $Q^TQ = I_n$
    - $(\overline{\mathbf{s}}^f_{k+1})^T = \mathbf{Q}\,(\mathbf{s}^f_{k+1})^T$
    - Substituting

$$\begin{aligned}\mathbf{P}^f_{k+1} &= \overline{\mathbf{s}}^f_{k+1}\,(\overline{\mathbf{s}}^f_{k+1})^T\\ &= \mathbf{s}^f_{k+1}\,\mathbf{Q}^T\,\mathbf{Q}\,(\mathbf{s}^f_{k+1})^T\\ &= \mathbf{s}^f_{k+1}\,(\mathbf{s}^f_{k+1})^T\end{aligned}$$

$\bar{S}^f_{k+1} \in R^{n\times 2n}$

$(\bar{S}^f_{k+1})^T$ $2n\times n$

S^f_{k+1} $n\times n$

Q-R

Q $2n\times n$ S^f_{k+1} $n\times n$

6

So, this doubling of the column increases a storage as well as time. We cannot allow this to happen. So, we have to reduce it back to n by n matrix computationally. It is required to reduce n by n matrices. Therefore, then we try to reduce the nn n by 2 n matrix to n by n matrix, we have to take care of the fact that we do not lose any information. So, this must be yeah information, lossless transformation. While I am trying to compress the number of columns from 2 n to n that is the goal.

This can be done very easily by the method that we already know, that is called Gram Schmidt Orthogonalization process, which we saw in the context of QR decomposition. So, what is the basic idea, if Q in this case, because my Ex k bar please understand my ex k bar fk plus 1 my Sk bar Xk plus 1 is the matrix of size n by 2 n the corresponding q. I would like to be able to, be of the size 2 n by n with orthogonal columns, and the property Q transpose. Q is the Q transpose Q is i Q transpose Q is i; that is the condition we need, that means.

Let us look at this now Sk plus 1 f bar belongs to r n by 2 n.

So, therefore, S bar k plus 1 f transpose is a 2 n by n matrix. This matrix has more columns than rows. So, this matrix is, it looks like. So, this essentially corresponds to the over determined case of h, that we have used the cut in the context of static inverse problems. So, this is, this matrix is a rectangular matrix with double the number of rows, than the number of columns q is the matrix. Again 2 n by m I would like to be able to now get a matrix Xk plus 1 f.

So, here what is the condition, I would like to get a matrix Xk plus 1 f; that is also n n by n. So, the left hand side is given. I have to find A Q and the Sk plus 1; such that this equation can be satisfied; that is a classical QR decomposition method, that is a classical QR decomposition method, QR decomposition method by in applying to this matrix X k plus 1 bar f transpose. So,I am going to apply the QR decomposition to this. The qr decomposition will then deliver A Q and an Xk plus 1. I am sorry there is no bar. Here the Sk plus 1, it will deliver an Xk plus 1 f, this will be nn by n matrix. This will be a 2 n by n matrix.

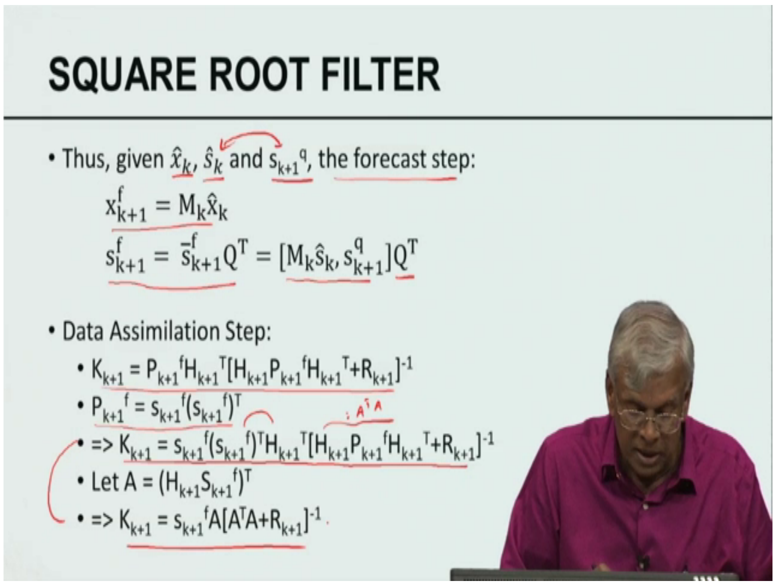
So, with these two. Now you can readily see I am trying to move towards reducing here n by 2 n matrix 2 n by n matrix, to see that what is that, we would like to do. We would like to be able to consider the old expression for k plus Pk plus 1 f, which is equal to Xk plus 1 bar f times its transpose. This comes from the previous slide. Now I have expressed the Xk plus 1 f using the QR decomposition Q transpose, Q is i therefore, its reduces to this. So, you can now readily see by using the trick of QR decomposition on the expanded matrices. I can now compress the matrix back to n by n.

So, this is an additional step I have to do, if I want to keep track of the full rank. So, here it comes some of the challenges in Kalman filter equation, when you want to rewrite from the covariance form to the covariance square root form.

Again I would like to remind you why do we go from covariance to covariant square root form, simply, because I would like to be able to increase the conditioning, and what is the conditioning. The conditioning for a square root is much better than the condition for the original matrix; that is a very well known result, that is a very well known result, why is that.

So, we have expressed Pk plus 1 in terms of our square root of 2 n by n 2 n by n matrices.

(Refer Slide Time: 24:37)



SQUARE ROOT FILTER

- Thus, given $\hat{x}_k$, $\hat{s}_k$ and $s_{k+1}{}^q$, the forecast step:

$$x^f_{k+1} = M_k\hat{x}_k$$

$$s^f_{k+1} = \bar{s}^f_{k+1}Q^T = [M_k\hat{s}_k, s^q_{k+1}]Q^T$$

- Data Assimilation Step:
  - $K_{k+1} = P_{k+1}{}^f H_{k+1}{}^T[H_{k+1}P_{k+1}{}^f H_{k+1}{}^T + R_{k+1}]^{-1}$
  - $P_{k+1}{}^f = s_{k+1}{}^f(s_{k+1}{}^f)^T$
  - $\Rightarrow K_{k+1} = s_{k+1}{}^f(s_{k+1}{}^f)^T H_{k+1}{}^T[\underbrace{H_{k+1}P_{k+1}{}^f H_{k+1}{}^T}_{A^TA} + R_{k+1}]^{-1}$
  - Let $A = (H_{k+1}S_{k+1}{}^f)^T$
  - $\Rightarrow K_{k+1} = s_{k+1}{}^f A[A^TA + R_{k+1}]^{-1}$

So, with this we come back to rewriting the filter in terms of square root matrices given. Now Xk bar Xk X Xk hat Sk hat Sk plus 1 Q. These are the two square roots of the analysis, and the analysis and the model noise, I am going to now consider the forecaster. This is the forecast.

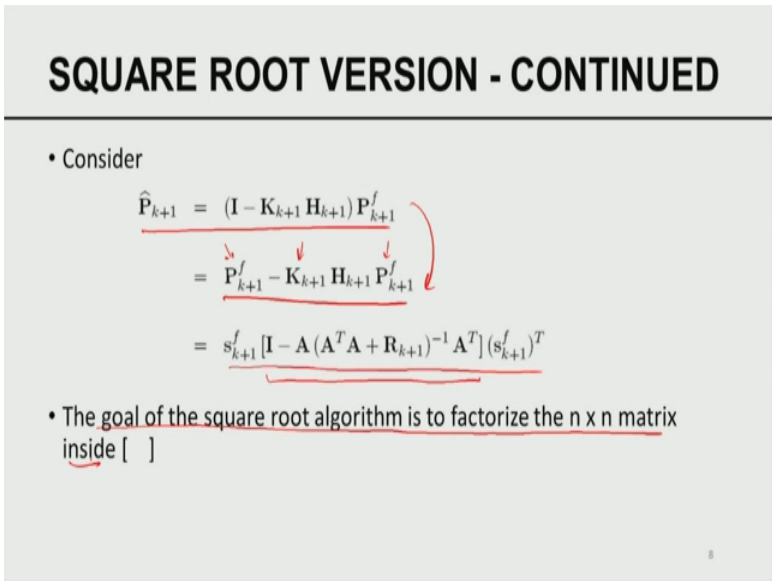
So, instead of updating the forecast covariance I am going to update the square root of the forecast covariance. The square root of the forecast covariance is related to analysis, and we have already broken them down. Therefore, Xk plus 1 f is equal to Xk plus 1 bar Q transpose; that is given by this times that, and this product is going to give us the required Cholesky decomposition. As we have seen in the previous slide, the data assimilation step. Now consists of the Kalman gain, the data assimilation step consists of the Kalman gain, it is the forecast covariance. The Kalman gain can be written in the square root, in the using the square root forms.

Now, I am going to change the matrices H k Xk, the product that comes in here I am going to call it A. So, this I can rewrite also as 8 transpose times A, using the same notation, and using the same factorization for Pk plus 1. Therefore, kk now gets this form and this form is this.

Square root version of the Kalman gain, we have come to a square root version of the

Kalman gain.

(Refer Slide Time: 26:49)



SQUARE ROOT VERSION - CONTINUED

- Consider

$$\hat{P}_{k+1} = (I - K_{k+1} H_{k+1}) P^f_{k+1}$$

$$= P^f_{k+1} - K_{k+1} H_{k+1} P^f_{k+1}$$

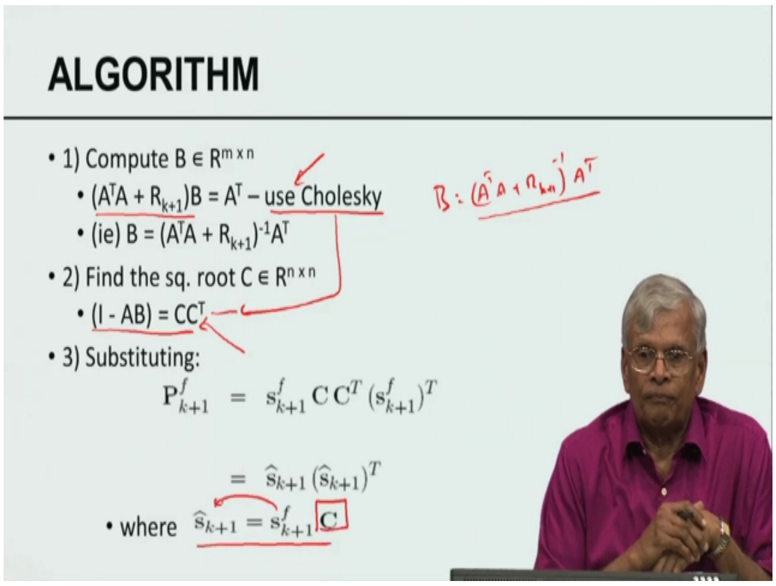
$$= s^f_{k+1} [I - A (A^T A + R_{k+1})^{-1} A^T] (s^f_{k+1})^T$$

- The goal of the square root algorithm is to factorize the n x n matrix inside [ ]

And I can now substitute the analysis covariance. This is analysis covariance form. Again I can substitute simplify this like this.

Now, I can express this in the square root form, I can express this in the square root form. I already have a square root form expression for K k plus 1 in the previous slide, by combining them all, I get the square root form for the analysis covariance. So, to complete the square root version I simply need to be able to compute the square root of the term, which is in the bracket. So, I would like to be able to get the square root algorithm to factorize the n by n matrix, inside the square bracket, in order to be able to do that.

(Refer Slide Time: 27:39)



ALGORITHM

- 1) Compute $B \in R^{m \times n}$
  - $(A^TA + R_{k+1})B = A^T$ – use Cholesky
  - (ie) $B = (A^TA + R_{k+1})^{-1}A^T$
- 2) Find the sq. root $C \in R^{n \times n}$
  - $(I - AB) = CC^T$
- 3) Substituting:

$$P^f_{k+1} = s^f_{k+1}\, C\, C^T\, (s^f_{k+1})^T$$

$$= \hat{s}_{k+1}\, (\hat{s}_{k+1})^T$$

- where $\hat{s}_{k+1} = s^f_{k+1} \boxed{C}$

Look at this now i minus A times A transpose A plus Rk inverse A transpose that is the matrix, whose square root, I am interested in, this has to be done in a couple of stages. First compute B as a solution of first compute B as a solution of a transpose A R plus R k plus 1. Please understand the B will then give us B will then give us a transpose A plus R k plus 1 inverse A transpose.

If you go back to the previous step, that is the expression which is one of the parts of the, which is one of the part of the equation, that I already have. So, this is the part that we are now concerned with, and that is computed by solving a linear system. This linear system is solved again by use of Cholesky, again by using Cholesky.

Please understand even though I need the inverse I, I even though the expression uses the inverse, I do not want the inverse I want the product, the product is simply the solution of a transpose A plus Rk plus 1 times B is equal to a transpose. You simply solving the system of linear equations, you get this.

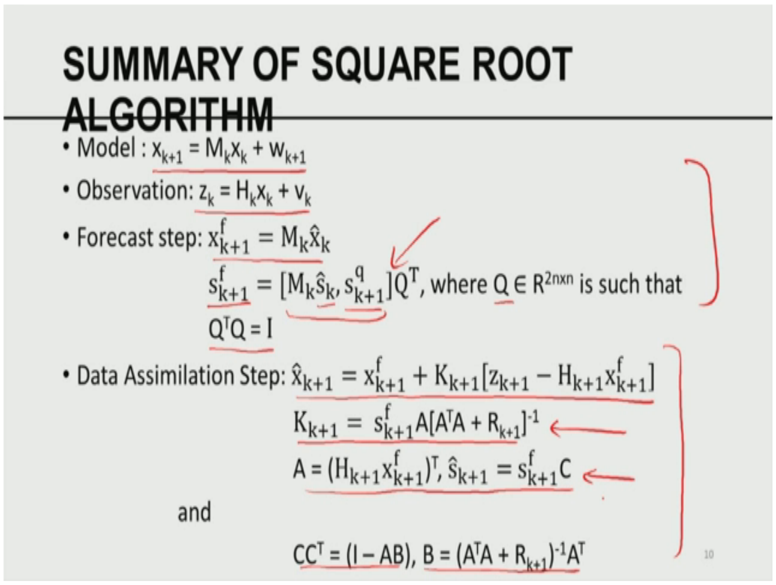
So, once you get B the term going back to the previous line Pk plus 1 is equal to Xk plus 1. The term within parentheses Sk plus 1 f transpose. So, I need to be able to find the square root of the term within the parentheses, and in order to find the square root of the term within the parentheses. Once I find B I need to be able to find the square root of I minus AB I minus AB.

So, I know what A is I. Now know what B is multiplied A B subtract from I and then find the square root, this square root also can be applied by using Cholesky factorization, substituting all this back I have Xk plus 1 C C transpose. This I am now going to define Xk plus 1 hat times Xk plus 1 hat transpose, where S k plus 1 hat is equal to Sk plus 1 f times C.

So, this is one of the fundamental equations in the square root operations to find C. So, the key is to find C. To find C I have to do a factorization here, and to find B I have to solve a linear system. So, by doing steps 1 2 and 3 I can find C. So, C what does it do. It essentially transforms the forecast square root covariance matrix to the analysis square root covariance.

So, from forecast to analysis, from forecast analysis that transformation is obtained by multiplying the forecast square root by C on the right, where C is obtained by solution of 1 and 2.

(Refer Slide Time: 31:01)



SUMMARY OF SQUARE ROOT ALGORITHM

- Model : $x_{k+1} = M_k x_k + w_{k+1}$
- Observation: $z_k = H_k x_k + v_k$
- Forecast step: $x^f_{k+1} = M_k \hat{x}_k$

  $s^f_{k+1} = [M_k \hat{s}_k, s^q_{k+1}]Q^T$, where $Q \in R^{2n\times n}$ is such that $Q^TQ = I$
- Data Assimilation Step: $\hat{x}_{k+1} = x^f_{k+1} + K_{k+1}[z_{k+1} - H_{k+1}x^f_{k+1}]$

  $K_{k+1} = s^f_{k+1}A[A^TA + R_{k+1}]^{-1}$

  $A = (H_{k+1}x^f_{k+1})^T$, $\hat{s}_{k+1} = s^f_{k+1}C$

  and

  $CC^T = (I - AB)$, $B = (A^TA + R_{k+1})^{-1}A^T$

10

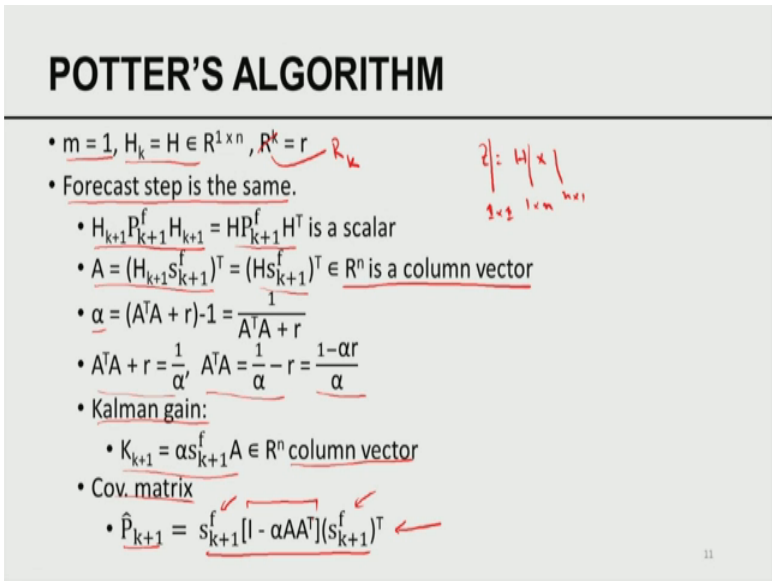
So, with this I have now come to the summary of the square root algorithm, square root version of the classical Kalman filter equation. The model equation is given by, this observation is given by that, the forecast is given by classical Kalman filter the. I am assuming, I am available, I have availability square root of the analysis. I am also assuming the availability of square root of the model noise Q is available from the QR decomposition, which is the Gram Schmidt organization. So, by doing a Gram Schmidt orthogonalization on this matrix I get S k plus 1 f. Please remember Q transpose Q is I.

Now, I am going to, go to the data assimilation step. The data assimilation step essentially the same as the previous one. So, except that, I am going to X plus kk in the square root version, where please remember Ak is given by this, and Xk plus 1 hat is equal to Xk plus 1 f times C, where C is equal to I minus A B and B is equal to a transpose A plus R k plus 1, whole inverse times a transpose. So, this essentially gives you the forecast part. This essentially gives the analysis part. This step takes care of the 4 square root. This type of the steps takes care of the farm square root. This step also takes care of the square root.

So, this is the summary of the square root version of the Kalman filter equation. You can see there are lots of mathematical ideas, but everything rush on the ability, to be able to compute the square root of a covariance matrix, and transformation that are needed to compress the expansion.

So, by combining the square root and the ability to A QR decomposition I can. In fact, derive the square root version of the Kalman filter. It was this kind of a square root version of the Kalman filter was there, was implemented in the Apollo mission that led to a very successful ability to track, and that is one of the major applications of the Kalman filter equations in space exploration.

(Refer Slide Time: 33:40)



POTTER'S ALGORITHM

- $m = 1$, $H_k = H \in R^{1\times n}$, $R^k = r$ R_k
- Forecast step is the same.
  - $H_{k+1}P^f_{k+1}H_{k+1} = HP^f_{k+1}H^T$ is a scalar
  - $A = (H_{k+1}s^f_{k+1})^T = (Hs^f_{k+1})^T \in R^n$ is a column vector
  - $\alpha = (A^TA + r)-1 = \frac{1}{A^TA + r}$
  - $A^TA + r = \frac{1}{\alpha}$, $A^TA = \frac{1}{\alpha} - r = \frac{1-\alpha r}{\alpha}$
  - Kalman gain:
    - $K_{k+1} = \alpha s^f_{k+1}A \in R^n$ column vector
  - Cov. matrix
    - $\hat{P}_{k+1} = s^f_{k+1}[I - \alpha AA^T](s^f_{k+1})^T$

$\underset{1\times 1}{z} = \underset{1\times n}{H}\,\underset{n\times 1}{x}$

11

So, given that as a background now I am going to talk about what is called potters algorithm. Potters algorithm is further simplification of the application of the square root version of the square root version of the Kalman filter equation, to be able to illustrate the potters algorithm.

I am going to assume m is 1; that means, observation I have only one single scalar observation. So, Hk is the state, is still m by m n by n n by n vector. So, Z is equal to H times X. This is a scalar, this is 1 by 1. Therefore, this is 1 by n times n by 1 Rk. This is R sub k R sub k, which is the variance of the scalar observation is R. So, the forecast step is the same, the forecast step is the same as the general equation that we have given in the previous slide.

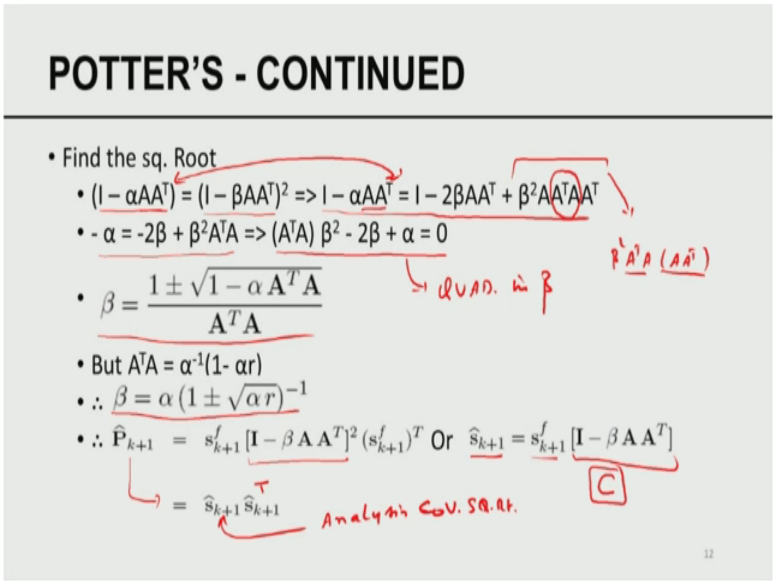
Now, I would like to be able to look at the simplification that results, when m is 1 in the analysis step. So, Hk plus 1 Pk plus 1 Hk plus 1 is equal to this; that is a scalar. So, A which is given by Hk plus 1 S k plus 1 f transpose. This is equal to H times, that is simply a column vector. So, yeah is a column vector. So, A transpose A is a scalar or is a scalar. So, alpha is alpha is given by this.

So, A transpose A plus R is 1 over alpha that for A transpose is cologne, or alpha minus R, which is equal to given by this. Therefore, I can write the Kalman gain in this particular case as k k plus 1 is equal to alpha times Sk plus 1 A, which is again a column vector. So, the covariance matrix analysis. covariance matrix can be written like this.

Now, please realize Sk plus 1 f times 1 minus alpha times A A transpose. I am sorry, this is not one, is a identity times A transpose. So, in this particular case, in this particular case, the A is a column vector A transpose is a row vector. So, this is an outer product. Matrix alpha is a constant. So, I minus constant times A matrix times this. So, this is this, this is the expression for the forecast covariance. I am sorry analysis covariance at time k plus 1. Please remember I am talking about the, I am talking about the data assimilation step, compute the Kalman gain, and the analysis covariance and here.

Now, if I want to be able to. So, this is, this factor in this case, is already square root form, this factor is already square root form. So, to get a square root I need to consider square root of only the term within the parentheses. The analysis is very similar to the general case we talked about.

(Refer Slide Time: 37:16)



POTTER'S - CONTINUED

- Find the sq. Root
  - $(I - \alpha AA^T) = (I - \beta AA^T)^2 \Rightarrow I - \alpha AA^T = I - 2\beta AA^T + \beta^2 AA^TAA^T$
  - $-\alpha = -2\beta + \beta^2 A^TA \Rightarrow (A^TA)\beta^2 - 2\beta + \alpha = 0$
  - $\beta = \frac{1 \pm \sqrt{1 - \alpha \mathbf{A}^T\mathbf{A}}}{\mathbf{A}^T\mathbf{A}}$
  - But $A^TA = \alpha^{-1}(1 - \alpha r)$
  - $\therefore \beta = \alpha(1 \pm \sqrt{\alpha r})^{-1}$
  - $\therefore \hat{\mathbf{P}}_{k+1} = \mathbf{s}^f_{k+1}[\mathbf{I} - \beta \mathbf{A}\mathbf{A}^T]^2(\mathbf{s}^f_{k+1})^T$ Or $\hat{\mathbf{s}}_{k+1} = \mathbf{s}^f_{k+1}\underbrace{[\mathbf{I} - \beta \mathbf{A}\mathbf{A}^T]}_{\boxed{C}}$

$= \hat{\mathbf{s}}_{k+1}\hat{\mathbf{s}}_{k+1}^T$

So, the square root of this matrix I minus alpha times A transpose is equal to. I would like to be able to express it as a square of the matrix with the change in constant beta. If I would like to be able to make this equal to the square of this matrix, I have the relation, this relation that essentially helps us to be able to find beta.

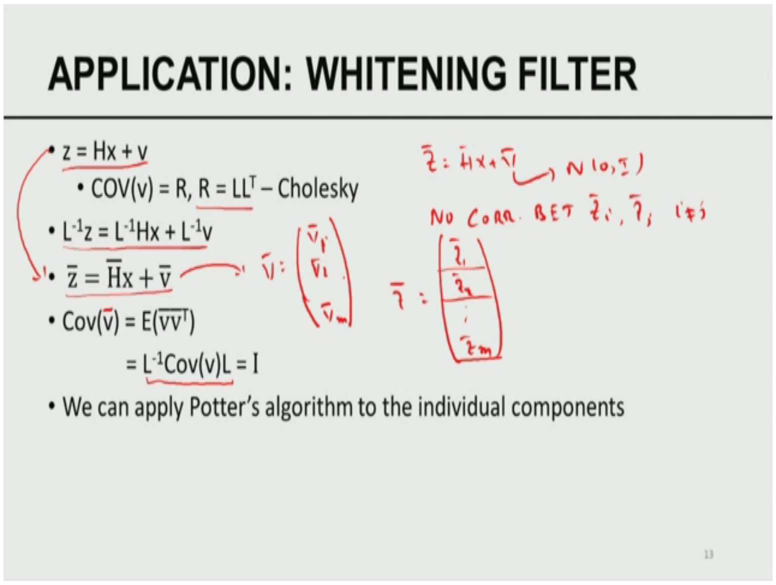
Let us go back and remind ourselves all the A is a column vector. So, A A transpose is a rank 1 matrix. So, this, that is what we are now trying to talk about . So, this relation essentially implies I minus alpha times A transpose is equal to 1 minus 2 beta A transpose, with beta square A A transpose A A transpose, because of the square involved, because of the square involved. So, I would like to be able to, I would like to be able to find an relation between alpha and beta, and that essentially tells us. So, by equating, by equating these two sorry by equating these two, I would like to be able to express beta in terms of alpha, and that gives rise to the relation that alpha and beta must be related to this alpha, and beta must be related to that .

Look at alpha and beta relate that in other words I am going to equate both sides, the corresponding terms. So, minus alpha must be equal to minus 2 beta times beta square times A transpose A A transpose A is a scalar. So, this term can be written succinctly as beta square A transpose A times A A transpose. So, A transpose is the term, that I am equating there is a A A transpose here, there is A A transpose here. So, if I consider the coefficients and equate the coefficients of A A transpose terms on both sides, I get this equality, that equality implies that beta and alpha must hand in relation, which is the equation, which is written by the equation.

Now, please understand A transpose A is a scalar A transpose A is a scalar, therefore, this becomes a quadratic equation in beta. This is a quadratic equation in beta. The solution for the quadratic equation is given by this. So, if I substitute back, and unwind all the expressions, I get beta is given by this expression. I can now substitute the matrix within this as 1. My I I minus beta times A A transpose square, because I have found the square root of that. I have found the square root of that, that essentially gives you the analysis square root is a forecast square root multiplied by this, and this plays the role of the C matrix that we have talked about in the previous slides, that is a transformation, that is the transformation.

Therefore Sk plus 1 is equal to Pk plus 1 hat is equal to Sk plus 1 hat times Sk plus 1 hat Sk plus 1 hat transpose. Therefore, this is the forecast, as this is the analysis, analysis covariance, analysis covariance square root analysis covariance square root. So, we have now a very simple factorization scheme, when there are only a one scalar observation. I do not have to take a matrix square root. Matrix square root is more expensive than scalar valued square root operations. So, that is the advantage of considering, when m is equal to 1.

(Refer Slide Time: 41:50)



APPLICATION: WHITENING FILTER

- $z = Hx + v$
  - $COV(v) = R$, $R = LL^T$ – Cholesky
- $L^{-1}z = L^{-1}Hx + L^{-1}v$
- $\bar{z} = \bar{H}x + \bar{v}$
- $Cov(\bar{v}) = E(\overline{vv^T})$

  $= L^{-1}Cov(v)L = I$
- We can apply Potter's algorithm to the individual components

$\bar{z} = \bar{H}x + \bar{v} \rightarrow N(0, I)$

NO CORR. BET $\bar{z}_i, \bar{z}_j$ $i \neq j$

$\bar{v} = \begin{pmatrix} \bar{v}_1 \\ \bar{v}_2 \\ \vdots \\ \bar{v}_m \end{pmatrix}$

$\bar{z} = \begin{pmatrix} \bar{z}_1 \\ \bar{z}_2 \\ \vdots \\ \bar{z}_m \end{pmatrix}$

13

How do we apply this m is equal to 1 to a general case. In general case, yeah my observation is a m vector is given by this.

Now, I am going to consider my V the covariance of V is R R can be expressed as a product of L, and in transpose that is a Cholesky decomposition. If you multiply both sides by L inverse I get a new equation for the observation. So, this becomes the new equation for the observation, and what is the point of this observation, what the point of this observation.

The covariance of V bar this must be V bar the covariance of V bar is equal to expected value of V bar, V bar transpose; that is equal to L inverse covariance of V times L that is equal to I. So, what does this mean this be essentially means the nice covariance is identity, if nice covariance is identity. So, what is that, we have Z is equal to I am sorry Z bar is equal to H bar X plus V bar V bar is a normal norm noise vector with zero mean, and I as I the identity matrix as the covariance.

Therefore there is no correlation between Zi and Z, the no correlation between Z bar i and z bar j, and i not equal to j ,when my i naught equal to j. If because these things are not correlated. Now i could consider Z bar 1 Z bar 2, and z bar m; that is Z bar is equal to that, and I could consider.

Now each of these observations as a single observation, there are m single observations I can try to, I can try to successively assimilate, each of these observations by the potters algorithm, that we have given in the previous case. Therefore, instead of assimilating all of them together, that involves matrix operations, I can by doing this transformation, I try to convert the problem into one of assimilating m sorry, one of assimilating ms scalar observations.

So, it is this ability to convert using the transformation, which is the L inverse is called whitening transformation. What is whitening before the transformation V had covariances; that means, Vi and Vj may be correlated, but after the transformation V bar i and D bar j are not correlated. So, in here please realize in here. In here the V bar V bar is equal to V bar i I am sorry V bar 1 V bar 2.

All the way up to V bar m, and any two. If you take any two of them, they are not correlated. Therefore, I can consider the vector to be made up of m single uncorrelated observation. If there are m single uncorrelated observation I can apply the potters formula one at A time. So, I start with the forecast. You assimilate the first component, you get an analysis, then start with that analysis, assimilate the second component. Then you get another analysis, you start with that second analysis assimilate, the third observation you get the third analysis.

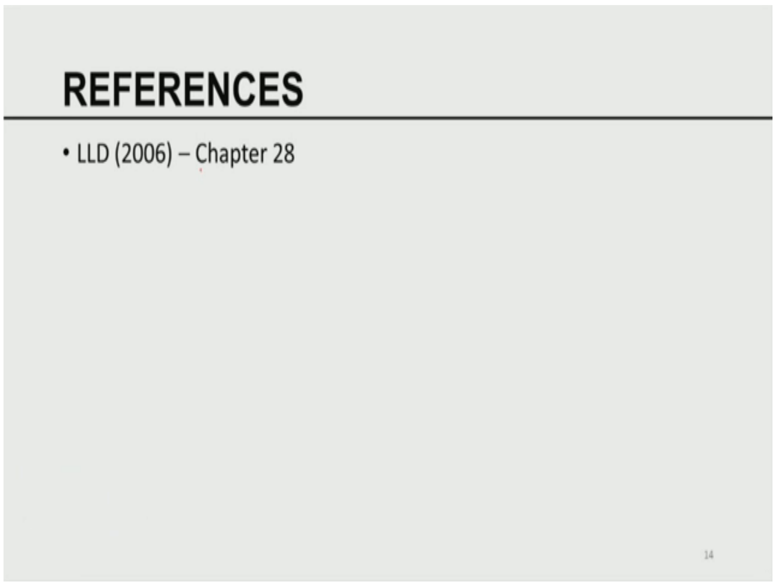
Likewise, if you come, if you assimilate all the m components 1 at a time, starting from a forecast in m steps, you will get the ultimate analysis, that takes advantage of that takes advantage of all the m individual observations, that arising out of whitening transformation. So, this way one can bring in simplicity into the computation by using suitable transformations.

Now we have talked about several different types of transformation in this lecture. In this lecture, we have talked about square root transformation of matrices QR decomposition to be able to reduce the reducer compress in a lossless way, the resulting square roots, and here we are also talking about the notion of potters algorithm, and how to apply potters algorithm component wise, by doing what is called a whitening transformation or a whitening filter.

What is the whitening filter, if I hash correlated noise through a whitening filter. The output becomes uncorrelated in the sense that in the sense that V was correlated V bar. The components of V bar are uncorrelated, this ability to transform a noise vector; that is a correlated covariance matrix to an uncorrelated covariance matrix, and the output that is called the whitening filter. By applying the whitening filter, we are able to reduce the problem of assimilating a vector of m observation to assimilating a sequence of scalar updates of m single observation. So, the assimilation step is broken down into m simple steps, where you simply take the square roots of scalars as supposed to taking square roots of matrices.

The total time required to compute the square roots of matrices can be much larger. Therefore, if you could apply potters algorithm, it could greatly simplify the analysis not only that it also improves the conditioning of the calculation, conditioning of the matrices, which in turn tries to induce the stability of the computations.with that, we come to the end of the discussion of this covariance square root filter.

(Refer Slide Time: 48:35)



REFERENCES

- LLD (2006) – Chapter 28

14

This is covered in chapter 28 of our book, and there are enough number of exercises, one has to perform in trying to verify all these things; that is why I have not given you any additional exercises, trying to verify all the calculations in here are themselves an exercise. I hope you have come to appreciate the notion of what type of square root you use. Do I use full rank approximation or partial to rank approximation? If I were to do a square root version of the covariance how, but how these transformations are aligned. Then if I want to be able to incorporate one observation at time, how potters algorithm come to the rescue.

So, by use the whitening transformation and potters algorithm, we can further simplify the simulation of vector observations. So, that is the overall summary of this module.

Thank you very much.