Data Assimilation as Parallel Minimization

Antti Solonen, Idrissa S. Amour, Harri Auvinen, John Bardsley, Heikki Haario and Tuomo Kauranne

Lappeenranta University of Technology

14th ECMWF Workshop on the Use of High Performance Computing in Meteorology
November 4, 2010

Overview

3D Variational Assimilation (3D-Var)

Algorithm

Minimize

$$
\begin{aligned}
J(\mathbf{x}(t_i)) &= J_b + J_o \\
&= \frac{1}{2}(\mathbf{x}(t_i) - \mathbf{x}^b(t_i))^{\mathrm{T}}\mathbf{B}_0^{-1}(\mathbf{x}(t_i) - \mathbf{x}^b(t_i)) \\
&+ \frac{1}{2}(H(\mathbf{x}(t_i)) - \mathbf{y}_i^o)^{\mathrm{T}}\mathbf{R}^{-1}(H(\mathbf{x}(t_i)) - \mathbf{y}_i^o),
\end{aligned}
$$

3D Variational Assimilation (3D-Var)

Where

- $\mathbf{x}(t_i)$ *is the analysis at time* t_i
- $\mathbf{x^b}(t_i)$ *is the background at time* t_i
- $\mathbf{y}_i^o$ *is the vector of observations at time* t_i
- $\mathbf{B}_0$ *is the background error covariance matrix*
- $\mathbf{R}$ *is the observation error covariance matrix*
- H *is the nonlinear observation operator*

3D Variational Assimilation (3D-Var)

Properties

- *3D-Var is computed at a snapshot in time where all observations are assumed contemporaneous*
- *3D-Var does not take into account atmospheric dynamics, by which*
- *It does not depend on the weather model*

4D Variational Assimilation (4D-Var)

Algorithm

Minimize

$$J(\mathbf{x}(t_0)) = J_b + J_o$$

$$= \frac{1}{2}(\mathbf{x}(t_0) - \mathbf{x}^b(t_0))^{\mathrm{T}}\mathbf{B}_0^{-1}(\mathbf{x}(t_0) - \mathbf{x}^b(t_0))$$

$$+ \frac{1}{2}\sum_{i=0}^{n}(H(M(t_i,t_0)(\mathbf{x}(t_0))) - \mathbf{y}_i^o)^{\mathrm{T}}\mathbf{R}^{-1}(H(M(t_i,t_0)(\mathbf{x}(t_0))) - \mathbf{y}_i^o)$$

4D Variational Assimilation (4D-Var)

Where

- $\mathbf{x}(t_0)$ *is the analysis at the beginning of the assimilation window*
- $\mathbf{x}^\mathbf{b}(t_0)$ *is the background at the beginning of the assimilation window*
- $\mathbf{B}_0$ *is the background error covariance matrix*
- $\mathbf{R}$ *is the observation error covariance matrix*
- H *is the nonlinear observation operator*
- M *is the nonlinear weather model*

4D Variational Assimilation (4D-Var)

Properties

- *The model is assumed to be perfect*
- *Model integrations are carried out forward in time with the nonlinear model and the tangent linear model, and backward in time with the corresponding adjoint model*
- *Minimization is sequential*
- *The weather model can run in parallel*

The Extended Kalman Filter (EKF)

Algorithm

Iterate in time

$$\mathbf{x}^f(t_i) = M(t_i, t_{i-1})(\mathbf{x}^a(t_{i-1}))$$

$$\mathbf{P}_i^f = \mathbf{M}_i \mathbf{P}^a(t_{i-1})\mathbf{M}_i^{\mathrm{T}} + \mathbf{Q}$$

$$\mathbf{K}_i = \mathbf{P}^f(t_i)\mathbf{H}_i^{\mathrm{T}}(\mathbf{H}_i\mathbf{P}^f(t_i)\mathbf{H}_i^{\mathrm{T}} + \mathbf{R})^{-1}$$

$$\mathbf{x}^a(t_i) = \mathbf{x}^f(t_i) + \mathbf{K}_i(\mathbf{y}_i^o - H(\mathbf{x}^f(t_i)))$$

$$\mathbf{P}^a(t_i) = \mathbf{P}^f(t_i) - \mathbf{K}_i\mathbf{H}_i\mathbf{P}^f(t_i)$$

The Extended Kalman Filter (EKF)

Where

- $\mathbf{x}^f(t_i)$ *is the prediction at time* t_i
- $\mathbf{x}^a(t_i)$ *is the analysis at time* t_i
- $\mathbf{P}^f(t_i)$ *is the prediction error covariance matrix at time* t_i
- $\mathbf{P}^a(t_i)$ *is the analysis error covariance matrix at time* t_i
- $\mathbf{Q}$ *is the model error covariance matrix*
- $\mathbf{K}_i$ *is the Kalman gain matrix at time* t_i
- $\mathbf{R}$ *is the observation error covariance matrix*
- H *is the nonlinear observation operator*
- $\mathbf{H}_i$ *is the linearized observation operator at time* t_i
- $\mathbf{M}_i$ *is the linearized weather model at time* t_i

The Extended Kalman Filter (EKF)

Properties

- *The model is not assumed to be perfect*
- *Model integrations are carried out forward in time with the nonlinear model for the state estimate and*
- *Forward and backward in time with the tangent linear model and the adjoint model, respectively, for updating the prediction error covariance matrix*
- *There is no minimization, just matrix products and inversions*
- *Computational cost of EKF is prohibitive, because* $\mathbf{P}^f(t_i)$ *and* $\mathbf{P}^a(t_i)$ *are huge full matrices*

The Variational Kalman Filter (VKF)

Algorithm

Iterate in time

Step 0: *Select an initial guess* $\mathbf{x}^a(t_0)$ *and a covariance* $\mathbf{P}^a(t_0)$*, and set* $i = 1$.

Step 1: *Compute the evolution model state estimate and the prior covariance estimate:*
(i) Compute $\mathbf{x}^f(t_i) = M(t_i, t_{i-1})(\mathbf{x}^a(t_{i-1}))$*;*
(ii) **Minimize**

$$(\mathbf{P}^f(t_i))^{-1} = (\mathbf{M}_i \mathbf{P}^a(t_{i-1}) \mathbf{M}_i^{\mathrm{T}} + \mathbf{Q})^{-1}$$

by the LBFGS method - or CG, as in incremental 4DVar;

The Variational Kalman Filter (VKF)

Algorithm

Step 2: *Compute the Variational Kalman filter state estimate and the posterior covariance estimate:*

(i) **Minimize**

$$\lambda(\mathbf{x}^a(t_i)|\mathbf{y}_i^o)$$
$$=(\mathbf{y}_i^o - \mathbf{H}_i\mathbf{x}^a(t_i))^{\mathrm{T}}\mathbf{R}^{-1}(\mathbf{y}_i^o - \mathbf{H}_i\mathbf{x}^a(t_i))$$
$$+(\mathbf{x}^f(t_i) - \mathbf{x}^a(t_i))^{\mathrm{T}}(\mathbf{P}^f(t_i))^{-1}(\mathbf{x}^f(t_i) - \mathbf{x}^a(t_i))$$

by the LBFGS method - or CG, as in incremental 3DVar;

(ii) Store the result of the minimization as a VKF estimate $\mathbf{x}^a(t_i)$*;*

(iii) Store the limited memory approximation to $\mathbf{P}^a(t_i)$*;*

Step 3: *Update* $t := t + 1$ *and return to Step 1.*

The Variational Kalman Filter (VKF)

Where

- *Step 1(ii) is carried out with an auxiliary minimization that has a trivial solution but a random initial guess, and thereby generates a non-trivial minimization sequence*
- $\mathbf{P}^f(t_i)$ *and* $\mathbf{P}^a(t_i)$ *are kept in vector format, as a weighted sum of a diagonal or sparse background* $\mathbf{B}_0$, *a diagonal model error variance matrix* $\mathbf{Q}$ *and a low rank dynamical component* $\mathbf{P}^f(t_i)$ *that*
- *Is obtained from the Hessian update formula of the Limited Memory BFGS iteration*
- *The Kalman gain matrix is not needed*

The Variational Kalman Filter (VKF)

Properties

- *The model is not assumed to be perfect*
- *Model integrations are carried out forward in time with the nonlinear model for the state estimate and*
- *Forward and backward in time for updating the prediction error covariance matrix*
- *There are no matrix inversions, just matrix products and minimizations*
- *Computational cost of VKF is similar to 4D-Var*
- *Minimizations are sequantial*
- *Accuracy of analyses similar to EKF*

Overview

Ensemble Kalman Filters (EnKF)

Properties

- *Ensemble Kalman Filters are generally simpler to program than variational assimilation methods or EKF, because*
- *EnKF codes are based on just the non-linear model and do not require tangent linear or adjoint codes, but they*
- *Tend to suffer from slow convergence and therefore inaccurate analyses*
- *Often underestimate analysis error covariance*

Ensemble Kalman Filters (EnKF)

Properties

- *Ensemble Kalman filters often base analysis error covariance on* **bred vectors**, *i.e. the difference between ensemble members and the background, or the ensemble mean*
- *One family of EnKF methods is based on perturbed observations, while*
- *Another family uses explicit linear transforms to build up the ensemble*

EnKF Cost functions

Algorithm

Minimize

$$(\mathbf{P}^f(t_i))^{-1} = (\beta \mathbf{B}_0 + (1-\beta)\frac{1}{N}\mathbf{X}^f(t_i)\mathbf{X}^f(t_i)^{\mathrm{T}})^{-1}$$

Algorithm

Minimize

$$\begin{aligned}
&\ell(\mathbf{x}^a(t_i)|\mathbf{y}_i^o) \\
&= (\mathbf{y}_i^o - H(\mathbf{x}^a(t_i)))^{\mathrm{T}}\mathbf{R}^{-1}(\mathbf{y}_i^o - H(\mathbf{x}^a(t_i))) \\
&+ \frac{1}{N}\sum_{j=1}^{N}(\mathbf{x}_j^f(t_i) - \mathbf{x}^a(t_i))^{\mathrm{T}}(\mathbf{P}^f(t_i))^{-1}(\mathbf{x}_j^f(t_i) - \mathbf{x}^a(t_i))
\end{aligned}$$

Ensemble 4DVar Data Assimilation (EDA)

Algorithm

Step 0: *Select an initial guess* $\mathbf{x}^a(t_0)$ *and a covariance* $\mathbf{B}_0$ *and set* $i = 1$

Step 1: *Compute perturbed observations and physics:*
(i) Create an ensemble of observations by $\delta_k \mathbf{y}_i^o \sim N(\mathbf{y}_i^o, \mathbf{R}_i)$;

Step 2: *(i)* **Minimize**, *for each k*

$$J(\mathbf{x}(t_{i-1})) = J_b + J_o$$
$$= \frac{1}{2}(\mathbf{x}(t_0) - \mathbf{x}^b(t_0))^{\mathrm{T}} \mathbf{B}_0^{-1} (\mathbf{x}(t_0) - \mathbf{x}^b(t_0))$$
$$+ \frac{1}{2} \sum_{i=0}^{n} (H(M(t_i, t_0)(\mathbf{x}(t_0))) - (\mathbf{y}_j^o + \delta_k \mathbf{y}_j^o))^{\mathrm{T}} \mathbf{R}^{-1}$$
$$\times (H(M(t_i, t_0)(\mathbf{x}(t_0))) - (\mathbf{y}_j^o + \delta_k \mathbf{y}_j^o)),$$

to obtain an ensemble of analyses from $\mathbf{X}^f(t_{i-1})$ *to* $\mathbf{X}^f(t_i)$ *by the LBFGS method;*

Ensemble 4DVar Data Assimilation (EDA)

Algorithm

(ii) Compute the updated error covariance for each time step in the assimilation window
$(\mathbf{P}^a(t_j))^{-1} = (\mathbf{B}_0 + \mathbf{X}^f(t_j)\mathbf{X}^f(t_j)^{\mathrm{T}})^{-1}, \quad j = t_{i-1} \dots t_i$
by the LBFGS method;

Step 3: *(i)* **Minimize:**

$$J(\mathbf{x}(t_{i-1}), \dots, \mathbf{x}(t_i)) = J_b + J_o + J_M$$
$$= \frac{1}{2}\sum_{j=t_{i-1}}^{t_i} (\mathbf{x}(t_j) - \mathbf{x}^b(t_j))^{\mathrm{T}}(\mathbf{P}^a(t_j))^{-1}(\mathbf{x}(t_j) - \mathbf{x}^b(t_j))$$
$$+ \frac{1}{2}\sum_{i=0}^{n} (H(\mathbf{x}(t_j)) - \mathbf{y}_j^o)^{\mathrm{T}}\mathbf{R}^{-1}(H(\mathbf{x}(t_j)) - \mathbf{y}_j^o)$$
$$+ \frac{1}{2}\sum_{j=t_{i-1}}^{t_i} (\mathbf{x}(t_j) - M(t_j, t_{j-1})(\mathbf{x}(t_{j-1})))^{\mathrm{T}}\mathbf{Q}^{-1}$$
$$\times (\mathbf{x}(t_j) - M(t_j, t_{j-1})(\mathbf{x}(t_{j-1}))),$$

Store the result as the EDA estimate $\mathbf{x}^a(t_{i-1}), \dots, \mathbf{x}^a(t_i)$*;*
(ii) Update $i := i + 1$ *and return to Step 1.*

The Variational Ensemble Kalman Filter (VEnKF)

Algorithm

Iterate in time

Step 0: *Select a state* $\mathbf{x}^a(t_0)$ *and a covariance* $\mathbf{P}^a(t_0)$ *and set* $i = 1$

Step 1: *Evolve the state and the prior covariance estimate:*
(i) Compute $\mathbf{x}^f(t_i) = M(t_i, t_{i-1})(\mathbf{x}^a(t_{i-1}))$*;*
(ii) Compute the ensemble forecast $\mathbf{X}^f(t_i) = M(t_i, t_{i-1})(\mathbf{X}^a(t_{i-1}))$*;*
(iii) **Minimize** *from a random initial guess* $(\mathbf{P}^f(t_i))^{-1} = (\beta\mathbf{B}_0 + (1-\beta)\frac{1}{N}\mathbf{X}^f(t_i)\mathbf{X}^f(t_i)^{\mathrm{T}} + \mathbf{Q}_i)^{-1}$ *by the LBFGS method;*

The Variational Ensemble Kalman Filter (VEnKF)

Algorithm

Step 2: *Compute the Variational Ensemble Kalman Filter posterior state and covariance estimates:*

(i) **Minimize**

$$\ell(\mathbf{x}^a(t_i)|\mathbf{y}_i^o) = (\mathbf{y}_i^o - H(\mathbf{x}^a(t_i)))^{\mathrm{T}}\mathbf{R}^{-1}(\mathbf{y}_i^o - H(\mathbf{x}^a(t_i))) + (\mathbf{x}^f(t_i) - \mathbf{x}^a(t_i))^{\mathrm{T}}(\mathbf{P}^f(t_i))^{-1}(\mathbf{x}^f(t_i) - \mathbf{x}^a(t_i))$$

by the LBFGS method;

(ii) Store the result of the minimization as $\mathbf{x}^a(t_i)$*;*

(iii) Store the limited memory approximation to $\mathbf{P}^a(t_i)$*;*

(iv) Generate a new ensemble $\mathbf{X}^a(t_i) \sim N(\mathbf{x}^a(t_i), \mathbf{P}^a(t_i))$*;*

Step 3: *Update* $i := i + 1$ *and return to Step 1.*

The Variational Ensemble Kalman Filter (VEnKF)

Properties

- *Follows the algorithmic structure of VKF*
- *Bred vectors are centered on the* **mode***, not the mean, of the ensemble, as in Bayesian estimation*
- *Like in VKF, a new ensemble and a new error covariance matrix is generated at every observation time*
- *No covariance leakage*
- *No tangent linear or adjoint code*
- *Asymptotically equivalent to VKF and therefore EKF when ensemble size increases*

The modified Local Ensemble Transform Kalman Filter (mLETKF)

Properties

- *The goal of mLETKF is to produce an Ensemble Kalman filter that*
- *Will not require a tangent linear or adjoint code*
- *But will converge faster and thereby produce more accurate analyses than EnKF methods in general*
- *mLETKF is based on the* **4D-LETKF** *method by* **Hunt, Kostelic** *and* **Szunyogh**
- *It incorporates certain features from VKF, in particular*
- *It uses an analysis produced by a 3D-Var minimization with LBFGS as the vector to base bred vectors on*

The modified Local Ensemble Transform Kalman Filter (mLETKF)

Properties

The cost function to be minimized is a "dual 3D-Var" cost function that optimizes the weight of each ensemble member in the analysis, using the LBFGS method:

$$J(\mathbf{w}) = \beta(n-1)\mathbf{w}^{\mathrm{T}}\mathbf{w} + (1-\beta)\times$$

$$(\mathbf{y}^f - H(\mathbf{x}_k^a(t_i)) - \mathbf{w}^{\mathrm{T}}Y)^{\mathrm{T}}\mathbf{R}^{-1}(\mathbf{y}^f - H(\mathbf{x}_k^a(t_i)) - \mathbf{w}^{\mathrm{T}}Y)$$

The modified Local Ensemble Transform Kalman Filter (mLETKF)

Where

- $\mathbf{y}^f$ *is the synthetic observation vector of the prior* $\mathbf{y}^f = H(\mathbf{x}^f(t_i))$
- $\mathbf{w}$ *is the vector of the weights* w^k *of each ensemble member* $\mathbf{x}_k^a(t_i)$
- Y *is the matrix of synthetic observations of each ensemble member* $\mathbf{y}_k^f = H(\mathbf{x}_k^a(t_i))$
- N *is the ensemble size*
- β *is an empirical weight factor between 0 and 1*

The modified Local Ensemble Transform Kalman Filter (mLETKF)

Algorithm

Iterate in time

Step 0: *Initialize the background state* $\mathbf{x}^a(t_0)$ *and the ensemble members* $\mathbf{x}_k^a(t_0)$ *for* $k = 1, \ldots, N$

Step 1: *Compute* $\mathbf{x}_k^f(t_i) = M(\mathbf{x}_k^a(t_{i-1}))$ *and* $\mathbf{x}^f(t_i) = M(\mathbf{x}^a(t_{i-1}))$;

Step 2: *Perturb the members* $\mathbf{x}_k^f(t_i)$ *and assemble them in matrix* Ψ;

Step 3: *Compute the matrix* $X^f(t_i) : \mathbf{x}_k^f(t_i) = \mathbf{x}^f(t_i) - \Psi^k$;

Step 4: *Compute the matrix* $Y^f(t_i) : \mathbf{y}_k^f(t_i) = H(\mathbf{x}_k^f(t_i)) - H(\mathbf{x}^f(t_i))$;

The modified Local Ensemble Transform Kalman Filter (mLETKF)

Algorithm

Step 5: *Minimize the dual 3D-Var cost function* $J(\mathbf{w})$ *using the LBFGS method.*

Step 6: *Compute the analysis* $\mathbf{x}^a(t_i) = \mathbf{x}^f(t_i) + \mathbf{w}^{\mathrm{T}} X^f(t_i)$

Step 7: *Compute the background ensemble* $X^a(t_i) : \mathbf{x}_k^a(t_i) = \mathbf{x}_k^f(t_i) + \mathbf{w}^{\mathrm{T}} X^f(t_i)$

Step 8: *Update* $i := i + 1$ *and return to Step 1.*

Overview

The Lorenz '95 Model

Properties

- *The Lorenz '95 model is computationally light and represents an analogue of mid-latitude atmospheric dynamics.*
- *The variables of the model can be thought of as representing some atmospheric quantity on a single latitude circle.*
- *The model consists of a system of coupled ordinary differential equations*

$$\frac{\partial c_i}{\partial t} = c_{i-2}c_{i-1} + c_{i-1}c_{i+1} - c_i + F,$$

- *Grid points range between* $i = 1, 2, ..., k$ *and* F *is a*

The Lorenz '95 Model

Where

- *The domain is set to be cyclic, so that* $c_{-1} = c_{k-1}, c_0 = c_k$ *and* $c_{k+1} = c_1$.
- *The parameter values used in the simulation of the system were selected as follows:*
- *the number of grid points* $k = 40$,
- *the climatological standard deviation of the model state,* $\sigma_{clim} \approx 3.64$,
- *the observation noise matrix* $\mathbf{R} = 0.15\sigma_{clim}\mathbf{I}$ *and*
- *prediction error covariance* $\mathbf{B_0} = 0.5\sigma_{clim}\mathbf{I}$.

The Lorenz '95 Model

Properties

- *The system was assimilated using each of EKF, VKF and VEnKF.*
- *In order to compare the quality of analyses produced by all three methods, we compute the following forecast statistics at every 8th observation.*
- *Take* $j \in \mathcal{I} := \{8i \mid i = 1, 2, \ldots, 100\}$ *and define*

$$[\textbf{forcast\_error}_j]_i = \frac{1}{40} \| M_{4i}(\mathbf{x}_j^{est}) - \mathbf{x}_{j+4i}^{true} \|^2, \quad i = 1, \ldots, 20,$$

The Lorenz '95 Model

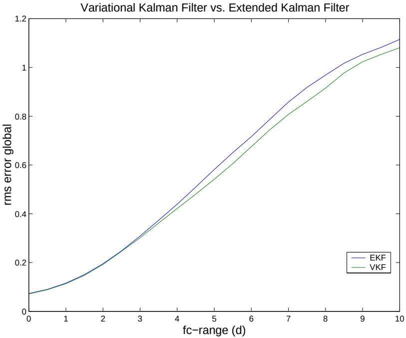
Where

- *M_n denotes a forward integration of the model by n time steps with the RK4 method.*
- *This vector gives a measure of forecast accuracy given by the respective filter estimate up to 80 time steps, or 10 days out.*
- *This allows us to define the forecast skill vector*

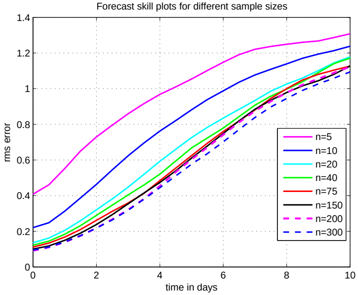
$$[\textbf{forecast\_skill}]_i = \frac{1}{\sigma_{\text{clim}}} \sqrt{\frac{1}{100} \sum_{j \in \mathcal{I}} [\textbf{forecast\_error}_j]_i},$$

i=1,. . . ,20,

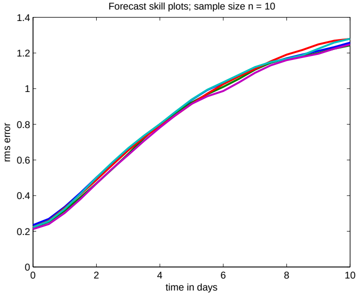
VKF vs. EKF


Variational Kalman Filter vs. Extended Kalman Filter
rms error global
fc−range (d)
EKF
VKF
0
0.2
0.4
0.6
0.8
1
1.2
0
1
2
3
4
5
6
7
8
9
10

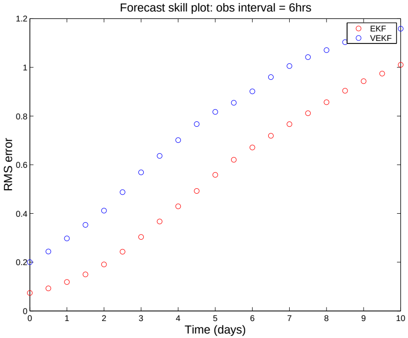
VEnKF, N = 5, ... , 300


Forecast skill plots for different sample sizes
rms error
time in days
0
0.2
0.4
0.6
0.8
1
1.2
1.4
0
2
4
6
8
10
n=5
n=10
n=20
n=40
n=75
n=150
n=200
n=300

Skills of several VEnKF analyses, N=10



mLETKF (N=150) vs. EKF



Overview

3D-Var Cost function

Algorithm

Minimize

$$
\begin{aligned}
J(\mathbf{x}(t_i)) &= J_b + J_o \\
&= \frac{1}{2}(\mathbf{x}(t_i) - \mathbf{x}^b(t_i))^{\mathrm{T}}\mathbf{B}_0^{-1}(\mathbf{x}(t_i) - \mathbf{x}^b(t_i)) \\
&+ \frac{1}{2}(H(\mathbf{x}(t_i)) - \mathbf{y}_i^o)^{\mathrm{T}}\mathbf{R}^{-1}(H(\mathbf{x}(t_i)) - \mathbf{y}_i^o)
\end{aligned}
$$

4D-Var Cost function

Algorithm

Minimize

$$
\begin{aligned}
J(\mathbf{x}(t_0)) &= J_b + J_o \\
&= \frac{1}{2}(\mathbf{x}(t_0) - \mathbf{x}^b(t_0))^{\mathrm{T}}\mathbf{B}_0^{-1}(\mathbf{x}(t_0) - \mathbf{x}^b(t_0)) \\
&+ \frac{1}{2}\sum_{i=0}^{n}(H(M(t_i,t_0)(\mathbf{x}(t_0))) - \mathbf{y}_i^o)^{\mathrm{T}}\mathbf{R}^{-1}(H(M(t_i,t_0)(\mathbf{x}(t_0))) - \mathbf{y}_i^o)
\end{aligned}
$$

EnKF Cost functions

Algorithm

Minimize

$$(\mathbf{P}^f(t_i))^{-1} = (\beta \mathbf{B}_0 + (1-\beta)\frac{1}{N}\mathbf{X}^f(t_i)\mathbf{X}^f(t_i)^{\mathrm{T}})^{-1}$$

Minimize

$$\ell(\mathbf{x}^a(t_i)|\mathbf{y}_i^o)$$
$$= (\mathbf{y}_i^o - H(\mathbf{x}^a(t_i)))^{\mathrm{T}}\mathbf{R}^{-1}(\mathbf{y}_i^o - H(\mathbf{x}^a(t_i)))$$
$$+ \frac{1}{N}\sum_{j=1}^{N}(\mathbf{x}_j^f(t_i) - \mathbf{x}^a(t_i))^{\mathrm{T}}(\mathbf{P}^f(t_i))^{-1}(\mathbf{x}_j^f(t_i) - \mathbf{x}^a(t_i))$$

EDA Cost functions - 1

Algorithm

Minimize, for each perturbation $\delta_k \mathbf{y}_j^o$

$$\begin{aligned} J(\mathbf{x}(t_{i-1})) &= J_b + J_o \\ &= \frac{1}{2}(\mathbf{x}(t_0) - \mathbf{x}^b(t_0))^{\mathrm{T}}\mathbf{B}_0^{-1}(\mathbf{x}(t_0) - \mathbf{x}^b(t_0)) \\ &+ \frac{1}{2}\sum_{i=0}^{n}(H(M(t_i,t_0)(\mathbf{x}(t_0))) - (\mathbf{y}_j^o + \delta_k \mathbf{y}_j^o))^{\mathrm{T}}\mathbf{R}^{-1} \\ &\times (H(M(t_i,t_0)(\mathbf{x}(t_0))) - (\mathbf{y}_j^o + \delta_k \mathbf{y}_j^o)) \end{aligned}$$

EDA Cost functions - 2

Algorithm

Minimize

$$(\mathbf{P}^a(t_j))^{-1} = (\beta \mathbf{B}_0 + (1-\beta)\frac{1}{N}\mathbf{X}^f(t_j)\mathbf{X}^f(t_j)^{\mathrm{T}})^{-1}, \quad j = t_{i-1} \ldots t_i$$

EDA Cost functions - 3

Algorithm

Minimize

$$J(\mathbf{x}(t_{i-1})) = J_b + J_o + J_M$$

$$= \frac{1}{2} \sum_{j=t_{i-1}}^{t_i} (\mathbf{x}(t_j) - \mathbf{x}^b(t_j))^{\mathrm{T}} (\mathbf{P}^a(t_j))^{-1} (\mathbf{x}(t_j) - \mathbf{x}^b(t_j))$$

$$+ \frac{1}{2} \sum_{i=0}^{n} (H(\mathbf{x}(t_j)) - \mathbf{y}_j^o)^{\mathrm{T}} \mathbf{R}^{-1} (H(\mathbf{x}(t_j)) - \mathbf{y}_j^o)$$

$$+ \frac{1}{2} \sum_{j=t_{i-1}}^{t_i} (\mathbf{x}(t_j) - M(t_j, t_{j-1})(\mathbf{x}(t_{j-1})))^{\mathrm{T}} \mathbf{Q}^{-\mathbf{1}} (\mathbf{x}(\mathbf{t_j}) - M(\mathbf{t_j}, \mathbf{t_{j-1}})(\mathbf{x}(\mathbf{t_{j-1}}))),$$

VEnKF Cost functions

Algorithm

Minimize

$$(\mathbf{P}^f(t_i))^{-1} = (\beta \mathbf{B}_0 + (1-\beta)\frac{1}{N}\mathbf{X}^f(t_i)\mathbf{X}^f(t_i)^{\mathrm{T}} + \mathbf{Q})^{-1}$$

Minimize

$$\begin{aligned}
&\ell(\mathbf{x}^a(t_i)|\mathbf{y}_i^o) \\
&= (\mathbf{y}_i^o - H(\mathbf{x}^a(t_i)))^{\mathrm{T}}\mathbf{R}^{-1}(\mathbf{y}_i^o - H(\mathbf{x}^a(t_i))) \\
&+ (\mathbf{x}^f(t_i) - \mathbf{x}^a(t_i))^{\mathrm{T}}(\mathbf{P}^f(t_i))^{-1}(\mathbf{x}^f(t_i) - \mathbf{x}^a(t_i))
\end{aligned}$$

VKF Cost functions

Algorithm

Minimize

$$(\mathbf{P}^f(t_i))^{-1} = (\mathbf{M}_i\mathbf{P}^a(t_{i-1})\mathbf{M}_i^{\mathrm{T}} + \mathbf{Q})^{-1}$$

Minimize

$$\begin{aligned}
&\lambda(\mathbf{x}^a(t_i)|\mathbf{y}_i^o) \\
&= (\mathbf{y}_i^o - \mathbf{H}_i\mathbf{x}^a(t_i))^{\mathrm{T}}\mathbf{R}^{-1}(\mathbf{y}_i^o - \mathbf{H}_i\mathbf{x}^a(t_i)) \\
&+ (\mathbf{x}^f(t_i) - \mathbf{x}^a(t_i))^{\mathrm{T}}(\mathbf{P}^f(t_i))^{-1}(\mathbf{x}^f(t_i) - \mathbf{x}^a(t_i))
\end{aligned}$$

mLETKF Cost function

Algorithm

Minimize

$$J(\mathbf{w}) = \beta(n-1)\mathbf{w}^{\mathrm{T}}\mathbf{w} + (1-\beta)\times$$
$$(\mathbf{y}^f - H(\mathbf{x}_k^a(t_i)) - \mathbf{w}^{\mathrm{T}}Y)^{\mathrm{T}}\mathbf{R}^{-1}(\mathbf{y}^f - H(\mathbf{x}_k^a(t_i)) - \mathbf{w}^{\mathrm{T}}Y)$$

Parallelism - 1

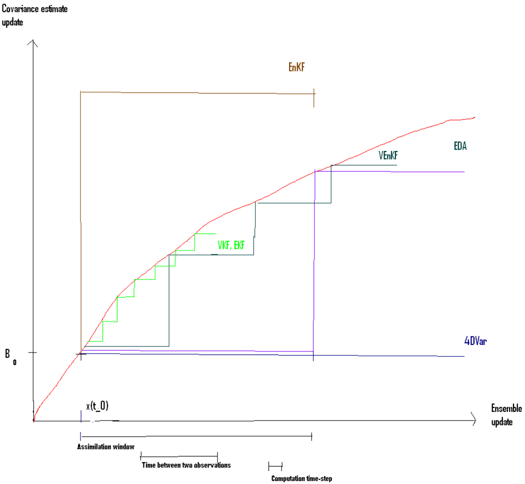
- Ensemble methods generally use their ensemble as a sample to approximate error covariance
- But the most accurate methods, such as 3DVar, 4DVar and VKF, are based on Krylov space approximation of the covariance
- Krylov space approximation is inherently serial, since a Krylov space is defined by iterative application of an operator on a vector:

$$\mathbf{K}_k(A, \mathbf{b}) = \mathrm{Span}(\mathbf{b}, A\mathbf{b}, A^2\mathbf{b}, \ldots, A^k\mathbf{b})$$

Parallelism - 2

- EnKF variants can be embarrassingly parallel and evolve its ensemble in parallel
- But the most accurate of them, such as VEnKF and EDA, still use a Krylov space method to approximate covariance: LBFGS, CG or Lanczos
- These methods are therefore inherently sequential, too
- The best accuracy of analysis in L95 tests follows from frequent and smooth updating of both the ensemble and the error covariance estimate

Updates to Ensemble and Error Covariance



Overview

Conclusions - 1

- VKF performs as well as EKF, with a computational cost comparable to 4D-Var, on Lorentz '95
- VEnKF is asymptotically as good as EKF or VKF in forecast skill, but can be run without an adjoint code
- VEnKF attains equal quality to EKF only on large ensemble sizes, but
- VEnKF performs better than EnKF especially with small ensemble size

Conclusions - 2

- The more frequent the inter-linked updates of the ensemble and the error covariance estimate, the more accurate the analysis
- There appears to be a trade-off between the accuracy of an assimilation method and its parallelism that needs to be decided by experiments

Thank You

Thank You!