Quantifying Uncertainty

Sai Ravela

M. I. T

2012
Uncertainty Propagates in Time-Dependent Processes

\[ M \equiv M(x_n; \alpha_n) \]

\[ x_{n+1} = M(x_n; \alpha_n) + \omega_n \]

M:  -Physical or Statistical Model

\[ X_0 \xrightarrow[]{} X_1 \xrightarrow[]{} X_2 \xrightarrow[]{} \ldots \xrightarrow[]{} X_n \]
Uncertainty Propagates in Bayesian Networks

Found in Hierarchical Bayes, Graphical Models.
Uncertainty Propagates in Spatial Processes

Grid nodes

Interactions
Inference Problems


2. Recursive Bayesian Estimation for Sequential Filtering and Smoothing.

Inference Problems


2. Recursive Bayesian Estimation for Sequential Filtering and Smoothing.


Propagating Uncertainty, a first step.
Variational Inference

\[ J(x_0) := \frac{1}{2} (x_0 - x_b)^T C_{00}^{-1} (x_0 - x_b) + \sum_{i=1}^{m} \left\{ \frac{1}{2} (y_i - Hx_i)^T R^{-1} (y_i - Hx_i) + \lambda_i^T [x_i - M(x_{i-1}; \alpha)] \right\} \]

Cannot deal with stochastic model (i.e. model error). Needs a Bayesian formalism.
Filters and Smoothers

Sequential Filtering:

\[ P(x_n | y_1 \cdots y_n) \propto P(y_n | x_n) P(x_n | x_{n-1}) P(x_{n-1} | y_1 \cdots y_{n-1}) \]  

\[ = P(y_n | x_n) P(x_n | y_1 \cdots y_{n-1}) \]  

\[ = P(y_n | x_n) P(x_n^f) \]  

The recursive form is simple when a perfect model is assumed, but the Kolmogorov-Chapman equation has to be used in the presence of model error. \( P(x_n | y_1 \cdots y_{n-1}) \) is the forecast distribution or prior distribution also seen as \( P(x_n^f) \)
Write the Objective

Sequential Filtering:

\[
J(x_n) := \frac{1}{2} (x_n - x_n^f)^T P_f^{-1} (x_n - x_n^f) + \frac{1}{2} (y_n - Hx_n)^T R^{-1} (y_n - Hx_n)
\]  

(4)

We have assumed a linear observation operator \( y_n = Hx_n + \eta \), with \( \eta \sim N(0, R) \).
Sequential Filtering:

\[
\hat{x}_n = x_n^f + P_f H^T (H P_f H^T + R)^{-1} (y_n - H x_n^f) \quad (5)
\]

\[
= x^n_a \quad (6)
\]

\[
P_a = (H^T R^{-1} H + P_f^{-1})^{-1} \quad (7)
\]

\[
= P_f - P_f H^T (H P_f H^T + R)^{-1} HP_f \quad (8)
\]

Then, launch a new prediction \( x_{n+1}^f = M(x_n) \) and the new uncertainty (predicted) is \( P_f = L P_a L^T \), where \( L = \frac{\partial M}{\partial x_n} \) when the model is nonlinear. Propagating produces the moments of \( P(x_{n+1} \mid y_1 \ldots y_n) \).
We are interested in the state estimates at all points in an interval, that is:

\[ P(x_1 \ldots x_n | y_1 \ldots y_n) \]  

(9)

The joint distribution can account for model errors, state and parameter errors within its framework. We break it down via Bayes Rule, Conditional Independence and Markov assumption, and marginalization and perfect model assumption, leading to a coupled set of equations that are recursively solved.
Uncertainty Propagation is Expensive

Forward (you’ll need this in the end)

\[
C_{ii} = \frac{\partial M}{\partial x_{i-1}} C_{i-1i-1} \frac{\partial M^T}{\partial x_{i-1}} \quad 0 < i \leq m
\]

Backward via information form:

\[
\hat{I}_{mm} = H^T R^{-1} H
\]

\[
\hat{I}_{ii} = \frac{\partial M^T}{\partial x_i} I_{i+1i+1} \frac{\partial M}{\partial x_i} + H^T R^{-1} H
\]

\[
\hat{C}_{00} = \left[ C_{00}^{-1} + \frac{\partial M^T}{\partial x_0} \hat{I}_{11} \frac{\partial M}{\partial x_0} \right]^{-1}
\]
The Dimensionality and Nonlinearity Challenges

Monte-Carlo
- Reduced-rank approximation
- Particle Filter

Domain Decomposition
- Localization, Localized Filters
- Scale-recursive Spatial Inference

Model Reduction & Interpolation
- Snapshots & POD
- Krylov Subspace

Response Surface Models
- Deterministic Equivalent Modeling Method
- Stochastic Response Surface Methodology

Polynomial Chaos Expansions
- Generalized Polynomial Chaos
Monte-Carlo

Isotropic Initial Perturbation $\rightarrow$ Leading Lyapunov vectors $\rightarrow$ time

Quantifying Uncertainty
Filter-Updating

\[ A^f = [x_1^f \ldots x_s^f] \Rightarrow \text{All at time } T \]

\[ \tilde{A}^f = [\tilde{x}_1^f \ldots \tilde{x}_s^f] \]

NOTE THAT

\[ P^f = \frac{1}{s-1} \tilde{A}^f \tilde{A}^{fT} \iff \text{Uncertainty} \]

So, propagate uncertainty through Samples “Integrated” forward. Model is not linearized.
No Linearization

\[ y = h(x) + \eta, \quad \eta \sim N(0, R) \]
\[ Z = [y + \eta_1, \ldots, y + \eta_s] \leftarrow \text{Perturbed Observations} \]
\[ R \approx \frac{1}{s-1} \tilde{Z} \cdot \tilde{Z}^T \]

Also, let
\[ \Omega^f = h(A^f) = [h(x_1^f) \ldots h(x_s^f)] \]
\[ \tilde{\Omega}^f \text{ defined similarly} \]
Uncorrelated Noise

Note

\[(\tilde{\Omega}^f + \tilde{Z})(\tilde{\Omega}^{fT} + \tilde{Z}^T) = (\tilde{\Omega}^f \tilde{\Omega}^{fT} + \tilde{Z}\tilde{Z}^T)\]

When observation noise is uncorrelated with state \(\equiv\) an assumption

Let

\(x^a\) be the estimate, analysis, ‘posterior’ rv.
\(A^a\) and \(\tilde{A}^a\) similarly, defined.
Easy Formulation

\[ A^a = A^f + \tilde{A}^f \tilde{\Omega}^{fT} \left[ \tilde{\Omega}^f \tilde{\Omega}^{fT} + \tilde{Z} \tilde{Z}^T \right]^{-1} \left[ Z - \Omega^f \right] \]

Identical to KF/EKF in linear/linearized case
⇒ No linearization of the model
⇒ No explicit uncertainty (covariance) propagation

\[ \left[ \tilde{\Omega}^f \tilde{\Omega}^{fT} + \tilde{Z} \tilde{Z}^T \right]^{-1} = \left( [\tilde{\Omega}^f + \tilde{Z}][\tilde{\Omega}^{fT} + \tilde{Z}^T] \right)^{-1} \]

\[ = (CC^T)^{-1} \]
Let

\[ C = \begin{bmatrix} U & S & V^T \end{bmatrix} \]

\[ [CC^T]^{-1} = US^{-2}U^T \]

\[ = (US^{-1})(US^{-1})^T \]

\[ = \sqrt{D}\sqrt{D}^T \]

\[ = D \]
Fast Calculation

\[ A^a = A^f + \tilde{A}^f \tilde{\Omega}^f T [US^{-2} U^T][Z - \Omega^f] \]

\[(n, s) (n, s) (n, s)(n, s)(n, s)(s, n)(n, s)(s, n)(n, s) (n, s)\]

Return by right to left, multiply; FAST, low-dimensional

\[ A^a = A^f + \tilde{A}^f X_5 \]
\[ = A^f (I_s + X_4) \]
\[ = A^f X_5 \]

A “weakly” nonlinear transformation \((X_5 \equiv X_5(A^f))\)
Time Dependent Example

Lorentz

\[ \dot{x} = -x_{i-2}x_{i-1} + x_{i-1}x_{i+1} - x_i + u \]

\[ = x_{i-1} [x_{i+1} - x_{i-2}] - x_i + u \]

- Advective
- Dissipative
- Forcing

Chaotic

Filter
Need to Get multimedia WORKING

Play ENKFLP.wmv!
Chalk Talk: Method 2.
Demo: Matlab.
Demo: PI Bottle.
Plug and Play

So,

\[ A_0^a = A_0^f I_s \leftarrow \text{No measurement} \]
\[ A_1^a = A_1^f X_5 \leftarrow \text{Filter, same as } X_5 \]
\[ A_1^s = A_1^a I_s \leftarrow \text{No future measurement} \]
\[ A_0^s = A_0^a + \tilde{A}_0^a \tilde{\Omega}_1^T [U_1 S_1^{-2} U_1^T] [Z_1 - \Omega_1^f] \]
\[ = A_0^a X_5 \]

Note: \( X_5 \) here is same as \( X_5 \) in earlier slide.
On the graph

Message sent from \(\hat{x}_1\) to \(\hat{x}_0(X5_1)\)
\(\hat{x}_0\) smoothed by \(y_1\) i.e \(A^s_0 \sim Pr(\hat{x}_0|y_1)\)
Fixed Interval & Fixed Lag

Fixed Interval

\[ Y_1 \quad Y_{n-1} \quad Y_n \]

\[ x_0 \quad x_1 \quad x_2 \quad x_3 \quad x_{n-1} \quad x_n \]

\[
\begin{align*}
P(x_0|y_1 \cdots y_n) \\
P(x_1|y_1 \cdots Y_n) \\
\vdots \\
P(x_n|y_1 \cdots y_n)
\end{align*}
\]

Smother

Filter
Fixed Interval & Fixed Lag

Fixed Lag

\[ Y_1, Y_{n-1}, Y_n \]

\[ Y_1, Y_{n-1}, Y_n \]

\[ P(x_0|y_1 \ldots y_n) \approx P(x_0|y_1 \ldots y_L), \quad (L < n) \]

\[ P(x_i|Y_0 \ldots Y_{i+L}) \]

Smoother up to a “window”
Fixed Interval: The Dumb Way

Graphical Model of Interval Smoothing

Observations

Filtering

Integration

Smoothing

Time

$A_0^g$ $A_5^g$ $A_{10}^g$ $A_{15}^g$ $A_{20}^g$ $A_{24}^f$ $A_{24}^a$ $Y_4$ $X_{54}$ $X_{512}$ $X_{524}$ $Y_{24}$
Backward Recursion

Key Assumption: Jointly Gaussian Distributions.

\[ A^s_k = A^a_k \prod_{j=k+1}^{N} X5_j \]

\[ C_k = \prod_{j=k+1}^{N} X5_j = X5_{k+1} C_{k+1} \]
Fixed Interval: The New Normal
Fixed Interval on Lorenz

The state is observed at every other location so giving $M$ giving $T$ and window is 120 h. Figure 1 compares the estimation error of 1.2 h, implying that our measurement interval therefore.

Smoothed analyses are produced at every model time step, thence marking the true initial condition at the beginning of the smoothing interval (transients). The system is completely observed at every time step so

$Lorenz-95$ system (Lorenz and Emanuel 1998). The continuous time Lorenz equations are:

$$\frac{dx_i}{dt} = \frac{1}{10} \left[ C_0 x_{i+1} + C_0 x_{i-1} - 2x_i + u \right]$$

for $i = 1, ..., 128$. The observation step is 0.05, giving $\tau = 5$. The constant forcing term is $u = 5$. Then an ensemble of 100 random initial condition observations over the interval, giving identical estimates although there can be numerical differences.

Fixed interval smoothing: FBF is faster than V1 while longer fixed lags give results closer to the fixed interval smoother (compare to Fig. 1). Short fixed lags give results closer to the ensemble Kalman filter, but considers the two fixed-lag smoothers. Figure 2 compares the root-mean-squared errors obtained from ensemble smoothing algorithms. The error in each state is computed between the analysis ensemble mean and the known true values. V1 and FBF algorithms are then:

- $V1 = R_{x_1}^{1/2}$
- $FBF$ = $R_{x_1}^{1/2}$

The incremental costs of the four numerical inversion in the FIFO-lag implementation. Variation could be expected as V1 computation time grows quadratically with the number of observations, and synthetic observations are

EnKF (Evensen 2004) with those obtained from the V1 and FBF algorithms. The unit costs defined earlier are then:

- $E_{x_1} = \frac{1}{2} R_{x_1}$
- $E_{x_2} = \frac{1}{2} R_{x_2}$
- $E_{x_3} = \frac{1}{2} R_{x_3}$

Our second experiment uses the same inputs as the first experiment, while longer fixed lags give results closer to the fixed interval smoother.

Comparison of V1-lag and FIFO-lag fixed-lag ensemble smoothing estimates for different lag window lengths. The other parameters are identical to those used in Fig. 1. V1-lag and FIFO-lag give identical estimates although there can be numerical differences.

The ratio of ensemble to state size is typically very small so FBF becomes more expected as V1 computation time grows quadratically with the number of observations, and synthetic observations are.
Costs of Inference, Toy Problem

Additional computational time (over model propagation and filtering) for V1 smoothing.

FBF smoothing is minor. See text for detailed definition of each.

The additional cost of V1 smoothing can be much more than filtering alone, while the additional cost of FBF smoothing is nearly independent of lag (small fluctuations are still observed time, i.e. \( \Delta C \)).

As one of the motivations for fixed-lag smoothing as an alternative to FBF suggestions that V1-lag is more expensive by a fixed factor. FIFO-lag computational benefits of the proposed algorithms (FBF, V1-lag and FIFO-lag) are even better. In the case where the number of experimental savings, therefore a comparison between the two is instructive. It is clear that V1-lag is less expensive than V1-lag and FIFO-lag suggests that V1-lag is more expensive than FIFO-lag by a fixed factor. FIFO-lag computational time growing with interval length at approximately the same rate as filtering alone.

In the Lorenz experiments, where the fixed-interval length, while the FBF computation time grows linearly. Figure 3 indicates for the Lorenz-95 example that V1 smoothing takes much more time than filtering, becoming prohibitively expensive for large lag and V1-lag suggests that V1-lag is more expensive than FIFO-lag. In our conservative approximation to fixed-interval smoothing is computationally time. In the Lorenz experiments, where the fixed-interval length, while the FBF computation time grows linearly. Figure 3 indicates for the Lorenz-95 example that V1 smoothing takes much more time than filtering, becoming prohibitively expensive for large lag and V1-lag suggests that V1-lag is more expensive than FIFO-lag. In our conservative approximation to fixed-interval smoothing is computational benefits of the proposed algorithms (FBF, V1-lag and FIFO-lag) are even better. In the case where the number of experimental savings, therefore a comparison between the two is instructive. It is clear that V1-lag is less expensive than V1-lag and FIFO-lag suggests that V1-lag is more expensive than FIFO-lag by a fixed factor. FIFO-lag computational time growing with interval length at approximately the same rate as filtering alone.

In the Lorenz experiments, where the fixed-interval length, while the FBF computation time grows linearly. Figure 3 indicates for the Lorenz-95 example that V1 smoothing takes much more time than filtering, becoming prohibitively expensive for large lag and V1-lag suggests that V1-lag is more expensive than FIFO-lag. In our conservative approximation to fixed-interval smoothing is computationally time. In the Lorenz experiments, where the fixed-interval length, while the FBF computation time grows linearly. Figure 3 indicates for the Lorenz-95 example that V1 smoothing takes much more time than filtering, becoming prohibitively expensive for large lag and V1-lag suggests that V1-lag is more expensive than FIFO-lag. In our conservative approximation to fixed-interval smoothing is computational benefits of the proposed algorithms (FBF, V1-lag and FIFO-lag) are even better. In the case where the number of experimental savings, therefore a comparison between the two is instructive. It is clear that V1-lag is less expensive than V1-lag and FIFO-lag suggests that V1-lag is more expensive than FIFO-lag by a fixed factor. FIFO-lag computational time growing with interval length at approximately the same rate as filtering alone.
Fixed Lag

Fixed Lag Smoother

V1-lag & FIFO-lag

Lag = 1

Lag = 9

Lag = 5

Lag = 13

Error

Time

0 50 100

0 0.2 0.4

0 0.2 0.4
Fixed Lag: The Dumb Way

![Graph showing computational times for different lags]
Fixed Lag is FIFO

\[
A_k^s = A_k^a \prod_{j=k+1}^{k+w} X5_j \\
= A_k^a C_k \\
C_k = X5_k^{-1} C_{k-1} X5_{k+w}
\]
Fixed Lag: The New Normal
We need to fix the multimedia!

Watch FLKSO.wmv!
Reading: Ravela and McLaughlin, Fast Ensemble Smoothing, Ocean Dynamics, 2007
Schneider 2001: Analysis of incomplete climate data: Estimation of mean values and covariance matrices and imputation of missing values, Journal of Climate
Where does ensemble come from?

Singular vectors

Low dimensional Subspace
Span \{u^{(0)} \ldots u^{(N)}\}

\[ u_0^{(0)} \rightarrow L_0 \approx \left. \frac{\partial M}{\partial x} \right|_{x=x_0} \rightarrow u_1^{(0)} \]
Things get tough... the Tough linearize

Thus

\[ x_1 = M(x_0) \]
\[ = M(\bar{x}_0 + \tilde{x}_0) \]
\[ = M(\bar{x}_0) + \frac{\partial M}{\partial x} \bigg|_{x=\bar{x}_0} \tilde{x}_0 \]

\[ \tilde{x}_1 = \mathcal{L} \tilde{x}_0 \]

\[ u_{1}^{(k)} = \mathcal{L} u_{0}^{(k)} \]
Now, let $C_1$ be a metric on vector $u_1$ and let $C_0$ be a metric on $u_0$.

$$\lambda = \frac{\langle \mathcal{L}u_0, C_1 \mathcal{L}u_0 \rangle}{\langle u_0, C_0 u_0 \rangle} = \frac{\langle u_0, \mathcal{L}^\# C_1 \mathcal{L}u_0 \rangle}{\langle u_0, C_0 u_0 \rangle}$$

Maximize ratio for the $k^{th}$ perturbation: $\lambda_k$:

$$\Rightarrow \mathcal{L}^\# C_1 \mathcal{L}u_0^{(k)} = \lambda_k C_0 u_0^{(k)}$$

Which is a generalized eigenvalue problem. Note that when $C_1 = I$, and $C_0 = P_0^f$ then $u_1^{(k)}$ are leading directions of $P_1^f$. 
SV approach

Notes
- Adjoint & TLM not easy to calculate but robust.
- $\mathcal{L}$ may be really large too! How can we reduce $\mathcal{L}$?
- Sensitivity to norm.
Breeding

Initial Perturbation

Align with leading directions of error growth (Lyapunov vectors)

\[ Q_{i+1} R_{i+1} = \mathcal{L}_{TLM} Q_i \]

- \( Q_0 \equiv I \)
- \( Q_0 \rightarrow Q_i \cdots Q_k \)
- \( Q_k \) forgets \( Q_0 \)
It’s easy to breed

1. Generate “random” initial perturbation
2. Let it grow; renormalize. (i.e propagate it)
3. Repeat

⇒ Breeding vectors
How many bred vectors?
⇒ Size of L?
Ways to simplify Models for Uncertainty Propagation

1. Spectral Truncation: Find a few leading directions of Covariance or Model and propagate them. Breed Vectors. Calculate a reduced local linear model from ensemble.

2. Localization: Localize filtering and smoothing, use scale-recursive decomposition.

3. Model Reduction: Reduce order of linearized model, construct a reduced model from snapshots.

4. Sample Input-Output pairs to create a simple auxiliary model.
Model Reduction

Model Reduction Structure

SVD
Nonlinear
POD Empirical Grammian

Linear
Balanced Truncation Hankel approx.

Krylov
Lanczos Arnoldi etc.

SVD Krylov

44
Model Bypass – Non-Intrusive Approaches

\[ X_0 \xrightarrow{M} X_1 \]

Response Surface Modeling, Polynomial Chaos.

\[ C_{10} C_{00}^{-1} \]
Extra-Special on Covariance Representations

If we have a large covariance matrix $C$ nonetheless representable by computer and if we know it is a block-circulant matrix, then the Fourier Transform can be used to diagonalize it:

$$D = UCU^T$$

(10)

For the unitary transform $U$, and $D$ is diagonal. So, subsequent processing with covariance is simplified, provided the model and state can be also expressed in fourier domain.

$$U\delta x_{n+1} = U \frac{\partial M}{\partial x} U^T Ux_n$$

(11)

$$\delta \xi_{n+1} = \mathcal{L}_\mathcal{F} \delta \xi_n$$

(12)

$$\delta x_n^T C_n^{-1} \delta x_n = \delta \xi^T D_n^{-1} \delta \xi$$

(13)

Spectral truncation to a few wave numbers in $U$ also leads to a reduced order model. Incidentally, similar process for wavelet decomposition.

DO MATLAB EXAMPLE
Iterative calculation

If a covariance $C$ has eigen vectors $U$ and eigenvalues $\lambda$, i.e. $CU = U\Lambda$, then we may recursively calculate the leading modes in $U$ because:

$$C = \sum_{k=1}^{N} u_k \lambda_{kk} u_k^T$$  \hspace{1cm} (14)

Where $U = [u_1 \ldots u_N]$ and $\Lambda = \text{diag}(\lambda_{11}, \ldots, \lambda_{NN})$, in decreasing order. Let $C_{11} = C$, and iteratively calculate:

$$\text{for } k = 1 \ldots N$$  \hspace{1cm} (15)

$$\{u_k, \lambda_{kk}\} = \text{LeadingEig}(C_{kk})$$  \hspace{1cm} (16)

$$C_{k+1k+1} = C_{kk} - u_k \lambda_{kk} u_k^T$$  \hspace{1cm} (17)

We need a procedure to calculate the leading Eigenvector and Eigenvalue.
Basic approach: Power Iteration

WARNING: There are many advanced methods for calculating eigen vectors and eigen values iteratively and one should use them (e.g. from ARPACK). Here, we provide an intuition for the process. To calculate the leading vector of $C$, let us consider a vector in the basis $z$, which we may expand as:

$$z = \sum_{k=1}^{N} c_k u_k$$  \hspace{1cm} (19)

Now, we can write for the $n^{th}$ power of $C$:

$$C^n z = \sum_{k=1}^{N} c_k C^n u_k$$  \hspace{1cm} (20)

$$= \sum_{k=1}^{N} c_k \lambda_{kk}^n u_k$$  \hspace{1cm} (21)

HOW?
Power Iteration Continued

\[ C^n z = c_1 \lambda_{11}^n (u_1 + \sum_{k=2}^{N} \frac{c_k}{c_1} \frac{\lambda_{kk}^n}{\lambda_{11}^n} u_k) \]  

Defining \( z_n = C^n z \), we note that

\[ n \to \infty \Rightarrow \frac{z_n}{||z_n||} \to u_1 \]  

Algorithm \textit{PowerIteration}(C):

1. Initialize \( z \); \( z \leftarrow \frac{z}{||z||} \)
2. Iterate: \( t \leftarrow Cz \), \( z \leftarrow \frac{t}{||t||} \)
LeadingEig(C)

\[ u = \text{PowerIteration}(C) \quad (24) \]
\[ \lambda = u^T Cu \quad (25) \]
\[ \text{return}(u, \lambda) \quad \text{end} \quad (26) \]
But the Covariance is too LARGE!

- The preceding discussion is all fine, but often the dimensionality is such that we have a really large covariance that cannot be represented. Fortunately, many physical problems have only a few modes of interest which we represent through data, e.g. an ensemble.

- So we begin with a skinny matrix $X$, and assume the covariance is $C = XX^T$. We would like a representation without explicitly calculating $C$ and exploiting the rank-deficiency due to a skinny $X$. 
Alternate form

- Let $X = USV^T$ be the singular value decomposition and here $S_{ii} \geq S_{i+1i+1}$. Then $C = U\Lambda U^T$ where $\Lambda = S^2$.
- We will calculate only a few top left and right singular vectors and singular values \textit{iteratively} for a reduced order representation $C_d = U_d\Lambda_d U_d^T$.
- Note that because $X$ is skinny, i.e. it is of size $n \times N$ with $N << n$. We may further only pick $d$ modes, $d \leq N$.
- We would like a representation of $C_d$ without explicitly calculating it.
- Notice that $D = X^TX$ is a small matrix when $X$ is skinny.
Alternate form

Let $X = USV^T$ be the singular value decomposition and here $S_{ii} \geq S_{i+1,i+1}$. Then $D = X^T X = V \Lambda V^T$ where $\Lambda = S^2$, a small matrix.

We calculate the eigen vectors and eigen values of $D$ recursively. Let $D_1 = D$; and for $k = 1 \ldots d$

\begin{align*}
    v_k &= \text{PowerIteration}(D_k) \quad (27) \\
    \lambda_{kk} &= v_k^T D_k v_k \quad (28) \\
    D_{k+1} &= D_k - v_k \lambda_{kk} v_k^T \quad (29)
\end{align*}

Noting that $S_d = \sqrt{\Lambda_d}$, we obtain $U_d$ as a skinny $n \times d$ matrix:

$$U_d = XV_d S_d^{-1} \quad (30)$$

Store $U_d$ and $\Lambda_d$ and use them to calculate the norm in an application. DEMO IN MATLAB
Applicable to Processes

\[ \frac{\partial \theta}{\partial t}(x, t) = F\theta(x, t) \rightarrow \text{System} \]

\[ R(\theta) = \frac{\partial \theta}{\partial t} - F\theta \rightarrow \text{Residual} \]

\[ \theta = u\eta(t) \rightarrow \text{KLT (POD or Krylov)} \]

\[ u^T R = 0 \rightarrow \text{Galerkin Projection} \]

\[ \frac{\partial \eta}{\partial t} = u^T F u \eta \rightarrow \text{ROM} \]