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

Lecture - 21
Inverse problems in deterministic continued

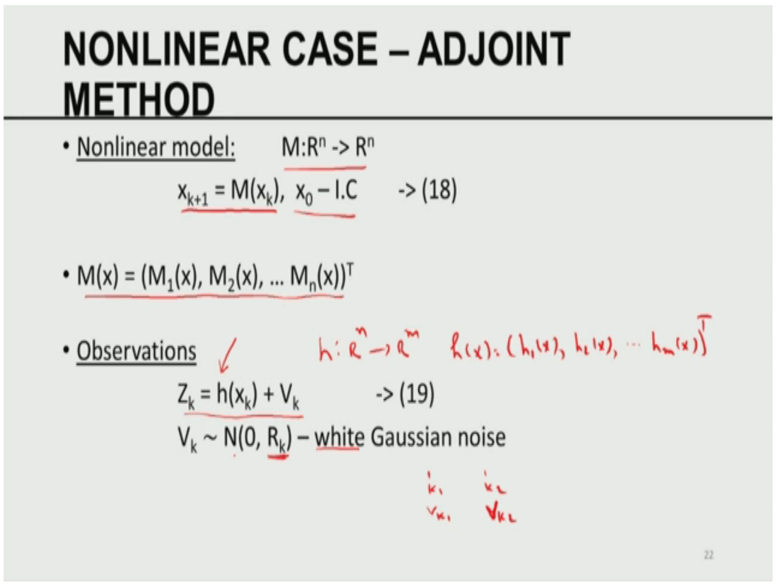
In the previous lecture we introduced the fundamental principles of 4-D war algorithm which is also called the first order join method. Using a linear deterministic dynamical system as a strong constraint we solved the constrained minimization problem using the Lagrangian multiplier technique. The whole aim of the exercise is to be able to compute the gradient of the cost functional in the context of the linear equality constraint. And once we computed the gradient numerically we then have to use the gradient in an optimization minimization algorithm such as gradient method or a conjugate gradient method.

So, since we have seen conjugate gradient methods as well as gradient method how they work, we did not elaborate on that part of the problem that is the bottom half of the problem the top half of the problem relates to computing the gradient itself. So, 4-D war adjoin method are essentially methods for computing the gradient of the cost functional with model as a strong constraint. That is the day.

Now I am going to provide an extension, in the previous case we considered linear perfect model deterministic, we considered a noisy observation, observations where linear functions of the state. We remember we considered four cases: model can be linear nonlinear observation can be linear, nonlinear; both being linear is the easy case, both being linear is the more complex case. Now I am going to go over to the nonlinear case when I say nonlinear everything is nonlinear; model is nonlinear, observations are none linear in the functions of time, observations are noisy observation.

So, using nonlinear noisy observations I have to determine the initial condition based on a finite number of indirect information about the states.

(Refer Slide Time: 02:42)



NONLINEAR CASE – ADJOINT METHOD

- Nonlinear model: $M: R^n \to R^n$

  $x_{k+1} = M(x_k)$, x_0 – I.C -> (18)

- $M(x) = (M_1(x), M_2(x), \ldots M_n(x))^T$

- Observations $h: R^n \to R^m$ $h(x) = (h_1(x), h_2(x), \cdots h_m(x))^T$

  $Z_k = h(x_k) + V_k$ -> (19)

  $V_k \sim N(0, R_k)$ – white Gaussian noise

  k_1 k_2

  V_{k_1} V_{k_2}

22

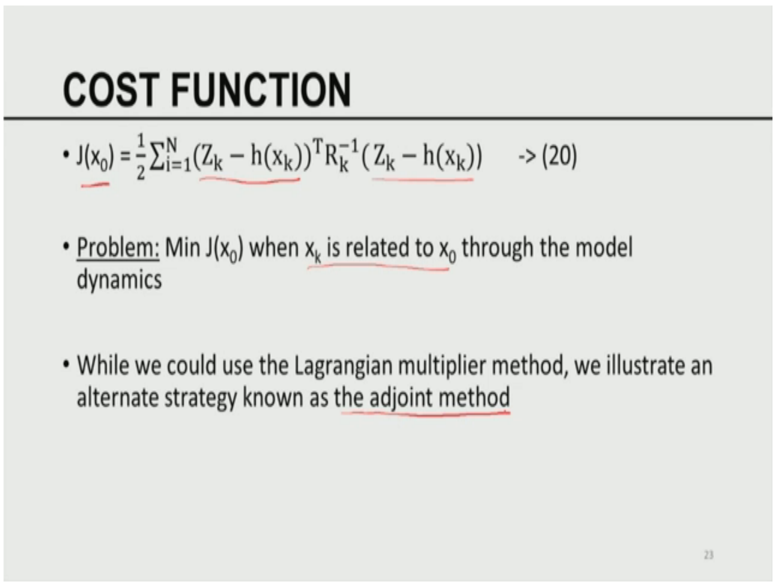
So given that, I have the nonlinear model deterministic initial condition it is x naught, M is a model map, M of X is the M 1 of X M 2 of x M n of x. So, M is a function from R n to R M is the vector valued function of a vector. Observations or nonlinear functions h is the forward operator, again h is a map from R n to R M. So, h of x is also is equal to h 1 of x h 2 of x and h m of x transpose or k is the observation noise; we are assuming white. What do you mean by white V k? So, if I consider two instances in time k 1 and k 2 I have V k 1 I have V k 2 I have V k 2; V k 1 and V k 2 are uncorrelated that is what white means. White means the temporally uncorrelated, but at a given instant the covariance matrix for the noise is given by R k. So, V k is normally distributed with mean 0 and R k as the covariance.

What do you mean by mean 0? Mean 0 means the instruments that measure the states do not have any bias. That means, if there is a bias in the instrument, I have already calibrated the instrument against known standards, I have subtracted the bias, therefore observations are bias free observation noise are mean 0 the observational covariance R k; R k is at a given time and observations at different times are temporally uncorrelated.

These are the standard assumptions. I want to make a comment if the observations are temporally correlated the analysis becomes much more complex. Therefore, in trying to solve the problem we would like to be able to make assumptions that are necessary to make the problem nontrivial. If you keep adding various levels of assumption the model the problem statement becomes so difficult we may not be able to mathematically solve it in a meaningful way.

So, what is the trick in modeling? You make all the necessary assumption to make the problem interesting, but at the same time not too complex. So, if you analyze the standard versions of the problem you get a feel for the solution based on which we can then tag on other assumptions other conditions to further explore the impact of assumptions; like if what if the observation are serially correlated and so on.

(Refer Slide Time: 05:43)



COST FUNCTION

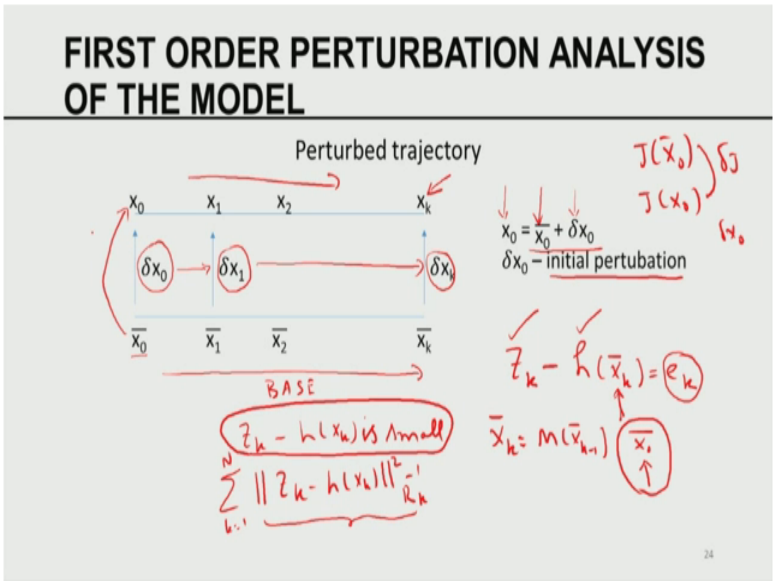
- $J(x_0) = \frac{1}{2}\sum_{i=1}^{N}(Z_k - h(x_k))^T R_k^{-1}(Z_k - h(x_k))$ -> (20)
- Problem: Min $J(x_0)$ when x_k is related to x_0 through the model dynamics
- While we could use the Lagrangian multiplier method, we illustrate an alternate strategy known as the adjoint method

23

So, the cost function is again J of x naught it is given by Z minus h of x k transpose R k inverse Z k minus h of x k. So, this is the forecast Z of x k is the forecast error. So, this is the sum of weighted sum of squared errors. The problem is to be able to minimize the J of x naught; where x naught is related to x k is related to x naught through the model dynamics. Again it is an implicit minimization problem. In the linear case we could replace x k implicit explicitly by f x k is equal to M to the power of k x naught. In the nonlinear case it is much more difficult. So, we need to be able to handle this case with little delicacy, which little bit delicately not delicacy; with a little bit delicately I am sorry delicately. So, while we could use the Lagrangian multiplier method we illustrate an alternate strategy called the adjoint method.

I would like to distinguish between adjoint equation and adjoint method. The adjoint method is a class of algebraic methods by which I can compute the gradient rather explicitly in a beautiful way.

(Refer Slide Time: 07:08)



FIRST ORDER PERTURBATION ANALYSIS OF THE MODEL

Perturbed trajectory
x_0 x_1 x_2 x_k
δx_0 δx_1 δx_k
$\bar{x}_0$ $\bar{x}_1$ $\bar{x}_2$ $\bar{x}_k$
BASE
$z_k - h(x_k)$ is small
$\sum_{k=1}^{N} \| z_k - h(x_k) \|^2_{R_k^{-1}}$
$x_0 = \bar{x}_0 + \delta x_0$
δx_0 – initial pertubation
$J(\bar{x}_0)$
$J(x_0)$
δJ
$z_k - h(\bar{x}_k) = e_k$
$\bar{x}_k = M(\bar{x}_{k-1})$
$\bar{x}_0$
24

So, first order perturbed analysis of the model, is key to the adjoint method that we are dealing with. So, let x bar naughtt be a base initial state from that I can generate a trajectory which is called the base trajectory. If you give me the base trajectory I can compute h of x k bar as the model predicted observation. The actual observations are Z k. If Z k is very close to h of x k then x k bar arising from x naught is pretty good and x naught bar could be a desonential condition to start the forecast from.

But, but in general Z of x k may not be equal to h of x k, Z of x k may be large Z k minus h of x k is equal to e k and e k will be far from being small. If e k is far from being small that means, I have a large forecast error. Our aim is to be able to annul the forecast error or reduce the forecast error. The forecast is x k bar, the forecast was generated from x naught bar. So, how do I; h is known fixed Z k is known fixed. So, how do I reduce the forecast error? The only way to reduce the forecast error is to x k bar. But how do I change x k bar? X k bar is the state of the model at time k, but x k bar depends on even the model equation is fixed. So, I would like to remind you x k bar is equal to M of x k minus 1.

So, that is the model equation. So, starting from x naught bar I calculated x k bar. So, look at this now my aim is to minimize or reduce the forecast error e k, the only way to reduce forecast error is to change x k, the only way to change x k is to change x naught. So, x naught is called the control element. It is called control because by changing the initial condition x naught bar I can change x k, if I change x k I can hope to change the forecast error.

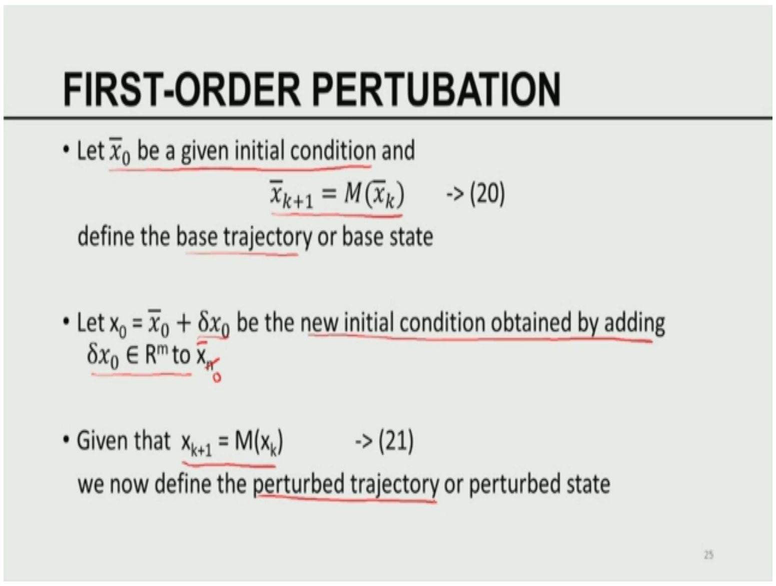
So, with this in mind I am now going to change the initial condition by adding a correction delta x naught to x bar naught. So, this is the old initial condition, this is the correction or a perturbation term, this is x naught is a new initial condition. So, delta x naught is called the initial perturbation. So, I am changing the initial condition from x bar naught to x naught. Now if I run the model forward in time from x naught I will get x 1 from x 2 x k. So, what is the hope? Z k minus h of x k must be smaller than e k I have already computed. If this can be made smaller by moving the initial condition then I am going the right direction to be able to reduce the forecast error. So, here is lies the idea of basic perturbation analysis.

So let me summarize: the only way to reduce the forecast error is to be able to change the model solution, the only way to change the model solution for a given model is to change the initial condition; I am assuming everything else in the model are fixed. So, by changing the initial condition means I am going to add a perturbation to the initial condition. So, this is the base trajectory, this is the base trajectory, this is the perturbed trajectory.

I would like to be able to find an optimal perturbation at an optimal increment in the initial stage such that Z k minus h of x k is small; small in what sense; in the sense of least squares. So, I would like to be able to minimize the norm the weighted norm of Z k minus h of x k where the weighted norm is R k inverse the square of the norm. So, what is this? This is the weighted sum of square errors at a given time by if I sum it up over k is 1 to N I get the total errors coming forecast errors coming from all the observations. So, I would like to be able to decide how I should perturb the initial condition to be able to annihilate the forecast errors at specified in sufficient time.

So, that is how we can look at the 4-D war problem. I hope the statement and the view of the problem is clear right now.

(Refer Slide Time: 12:52)



FIRST-ORDER PERTUBATION

- Let $\bar{x}_0$ be a given initial condition and

$$\bar{x}_{k+1} = M(\bar{x}_k) \quad \text{-> (20)}$$

define the base trajectory or base state

- Let $x_0 = \bar{x}_0 + \delta x_0$ be the new initial condition obtained by adding $\delta x_0 \in R^m$ to $\bar{x}_0$

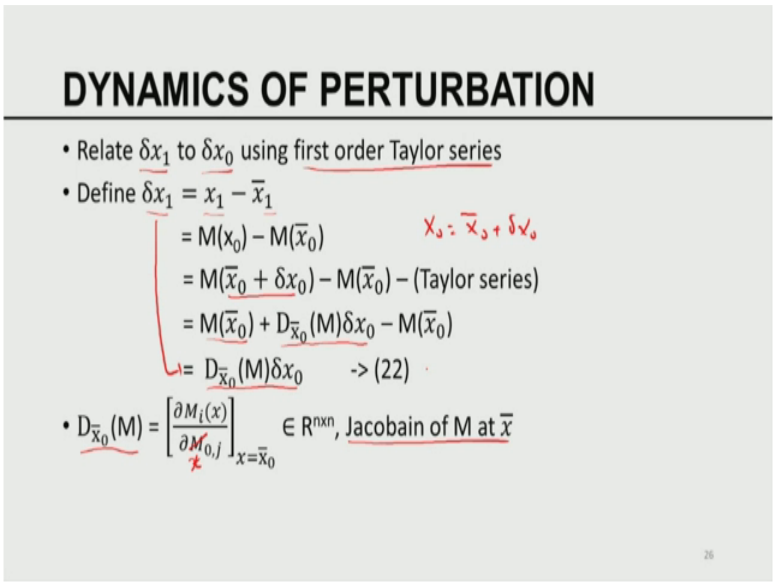
- Given that $x_{k+1} = M(x_k) \quad$ -> (21)

we now define the perturbed trajectory or perturbed state

So, I am now going to be talking about first order perturbation theory. This is also called variational analysis that is why the notion of 4-D war variational method came into being. So, first order perturbation analysis how does it go? Let x naught be the initials; let x naught bar be the initial condition. This is the base state given by 20, the base trajectory. I am going to perturb the initial condition, I am going to get the new initial condition by adding a perturbation to to x naught bar; I am sorry this is x naught bar

So, I can now compute the perturbed trajectory; the perturbed trajectory is given by the equation 21. So, the base trajectory is given by 20; the perturb trajectory is given by 21.

(Refer Slide Time: 13:51)



DYNAMICS OF PERTURBATION

- Relate δx_1 to δx_0 using first order Taylor series
- Define $\delta x_1 = x_1 - \bar{x}_1$

$$= M(x_0) - M(\bar{x}_0) \qquad X_0 = \bar{X}_0 + \delta X_0$$

$$= M(\bar{x}_0 + \delta x_0) - M(\bar{x}_0) - \text{(Taylor series)}$$

$$= M(\bar{x}_0) + D_{\bar{x}_0}(M)\delta x_0 - M(\bar{x}_0)$$

$$= D_{\bar{x}_0}(M)\delta x_0 \qquad \text{-> (22)}$$

- $D_{\bar{x}_0}(M) = \left[\frac{\partial M_i(x)}{\partial \overset{x}{\cancel{M}}_{0,j}}\right]_{x=\bar{x}_0} \in R^{nxn}$, Jacobain of M at $\bar{x}$

26

Now, what is the whole idea? Let me go back to the picture now. If x naught is the perturbation at time 0; let x 1 be the perturbation at time 1, delta x k be the perturbation at time k. Delta x naught is the first perturbation I am giving at the initial condition that induces a perturbation at time 1, that induces a perturbation at time k. So, what is our first goal? Our first goal is to find out how delta x k transforms to delta x 1, how delta x 1 transforms to delta x k. In other words I am interested in understanding the dynamics of propagation of the initial perturbation through the model. So, that is the first task.

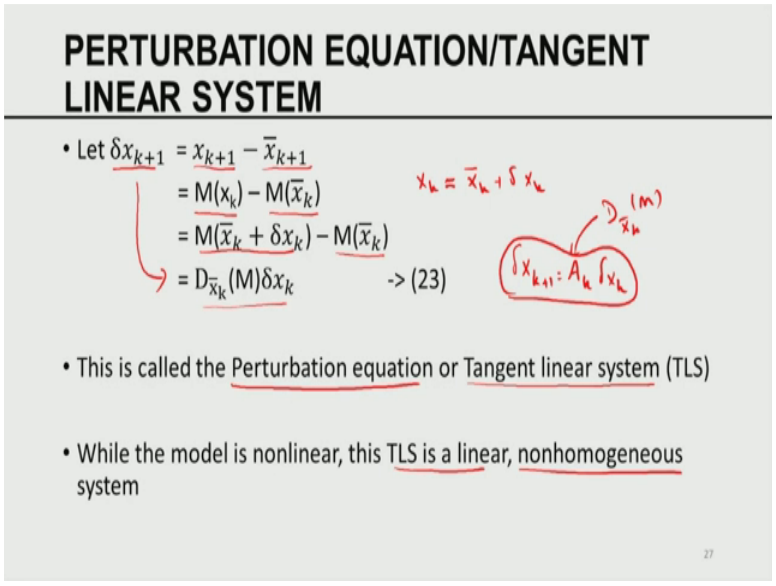
So, in time to write the dynamics of perturbation let us consider how delta x 1 is related to delta x naught. Please understand delta x 1 is the induced perturbation at time 1, induced by the initial perturbation delta x naught at time 0 to this to compute delta x 1 from delta x naught we are going to invoke to the first order Taylor series. We have already talked about first started Taylor series for maps in one of the previous modules.

So, delta x 1 by definition is equal to the perturbed state at time 1 with the unperturbed state at the base state at time 1, x 1 is M of x naught x 1 bar is M of x naught bar, but x naught we already know x naught is equal to; we already know x naught is equal to x naught bar plus delta x naught. So, that is what delta x naught is all about.

I am now going to apply the first order Taylor series expansion for the first term. When I apply the first order Taylor series expansion I get M of x naught bar plus the Jacobian of M at x naught times delta x naught minus M of x naught that first and the third term cancels; I get delta x 1 is equal to the Jacobian of M at x naught times delta x naught. Please recall the Jacobian is a n by M matrix, Jacobian of M is n by M matrix it is the partial derivative of the i-th component with respect to this I must say the following this is x I am sorry this is not M this is x; x of J naught. So, x naught of J is the j-th component of the initial condition M i is the i-th component of M. So, I varies from 1 to N J varies from 1 to N. So, that is the Jacobian of M at the initial base state.

So, if you know M i can compute the Jacobian, I can evaluate the Jacobian; therefore, you can readily see equation 22 that relates the perturbation at time 1 to the perturbation at time 0 is a linear relation. So, even though the model is nonlinear the perturbation dynamics is linear. That comes out very easily from equation 22.

(Refer Slide Time: 17:17)



PERTURBATION EQUATION/TANGENT LINEAR SYSTEM

- Let $\underline{\delta x_{k+1}} = \underline{x_{k+1}} - \underline{\bar{x}_{k+1}}$
  $= \underline{M(x_k)} - \underline{M(\bar{x}_k)}$
  $= \underline{M(\bar{x}_k + \delta x_k)} - \underline{M(\bar{x}_k)}$
  $= \underline{D_{\bar{x}_k}(M)\delta x_k}$ -> (23)

$x_k = \bar{x}_k + \delta x_k$

$D_{\bar{x}_k}^{(M)}$

$\boxed{\delta x_{k+1} = A_k \delta x_k}$

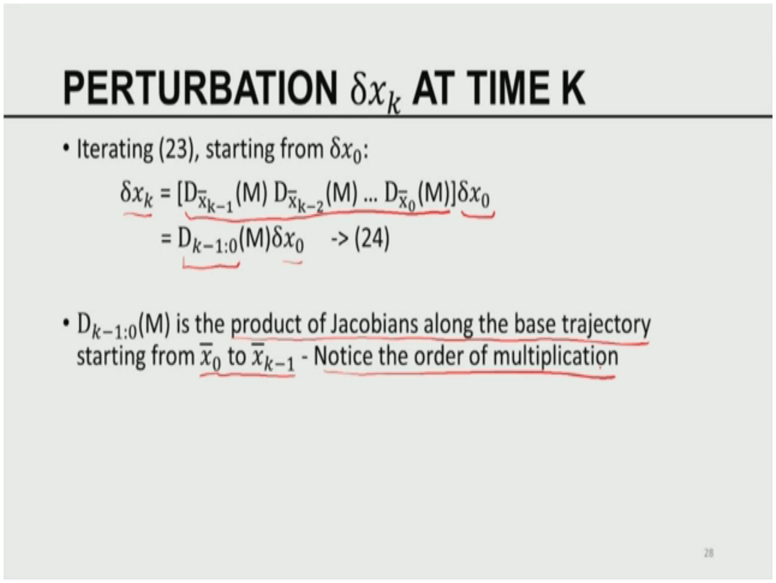
- This is called the Perturbation equation or Tangent linear system (TLS)

- While the model is nonlinear, this TLS is a linear, nonhomogeneous system

27

In general, let the delta x k plus 1 be the perturbation at time k plus 1. This is equal to x k plus 1 minus x bar k plus 1, this is M of x k, M of x k plus x bar k.Nnow please remember x k to a first order approximation is equal to x bar k plus delta x k. So, that is essentially this. Then you have the M of x bar k. By applying the first order Taylor series you get this. Now, you can see this is the perturbation dynamics. This dynamics in mathematics called variational equation; this is also called perturbation equation, but in meteorological circle they have a very different name called tangent linear system they call it TLS. So, TLS is a linear system. It is a non-homogeneous system, because the Jacobian matrix varies along the base state. So, even though this is the matrix the matrix changes in time. Therefore, you can try to relate delta x as this is equal to A k times delta x k; where A k is the Jacobian at x k of m. So it varies with time, therefore this the nonhomogeneous system is a linear system it is a non homogeneous linear dynamics that gives you the evolution of the perturbation; this is the first order perturbation equation.

(Refer Slide Time: 19:12)



PERTURBATION δx_k AT TIME K

- Iterating (23), starting from δx_0:

$$\delta x_k = [D_{\bar{x}_{k-1}}(M)\, D_{\bar{x}_{k-2}}(M) \dots D_{\bar{x}_0}(M)]\delta x_0$$
$$= D_{k-1:0}(M)\delta x_0 \quad \text{-> (24)}$$

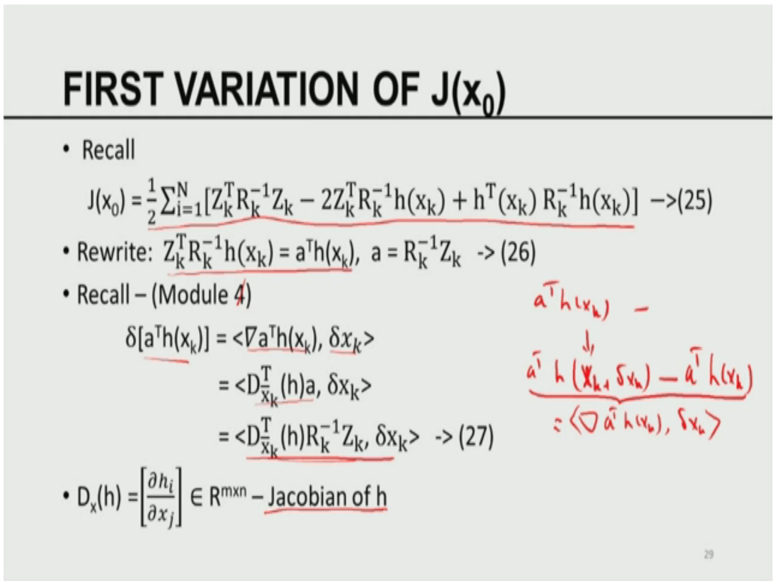
- $D_{k-1:0}(M)$ is the product of Jacobians along the base trajectory starting from $\bar{x}_0$ to $\bar{x}_{k-1}$ - Notice the order of multiplication

28

By iterating 23; so by iterating this equation you can really see delta x k is equal to the product of Jacobian's along the trajectory; the product of Jacobian along the trajectory times the initial perturbation. I am now going to concoct a new solution; a new notation D subscript k minus 1 colon 0 to M essentially represents this product. This is the product of k matrices going from time 0 to time k minus 1. So, with this notation you can readily see delta x k the perturbation at time k is related to the perturbation at time 0 through the product of Jacobian's, the product taken along the trajectory starting from x 0 1 2 3 all the way up to k minus 1. This matrix is the product of Jacobian's along the base trajectory starting from x naught bar to x k minus 1 bar: notice the order of multiplication. Please remember matrix multiplication is not commutative, therefore you have to take great care in trying to when you want to involve matrix matrix multiplication you have to worry about the order and be sensitive to the order.

Now come back to; so what is that we have done, absolutely quickly summarize. We stated the problem, we saw the problem of forecast correction is equal to changing the initial condition, change the initial condition is equal to adding perturbation to the initial condition, so we were interested in trying to quantify how an initial perturbation propagated through the model in time. So, we have simply the original model equation, we also have the forward dynamics which is the tangent linear system that describes the evolution of the perturbation forward in time. That is what we have accomplished so far; sorry I am going back instead of going forward.

(Refer Slide Time: 21:20)



FIRST VARIATION OF $J(x_0)$

- Recall

$$J(x_0) = \frac{1}{2}\sum_{i=1}^{N}[Z_k^T R_k^{-1} Z_k - 2Z_k^T R_k^{-1} h(x_k) + h^T(x_k) R_k^{-1} h(x_k)] \rightarrow (25)$$

- Rewrite: $Z_k^T R_k^{-1} h(x_k) = a^T h(x_k)$, $a = R_k^{-1} Z_k$ -> (26)
- Recall – (Module 4)

$$\delta[a^T h(x_k)] = \langle \nabla a^T h(x_k), \delta x_k \rangle$$
$$= \langle D_{x_k}^T(h)a, \delta x_k \rangle$$
$$= \langle D_{x_k}^T(h) R_k^{-1} Z_k, \delta x_k \rangle \rightarrow (27)$$

- $D_x(h) = \left[\frac{\partial h_i}{\partial x_j}\right] \in R^{m\times n}$ – Jacobian of h

$$a^T h(x_k) -$$
$$\downarrow$$
$$\underbrace{a^T h(x_k + \delta x_k) - a^T h(x_k)}$$
$$= \langle \nabla a^T h(x_k), \delta x_k \rangle$$

29

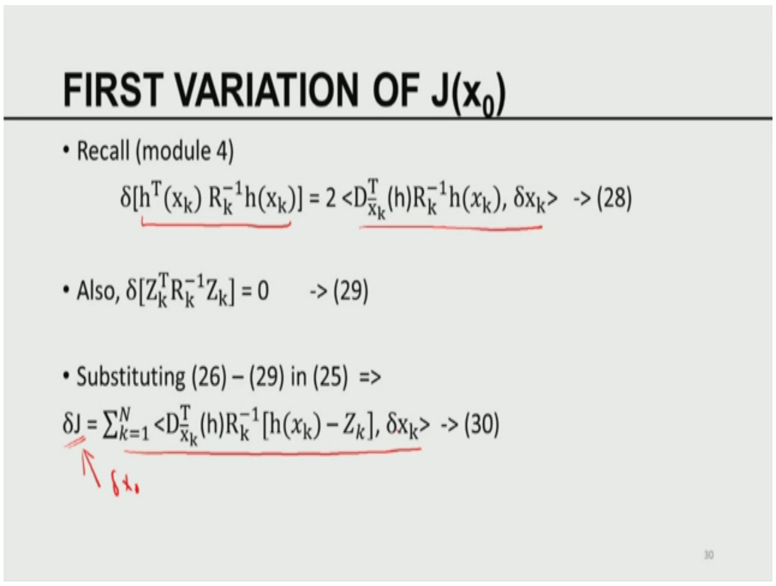
Now, I would like to come back to my J of x naught; the thing to be minimized. Please recall J of x naught is given by this summation, this is given in equation 20. So, equation 20 when you multiply that all the things now becomes equation 25. 25 is simply a rewritten version of 20. Now I would like to change the notation to make the application of multivariate calculus lot easier. So, let me call the linear term like this, let me call a is R k inverse Z k.

From module 4 I do not think its model 4 the number is slightly different I will compute the mark of the number later. Recall from one of the modules and multivariate calculus which is two point something. I am now concerned with what is called the first variation, you remember the notion of her to the first variation. The first variation is simply the inner product of the gradient and the variation x k. So, the first variation of a transpose h of x k is the gradient of a transpose h of x k times delta x k. The, a transpose x k is given so its gradient is given by this equation; a by substituting this is given by this equation. Therefore, the first variation of the linear term is given by 27.

So, what does first variation means? I would like to be able to talk about that for a moment before we go further. I have a transpose h of x k. So, if I change x k to x k plus delta x k, and if I compute the difference between this and that term and that is what is called the difference between a transpose h of x k plus delta x k minus a transpose h of x k; that is called the actual difference. This difference is equal to the gradient of a transpose h of x k times delta x k; that comes from the basic definition of the notion of first variation.

So, how is the induced variation affects the linear term and that is going to be affected by this; where D h is the Jacobian of the forward operator h. I hope this idea of first variation is very clear. Why am I talking about the first variation? Look at this now: delta x k depends on delta x naught, I have already related delta x k to delta x naught through the tangent linear system. So, if I can relate the first variations of each of these terms with respect to delta x naught; of delta x k I am sorry I have already related delta x k to delta x naught, so I can relate the variation of delta x naught to the initial variations. So, that is the basic principle. That is not work here.

(Refer Slide Time: 25:06)



FIRST VARIATION OF $J(x_0)$

- Recall (module 4)

$$\delta[\underline{h^T(x_k)\, R_k^{-1} h(x_k)}] = 2 <\underline{D_{x_k}^T(h) R_k^{-1} h(x_k), \delta x_k}> \ \rightarrow (28)$$

- Also, $\delta[Z_k^T R_k^{-1} Z_k] = 0 \quad \rightarrow (29)$

- Substituting (26) – (29) in (25) =>

$$\delta J = \underline{\sum_{k=1}^{N} <D_{x_k}^T(h) R_k^{-1}[h(x_k) - Z_k], \delta x_k>} \ \rightarrow (30)$$

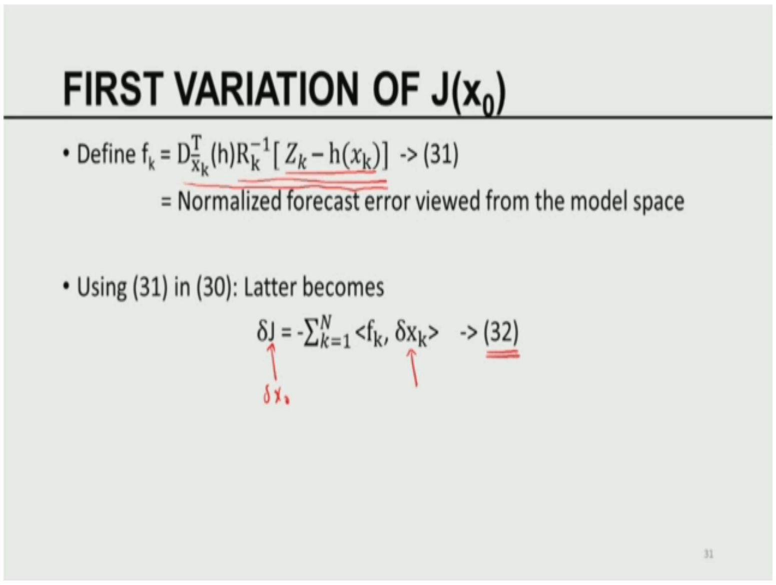
$\nwarrow \delta x_0$

30

If I consider the second term, the second term is a quadratic in h from the module on multivariate calculus; we already know the first variation of this quantity with respect to x k is given by the right-hand side of 28; the first variation of Z k transverse R k inverse Z k 0, because it does not depend on x k. So, substituting the whole thing we can readily see; substituting 25 through 29 and 25 the first variation in the cost function induced by delta x naught is given by this expression in equation 30.

Now the right-hand side depends only on delta x k, but the left-hand side I am talking about variation induced by delta x naught, through the tangent linear system I already know delta x k is related to delta x naught. So, combining tangent linear system and 30 I now know how to relate the first variation in J to delta x naught. So, that is the line of arguments.

(Refer Slide Time: 26:36)



FIRST VARIATION OF $J(x_0)$

- Define $f_k = D^T_{\bar{x}_k}(h) R_k^{-1} [\underline{Z_k - h(x_k)}]$ -> (31)
  = Normalized forecast error viewed from the model space
- Using (31) in (30): Latter becomes

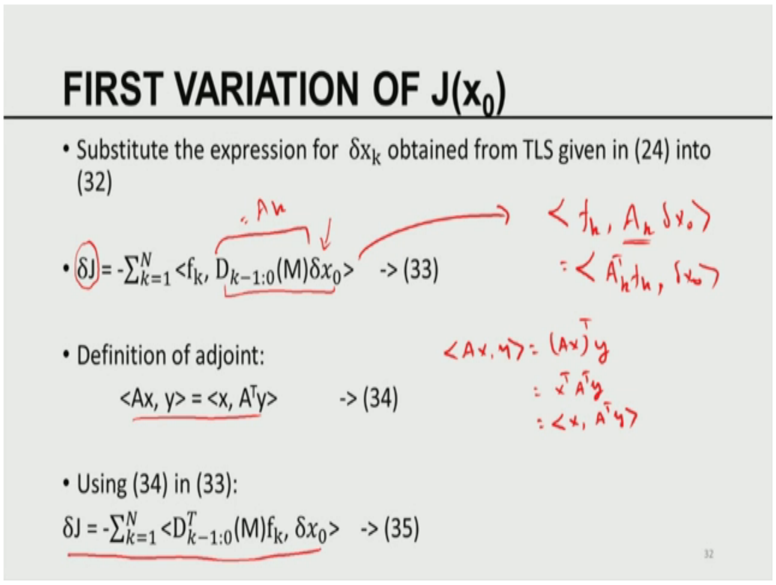
$$\underset{\delta x_0}{\delta J} = -\sum_{k=1}^{N} \langle f_k, \delta x_k \rangle \quad \text{->} \ \underline{(32)}$$

31

Again I am going to define to simplify notations f of k is equal to the forecast error; this is the forecast error, this is the normalized forecast error, this is the forecast normalized forecast error viewed from the model space, this is same f of k as we have used in the previous case. So, I can rewrite the equation this equation 30 succinctly as the variation in the variation sorry; the variation in delta J induced by delta x naught to the sum of the variations induced along the path all with respect to delta x k, k running from 1 to N recognizing the fact delta x k and delta x naught are related through the tangent linear system and that is 32. So, this is the first and the most fundamental relation in adjoint approach to 31.

So, let me go back to the picture to illustrate a little bit further, I think it is very useful. So, I when I change from x bar naught to x naught my J function changes. So, originally I had sorry; I originally I had J of x naught bar now I have J of x naught's the difference between the two is delta J, this delta J is induced by delta x naught. Therefore, I would like to be able to talk about the induced change in J at a given time through the initial condition and that is what we have a complete so far through this variational analysis through the tangent linear system if we wish. So, that is our.

(Refer Slide Time: 28:50)



FIRST VARIATION OF $J(x_0)$

- Substitute the expression for δx_k obtained from TLS given in (24) into (32)
- $\boxed{\delta J} = -\sum_{k=1}^{N} \langle f_k, \overset{A_k}{\underline{D_{k-1:0}(M)}}\delta x_0 \rangle$ -> (33) $\quad \langle f_k, \underline{A_k} \delta x_0 \rangle = \langle A_k^T f_k, \delta x_0 \rangle$
- Definition of adjoint:

  $\underline{\langle Ax, y \rangle} = \langle x, A^T y \rangle$ -> (34) $\quad \langle Ax, y \rangle = (Ax)^T y = x^T A^T y = \langle x, A^T y \rangle$
- Using (34) in (33):

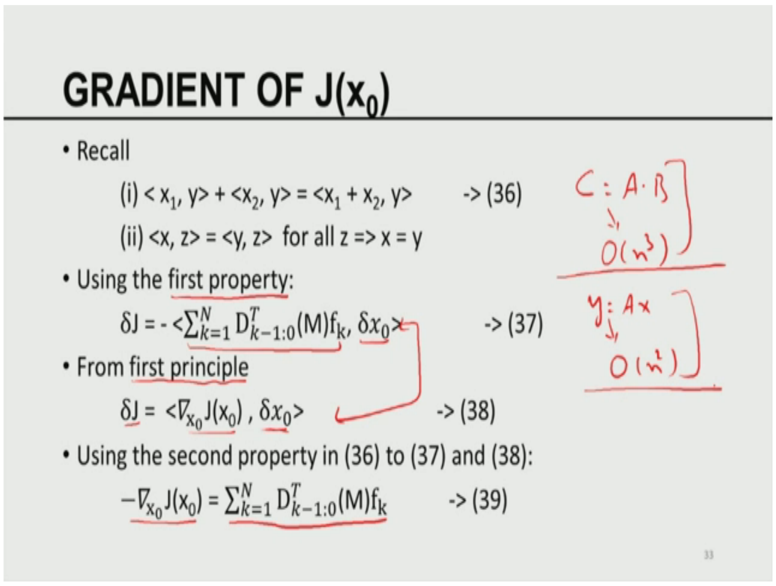
  $\underline{\delta J = -\sum_{k=1}^{N} \langle D_{k-1:0}^T(M) f_k, \delta x_0 \rangle}$ -> (35)

Now, I would like to remind you that delta x k is given by the tangent linear system in 24. So, let me go back to 24 and remind you what is there. So, 24 essentially relates delta x k to delta x naught through the product of the Jacobian's; so 24. So, using 24 I am changing dependence from delta x k to delta x naught, let me come back here. So, I am going to substitute for delta x naught here, if I did that this is the expression I get; that is the expression I get. Now you can readily see delta J induced by a change in the initial condition is related directly to the change the initial condition itself; everything becomes explicit.

Now we are going to go back to matrix algebra to define the notion of an adjoint operator. So, if A is a matrix then the product of A x comma y; the inner product of A x comma y by definition is A x transpose y which is equal to x transpose A transpose y which can be written as x comma A transpose y. So, that is what this relation is. Therefore, the inner product A x with y is the same as inner product of x with A transpose y. So, a transpose is called the adjoint of A; the transpose of a matrix is called the adjoint of A. So, that is the definition of the adjoint of A operator or a matrix A.

So, using this adjoint property I can now apply to 33. So, you can think of 33 as f of k comma A of k delta x naught which by adjoint property. So, let A of k be equal to this matrix, let that be equal to A of k. Therefore, by applying the adjoint property in the reverse way I can essentially say this is A k transpose f of k at delta x naught; this is delta x naught. So, A k transposes D of k minus 1 colon 0 transpose f of x k. So, by applying the adjoint property 34 to 33 I get this relation. This relation is one of the most fundamental relation the method is called variational methods because I am talking about variations induced by the initial variation, I am also talking about this adjoint method because I am using the adjoint property of matrices. So, is adjoint and variational schemes all put together it uses many number of interesting properties that join from matrix theory.

(Refer Slide Time: 31:57)



GRADIENT OF $J(x_0)$

- Recall
  - (i) $< x_1, y> + <x_2, y> = <x_1 + x_2, y>$ -> (36)
  - (ii) $<x, z> = <y, z>$ for all $z \Rightarrow x = y$
- Using the first property:
  - $\delta J = - <\sum_{k=1}^{N} D_{k-1:0}^{T}(M)f_k, \delta x_0>$ -> (37)
- From first principle
  - $\delta J = <\nabla_{x_0} J(x_0), \delta x_0>$ -> (38)
- Using the second property in (36) to (37) and (38):
  - $-\nabla_{x_0} J(x_0) = \sum_{k=1}^{N} D_{k-1:0}^{T}(M)f_k$ -> (39)

$C : A \cdot B \rightarrow O(n^3)$

$y : Ax \rightarrow O(n^2)$

33

Recall some of the basic things from again linear algebra. If I have the inner product x 1 plus y plus x 2 plus y that is equal to x 1 plus x 2 with y. If I have x is inner product x and Z is equal to inner product of y Z for all Z then that implies x must be equal to y; it comes very. So, first is a linearity property, second is the property that essentially tells you if the inner product of x with Z is equal to the inner product of y with Z for all Z x better be equal to y. So, that is the basic property.

By applying the first property the sum of the inner product is equivalent to the inner product of the sum. So, inner product of the I am sorry; this is the inner product of the sum with x naught. So, I have applied the first property. The first property is sum of the inner product is inner product to the sum; that is essentially. So, if I have your first variation is given with the inner product also from first principles from variational calculus we have seen in multi calculus delta J is equal to the gradient times delta x naught. So, I am now going to have two different expressions related to delta J and these two expressions are equal for all delta x naught. Delta x naught is arbitrary, therefore, by applying the second property I can readily now conclude the negative with the gradient is given by the sum which is summation k is equal to 1 to N transpose of the product of matrices along the trajectory times f of k and f of k is the forecast error mute for the model space. That is the whole idea.

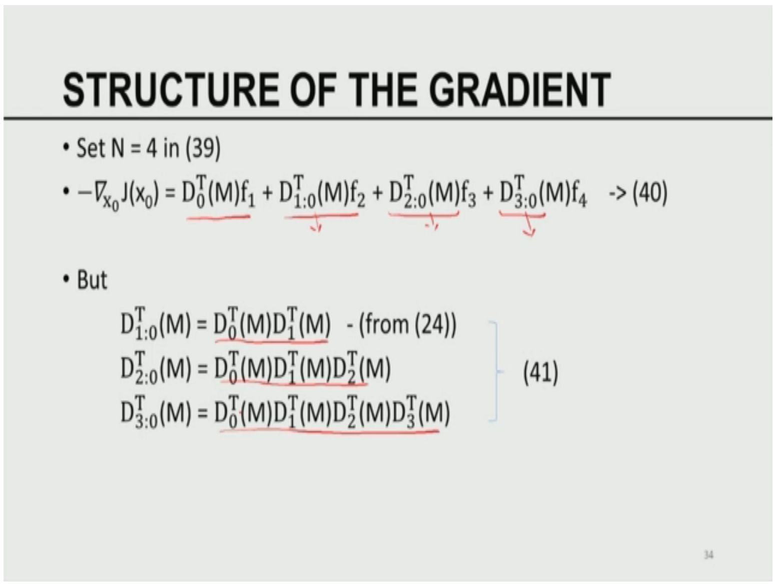
My idea is to be able to find an expression for the gradient and we have achieved it. So, instead of using Lagrangian multiplier and equating the gradient we have worked the path of being able to use the variational analysis; the variation analysis is very intuitive in the sense that the only way to be able to correct the forecast error is to change the initial condition; change the initial condition is equal to perturbing the initial condition if I perturb the initial condition the initial perturbation propagates through the model. So, if I can quantify to a first order accuracy the propagation of model errors or model perturbations along the trajectory I have handle on one of the issues to be tackled using tangent linear system. The other part of the story is to be able to compute the induced first variation in the cost function itself, then we relate these two first variations. In trying to relate this we use several properties one is the adjoint property another is the linearity property another is the property where the inner product of x and Z and y and Z are the same for all Z x is equal to y all these three basic properties are intimately related and they are applied. And that essentially gives you an expression for the gradient which is given in 39.

Now look at that everything on the right hand side of 39 is computable f of k is computable the Jacobian of the model is computable, but what is the only thing that we do not like it involves product of Jacobian's. Jacobian's are matrices matrixes; matrix product is a no. So, you never multiply matrices unless that is what you want, you never invert a matrix unless that is what you want. These are some of the golden rules in numerical analysis in numerical in linear algebra, because these are too expensive operations.

So, in theory you may involve matrix matrix multiplication, but when you practice you should avoid as much as possible whenever wherever possible. So, even though we have got an x 39 for the gradient still we are in the same boat as we were in the linear analysis this still involves matrix matrix multiplication, we want to avoid matrix matrix multiplication; matrix vector multiplication is cheaper than matrix matrix multiplication let me talk about it for a moment now. If I have two matrices if I want to multiply C is equal to A B the cost is o of n cube. If I have y is equal to A x when a is the matrix x is a vector this matrix vector computation is o of n square. o of n square is much cheaper than o of n cube. So, whenever there is a matrix matrix operation avoided, whenever you can get away with matrix vector multiplication instead of matrix matrix multiplication you introduce them.

Now, we are going to rewrite the computation of 39 without doing any matrix matrix multiplication, but involving only matrix vector multiplication very intelligently. So, we are going to talk about in a very efficient way this way is essentially the adjoint equation we already saw. So, the rest of the problem rests in computing 39 efficiently essentially involving only matrix vector multiplication.

(Refer Slide Time: 37:22)



STRUCTURE OF THE GRADIENT

- Set N = 4 in (39)
- $-\nabla_{x_0} J(x_0) = \underline{D_0^T(M)f_1} + \underline{D_{1:0}^T(M)}f_2 + \underline{D_{2:0}^T(M)}f_3 + \underline{D_{3:0}^T}(M)f_4$ -> (40)

- But

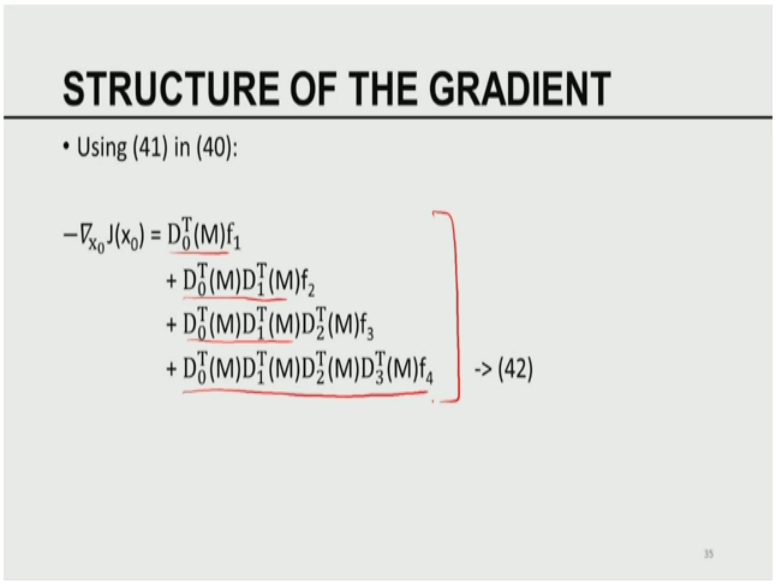
$$\left.\begin{aligned} D_{1:0}^T(M) &= \underline{D_0^T(M)D_1^T(M)} \quad \text{- (from (24))} \\ D_{2:0}^T(M) &= D_0^T\underline{(M)D_1^T(M)D_2^T}(M) \\ D_{3:0}^T(M) &= D_0^T\underline{(M)D_1^T(M)D_2^T(M)D_3^T}(M) \end{aligned}\right\} \quad (41)$$

34

To bring this, to bring the structure of the expression 39 let us assume n is 4 just to be able to; look at how this one comes up. This is in here we have involved a product of two Jacobian's, in here they are involved with the product of three Jacobian's, in here we are involved with the product of four Jacobian's that is how the terms come in. So, please understand these Jacobian's these are all product of Jacobian's, from the computational point of view the Jacobian's if you did it like this that is deadly.

So, 41 is simply an explanation, but 41 while it is meaningful to interpret it is computationally a disastrous. But to understand the kind of challenge we have I have simply expressed 39 in a simple form of n is equal to 4 and expresses 40, so you can you can see how much of competition involved in computing the gradient. Now how do we avoid a matrix matrix multiplication and introduce matrix vector multiplication that is done very cleverly by what is called the adjoint equation. So, let us look at 41.

(Refer Slide Time: 38:38)



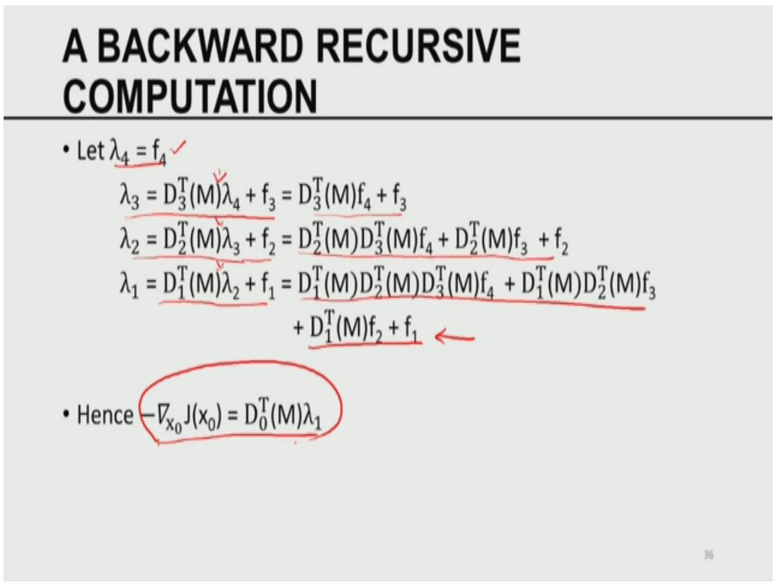
STRUCTURE OF THE GRADIENT

- Using (41) in (40):

$$\left.\begin{aligned} -\nabla_{x_0} J(x_0) = {} & \underline{D_0^T(M)}f_1 \\ & + \underline{D_0^T(M)D_1^T(M)}f_2 \\ & + \underline{D_0^T(M)D_1^T(M)}D_2^T(M)f_3 \\ & + \underline{D_0^T(M)D_1^T(M)D_2^T(M)D_3^T(M)f_4} \end{aligned}\right] \rightarrow (42)$$

So, in this case the gradient is given by the sum of these products I am sorry; sum of these products; sum of these products. Now look at look at the beauty of the structure of this now. We are going to successively expand the matrix matrix multiplication involved in here recursively. So, this is this is the structure I want to be able to compute efficiently. In order to compute this efficiently again I am going to give an example.

(Refer Slide Time: 39:22)



A BACKWARD RECURSIVE COMPUTATION

- Let $\underline{\lambda_4 = f_4}$ ✓

$$\lambda_3 = \underline{D_3^T(M)\lambda_4 + f_3} = \underline{D_3^T(M)f_4 + f_3}$$

$$\lambda_2 = \underline{D_2^T(M)\lambda_3 + f_2} = \underline{D_2^T(M)D_3^T(M)f_4 + D_2^T(M)f_3} + f_2$$

$$\lambda_1 = \underline{D_1^T(M)\lambda_2 + f_1} = \underline{D_1^T(M)D_2^T(M)D_3^T(M)f_4 + D_1^T(M)D_2^T(M)f_3}$$

$$+ \underline{D_1^T(M)f_2 + f_1} \leftarrow$$

- Hence $\boxed{-\nabla_{x_0}J(x_0) = D_0^T(M)\lambda_1}$

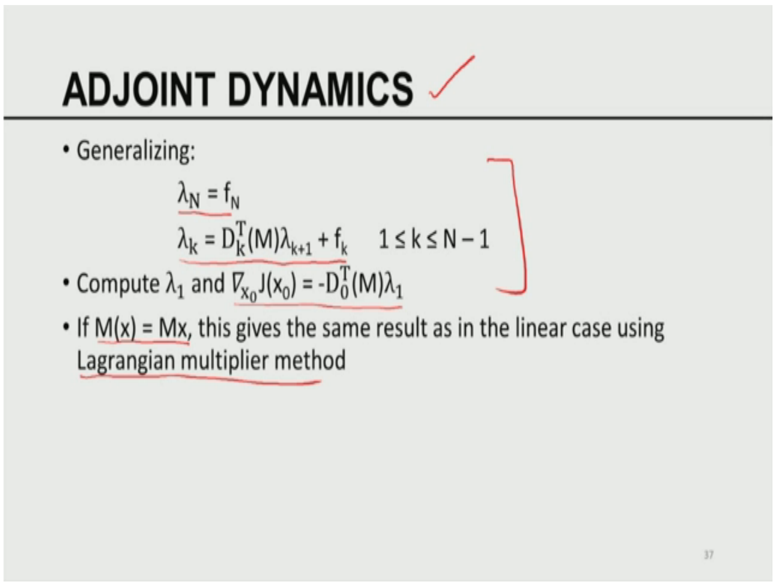
36

Let lambda 3; so, I am going to introduce some new variables. These variables lambdas are related are somewhat related to the Lagrangian multiplier that we have seen earlier. So, let me set lambda 4 is equal to f of 4; lambda 4 is equal to f of 4. Then I will compute lambda 3 to be equal to D 3 transpose lambda 4 plus f 3. So, that gives rise to this formula. Lambda 2 is equal to D 2 transpose lambda 3 plus f 2 that give rise to this expression. Lambda 1 is equal to D 1 transpose lambda 2 that gives rise to this expression, you can see that is delta x.

Therefore, when I have only four observations I can compute the gradient simply by involving matrix vector multiplication. What is the matrix vector multiplication? Look at this now. Lambda 4 is f 4 there is no matrix multiplication. To compute lambda 3 this is D 3 transpose M is a matrix lambda is a vector. So, there is matrix vector multiplication addition of a vector: one matrix vector multiplication addition of a vector, one matrix vector multiplication addition of a vector.

Now if I did this like this involving only matrix vector operation you can see by iterating backward in time I have gotten the explicit expression for the gradient. So, what does this suggest? This suggests that we can avoid matrix matrix multiplication by introducing a backward recurrence relation much like the adjoint equation that we had earlier and that is the essence of the story.

(Refer Slide Time: 41:15)



ADJOINT DYNAMICS ✓

- Generalizing:

$$\lambda_N = f_N$$

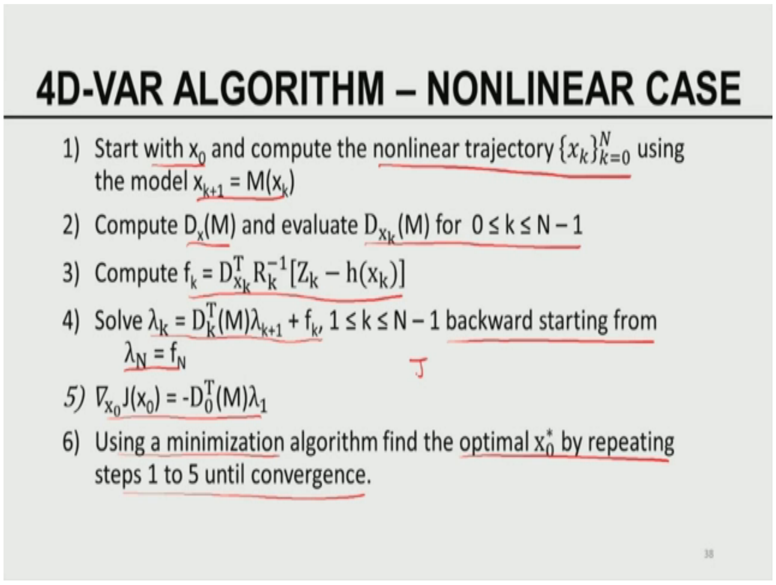
$$\lambda_k = D_k^T(M)\lambda_{k+1} + f_k \quad 1 \le k \le N-1$$

- Compute λ_1 and $\nabla_{x_0} J(x_0) = -D_0^T(M)\lambda_1$
- If $M(x) = Mx$, this gives the same result as in the linear case using Lagrangian multiplier method

Therefore generalizing: I have given you a specific example to be able to get a handle on the problem with n is equal to 4. For a general n I am going to define lambda N is equal to f of N, I am going to define lambda k using D k transpose M lambda k plus 1 plus f of k. So, compute from lambda N to lambda 1: if you compute lambda 1 then you can come you can verify that the gradient of J x naught with refer to x naught is given by minus D 0 transpose M delta lambda 1. If M of x is equal to M x; that means if the model is a linear you can see this equation reduces to the one that we did in the Lagrangian multiplier technique.

Therefore, this relation is a generalization of the backward adjoint dynamics that we considered for the Lagrangian multiplier technique; that is why this is called adjoint. Again at adjoint dynamics this whole idea of using first the propagation of the perturbation adjoint principles from matrix theory and combination of adjoint dynamics is this in the context of nonlinear system provides a very simple and a elegant means of computing the gradient of the cost function with respect to the initial condition.

(Refer Slide Time: 42:34)



4D-VAR ALGORITHM – NONLINEAR CASE

1) Start with x_0 and compute the nonlinear trajectory $\{x_k\}_{k=0}^{N}$ using the model $x_{k+1} = M(x_k)$
2) Compute $D_x(M)$ and evaluate $D_{x_k}(M)$ for $0 \leq k \leq N-1$
3) Compute $f_k = D_{x_k}^T R_k^{-1}[Z_k - h(x_k)]$
4) Solve $\lambda_k = D_k^T(M)\lambda_{k+1} + f_k$, $1 \leq k \leq N-1$ backward starting from $\lambda_N = f_N$
5) $\nabla_{x_0} J(x_0) = -D_0^T(M)\lambda_1$
6) Using a minimization algorithm find the optimal x_0^* by repeating steps 1 to 5 until convergence.

J

38

Once the gradient is available we can again use it in a general gradient or conjugate gradient method. So, here is this summary of the 4-D war algorithm for the nonlinear case. So, start with x naught, compute the nonlinear trajectory, only for the sake of illustration we use the base state and perturbation state and other things. Now that we have understood everything I am now going to encapsulate all the analysis into the form of a algorithm this is called 4-D war algorithm for the nonlinear case.

Start with the initial condition x naught, compute the nonlinear trajectory from the model. Then compute the Jacobian and evaluate the Jacobian along the model. Please understand computationally it is expensive why is that each Jacobian is a n by n matrix there are n square elements. So, I have to evaluate the n square elements of the matrix at each of the n instances in time. So, that is going to be a pretty demanding computationally.

So, the question comes in: do you compute these matrices and store them? That is a no, why? Storing these matrices which requires the n square space when n is of the order of million there is no space in the world; in computers to be able to store all these matrices for all time. So, what is that you do? You have a general expression for the matrix some place, you evaluate that matrix on demand whenever you need the value at a given state.

So, there is a notion of a spacetime trade off involved in implementing these and one has to be very clever in not to burn the space requirement by storing too many things. But I also want to recognize, if you can store few things you can speed up the computation, but the demands on the space becomes too large it makes no sense because it will eat up all the memory and then sum and you may not have much space left for doing anything else. So, great caution must be exercised in trying to be able to evaluate; you have to evaluate the Jacobian. The question is should I store them or not? You probably cannot afford to store them you simply compute them on the fly and use them wherever you want. That is the basic idea.

So, once the Jacobian's are computed I can compute f of k; f of k are vectors say f of k can be stored matrixes is expensive; f of k is an it could be a million by vector. Million vector is easier to store than million by million matrix. So, there is a tradeoff between storage in time. And now what is that I need to do? I need to be able to start the backward dynamics starting from f n is equal to lambda N and lambda k is equal to transpose of the Jacobian at time k; times lambda k plus 1 plus f of k. So, once I have lambda 1 I can compute the gradient by this simple formula in step five. Once I have the gradient I can compute the optimal x naught by simply repeating this until convergence using a minimization algorithm the gradient algorithm or the conjugate gradient algorithm.

But one thing I want to remind you the J function that is being minimized is not a quadratic function. I would like to be able to bring that part of the argument very carefully I want you to appreciate that the J of x naught- let us go back J of x naught is given in 20 h of x is a nonlinear function. So, therefore, if we multiply both sides it is not quadratic, it is much more nonlinear than quadratic. So, if the function J of x is not quadratic but much more nonlinear than quadratic; what does it mean? The performance of the gradient algorithm and conjugate gradient algorithm are guaranteed only on quadratic functions. The function that is of interest here is not a quadratic function.

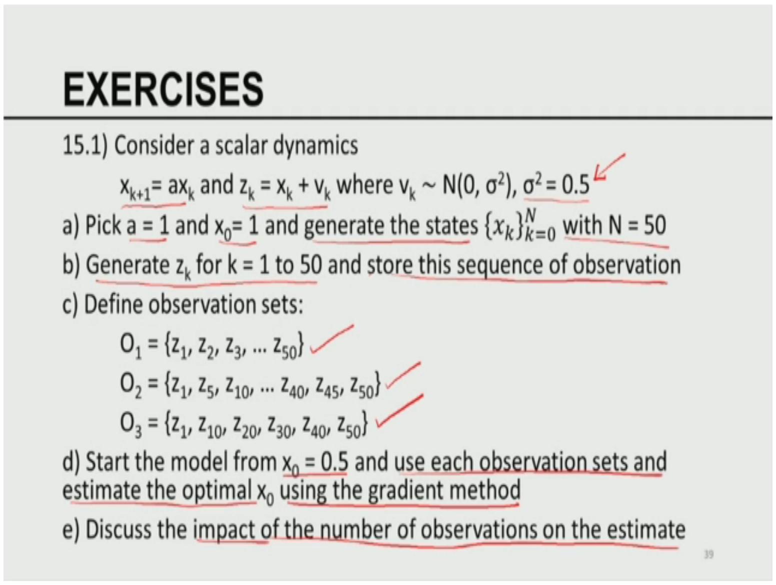
Therefore, the performance of gradient algorithm or the conjugate gradient algorithm for the nonlinear case model nonlinear and the observational nonlinear could take longer to minimize because are no general results guaranteeing convergence.

So, when to stop, how far to go all these things are questions whether one has to contend with in trying to decide what kind of stopping criterion one could use to stop the minimization process. That is a problem quite apart from the theory that we have developed. But I want you to understand that that is an integral part of the overall development of the package if you are interested in trying to program this.

So, with this we have come to the end of the introduction to 4-D war. So, let me quickly summarize a 4-D war or adjoint method is essentially a solution of the inverse problem within the dynamic context. Given a bunch of n observations noisy observations that contain information about the state I would like to be able to determine the optimal initial state we pose this as the minimization problem is the constrained minimization problem this can be solved in one of two ways; either as a strong constraint Lagrangian multiplier problem or by using variational methods. And I am trying to understand and quantify propagation of variation through the model and adjoint properties of adjoint operators.

So, I have illustrated these two complimentary methods of analysis by one using linear model linear observation another using nonlinear model nonlinear observation; this covers pretty large area of techniques that is known for solving deterministic dynamic inverse problem where the model is deterministic and assumed to be perfect, but the observations are noisy.

(Refer Slide Time: 48:56)



EXERCISES

15.1) Consider a scalar dynamics

$x_{k+1} = ax_k$ and $z_k = x_k + v_k$ where $v_k \sim N(0, \sigma^2)$, $\sigma^2 = 0.5$

a) Pick $a = 1$ and $x_0 = 1$ and generate the states $\{x_k\}_{k=0}^{N}$ with $N = 50$

b) Generate z_k for $k = 1$ to 50 and store this sequence of observation

c) Define observation sets:

$O_1 = \{z_1, z_2, z_3, \dots z_{50}\}$

$O_2 = \{z_1, z_5, z_{10}, \dots z_{40}, z_{45}, z_{50}\}$

$O_3 = \{z_1, z_{10}, z_{20}, z_{30}, z_{40}, z_{50}\}$

d) Start the model from $x_0 = 0.5$ and use each observation sets and estimate the optimal x_0 using the gradient method

e) Discuss the impact of the number of observations on the estimate

We would like to encourage you to perform several relay computation a related experiments. I am going to go over this in some detaill let us spend a couple of minutes on this. Consider a very simple scalar model x k plus 1 is equal to a times x k; consider an observation Z k is equal to V k. Please remember the model is linear the observation is linear; there is no h, h is identity is the simplest of the possible model. The model M is the matrix is simply a scalar a. V k is the observation covariance which is sigma square I am going to ask you to make take it as 0.5. So, my suggestion to you is to pick a is equal to 1, generate a sequence of state with n is equal to 50 generate observations from one to 50 by adding noise to that store this sequence of observations.

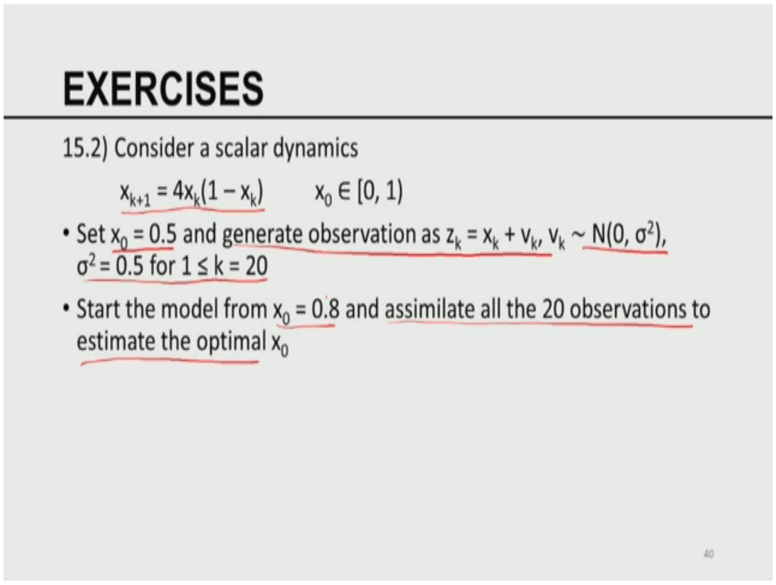
Once we have chosen a set of sequence of observations now I am going to consider different observation sets. O 1 I am going to consider assimilating all the observations. Another one I am going to consider assimilating only a subset of observation spaced four units apart. And then I am going to ask you to simulate even a further subset 1 10 20 30 40 and 50. So, what is that we are trying to do? We are trying to assimilate different subsets of observation smaller larger and the largest possible observation.

What is the idea? I would like you to be able to perform the data simulation experiment with different observation sets, and I would like you to be able to compare the quality of the recovered x naught initial condition. So in order to do that, once you have developed the observation now I would like to be able to start the forecast mode. To start the forecast mode I do not know what is up let us forget about the states that are used for observation. So I am going to assume, I am going to start the forecast model with 0.5, I am still going to use a is equal to 1. I want to use each set of observations and estimate the optimal x naught using the procedure we talked about. I would like you to in particular use the gradient method.

Being the linear case the J function is quadratic. If the J function is quadratic gradient method should work reasonably well. So, I would like you to discuss the impact of the varying the number of observation and the quality of estimate. This I would like you to do with one variance, then you could once you program this you can then change sigma square we can make sigma square less sigma square more. So, you can discuss a variety of cases.

The impact of the estimate on the observation noise, you can also discuss the impact of estimate and the number of observations. So, one can create quite a variety of computer related projects. So, this is a very good example that can be used to test. Your understanding of the basic mechanics of the 4-D war this problem can be solved using Lagrange multiplier technique. So, I would like to encourage you to be able to apply the graduate multiplier technique to solve this linear problem I believe it will be a very educative exercise to do this pencil paper and program and plot and discuss the results.

(Refer Slide Time: 52:30)



EXERCISES

15.2) Consider a scalar dynamics

$$x_{k+1} = 4x_k(1 - x_k) \qquad x_0 \in [0, 1)$$

- Set $x_0 = 0.5$ and generate observation as $z_k = x_k + v_k$, $v_k \sim N(0, \sigma^2)$, $\sigma^2 = 0.5$ for $1 \leq k = 20$
- Start the model from $x_0 = 0.8$ and assimilate all the 20 observations to estimate the optimal x_0

40

Then I am going to give you a nonlinear dynamics which is a again a very interesting dynamics, this is called the logistic equation. When the parameter is 4, this logistic equation is supposed to exhibit extreme sensitivity with the initial condition. The model actually exhibits chaotic behavior for any initial condition in this range. So, the previous problem is an easy problem this is a slightly more difficult problem, but still I would like you to make your hands dirty by doing this.

Start with the x naught equal to 0.5, generate a set of observations. Again I have given you observational covariance use sigma square is equal to 0.5, generate 20 of observations. Then start the model from a wrong initial condition. Then assimilate all the 20 observations into the esteem and find the optimal estimate. Now look at that the observations were generated using 0.5 the forecast is used started from 0.8. So, forecast will have the error you are going to use the 4-D war method to be able to correct the forecast error in the nonlinear model.

So, I would like you to utilize the adjoint method that we have described in solving this problem. I hope you will spend enough time to be able to solve these two problems. If you solve these two problem by applying this method first pencil paper and then program it you can say you understand 4-D war methods (Refer Time: 53:58).

With that we conclude our coverage of an introduction to solving inverse problems within the context of deterministic dynamic models with noisy observations.

Thank you.