# Variational Ensemble Kalman Filtering on Parallel Computers

Idrissa S. Amour, Harri Auvinen, Heikki Haario, Tuomo Kauranne

Lappeenranta University of Technology

13th ECMWF Workshop on the Use of High Performance Computing in Meteorology
November 6, 2008

# Overview

# 3D Variational Assimilation (3D-Var)

## Algorithm

**Minimize**

$$\begin{aligned} J(\mathbf{x_0}) &= J_b + J_o \\ &= \frac{1}{2}(\mathbf{x_b} - \mathbf{x_0})^{\mathrm{T}} \mathbf{S}_{apr}^{-1} (\mathbf{x_b} - \mathbf{x_0}) \\ &+ \frac{1}{2}(\mathbf{y}(0) - \mathcal{K}_t(\mathbf{x_0}))^{\mathrm{T}} \mathbf{Se}_t^{-1} (\mathbf{y}(0) - \mathcal{K}_t(\mathbf{x_0})), \end{aligned}$$

# 3D Variational Assimilation (3D-Var)

## Where

- $\mathbf{x_0}$ *is the analysis at time 0*
- $\mathbf{x_b}$ *is the background at time 0*
- $\mathbf{y}$ *is the vector of observations at time 0*
- $\mathbf{S}_{apr}$ *is the background error covariance matrix*
- $\mathbf{Se}_t$ *is the observation error covariance matrix*
- $\mathcal{K}_t$ *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**

$$\begin{aligned} J(\mathbf{x_0}) &= J_b + J_o \\ &= \frac{1}{2}(\mathbf{x_b} - \mathbf{x_0})^{\mathrm{T}} \mathbf{S}_{apr}^{-1} (\mathbf{x_b} - \mathbf{x_0}) \\ &+ \frac{1}{2} \sum_{t=0}^{T} (\mathbf{y}(t) - \mathcal{K}_t(\mathcal{M}_t(\mathbf{x_0})))^{\mathrm{T}} \mathbf{Se}_t^{-1} (\mathbf{y}(t) - \mathcal{K}_t(\mathcal{M}_t(\mathbf{x_0}))), \end{aligned}$$

# 4D Variational Assimilation (4D-Var)

## Where

- $\mathbf{x_0}$ *is the analysis at the beginning of the assimilation window*
- $\mathbf{x_b}$ *is the background at the beginning of the assimilation window*
- $\mathbf{S}_{apr}$ *is the background error covariance matrix*
- $\mathbf{Se}_t$ *is the observation error covariance matrix*
- $\mathcal{M}_t$ *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**

$$\begin{aligned}
\mathbf{x}_a(t) &= \mathcal{M}_t(\mathbf{x}_{est}(t-1)) \\
\mathbf{S}_a(t) &= \mathbf{M}_t\mathbf{S}_{est}(t-1)\mathbf{M}_t^{\mathrm{T}} + \mathbf{SE}_t \\
\mathbf{G}_t &= \mathbf{S}_a(t)\mathbf{K}_t^{\mathrm{T}}(\mathbf{K}_t\mathbf{S}_a(t)\mathbf{K}_t^{\mathrm{T}} + \mathbf{Se}_t)^{-1} \\
\mathbf{x}_{est}(t) &= \mathbf{x}_a(t) + \mathbf{G}_t(\mathbf{y}(t) - \mathcal{K}_t(\mathbf{x}_a(t))) \\
\mathbf{S}_{est}(t) &= \mathbf{S}_a(t) - \mathbf{G}_t\mathbf{K}_t\mathbf{S}_a(t),
\end{aligned}$$

# The Extended Kalman Filter (EKF)

## Where

- $\mathbf{x_a}$ *is the prediction*
- $\mathbf{x_{est}}$ *is the analysis*
- $\mathbf{S}_a$ *is the prediction error covariance matrix*
- $\mathbf{S}_{est}$ *is the analysis error covariance matrix*
- $\mathbf{SE}_t$ *is the model error covariance matrix*
- $\mathbf{G}_t$ *is the Kalman gain matrix*

# 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* $\mathbf{S}_a$ *is a huge full matrix*

# The Variational Kalman Filter (VKF)

## Algorithm

**Iterate in time**

**Step 0:** *Select an initial guess* $\mathbf{x}_{est}(0)$ *and a covariance* $\mathbf{S}_{est}(0)$, *and set* $t = 1$.

**Step 1:** *Compute the evolution model state estimate and the prior covariance estimate:*
*(i) Compute* $\mathbf{x}_a(t) = \mathcal{M}_t(\mathbf{x}_{est}(t-1))$;
*(ii) Approximate* $(\mathbf{S}_a(t))^{-1} = (\mathbf{M}_t \mathbf{S}_{est}(t-1)\mathbf{M}_t^{\mathrm{T}} + \mathbf{SE}_t)^{-1}$ *by the LBFGS method;*

## Algorithm

**Step 2:** *Compute the Variational Kalman filter state estimate and the posterior covariance estimate:*
*(i) Minimize*

$$\ell(\mathbf{x}_{est}(t)|\mathbf{y}) = (\mathbf{x}_a(t) - \mathbf{x}_{est}(t))^{\mathrm{T}}(\mathbf{S}_a(t))^{-1}(\mathbf{x}_a(t) - \mathbf{x}_{est}(t)) +$$

$$(\mathbf{y} - \mathcal{K}_t(\mathbf{x}_{est}(t)))^{\mathrm{T}}(\mathbf{Se}_t)^{-1}(\mathbf{y} - \mathcal{K}_t(\mathbf{x}_{est}(t)))\ ;$$

*by the LBFGS method;*

*(ii) Store the result of the minimization as a VKF estimate* $\mathbf{x}_{est}(t)$*;*
*(iii) Store the limited memory approximation to* $\mathbf{S}_{est}(t)$*;*

**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*
- $\mathbf{S}_a(t)$ *and* $\mathbf{S}_{est}(t)$ *are kept in vector format, as a sum of a diagonal or sparse background* $\mathbf{S}_{apr}$ *and a low rank dynamical component* $\breve{\mathbf{S}}_a(t)$ *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*

# The Variational Ensemble Kalman Filter (VEnKF)

## Properties

- *The goal of VEnkF 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*
- *VEnKF 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, and not the ensemble mean or background*

# The Variational Ensemble Kalman Filter (VEnKF)

## 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:*

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

$$(y_{apr} - \mathcal{K}(\mathbf{x}_a^{(i)}(t)) - Y\mathbf{w})^{\mathrm{T}}(\mathbf{Se}_t)^{-1}(y_{apr} - \mathcal{K}(\mathbf{x}_a^{(i)}(t)) - Y\mathbf{w})$$

# The Variational Ensemble Kalman Filter (VEnKF)

## Where

- $y_{apr}$ *is the synthetic observation vector of the prior* $y_{apr} = \mathcal{K}(\mathbf{x}_{apr}(t))$
- $\mathbf{w}$ *is the vector of the weights* $w^{(i)}$ *of each ensemble member* $\mathbf{x}_a^{(i)}(t)$
- $Y$ *is the matrix of synthetic observations of each ensemble member* $Y^{(i)} = \mathcal{K}(\mathbf{x}_a^{(i)}(t))$
- $n$ *is the ensemble size*
- $\beta$ *is an empirical weight factor between 0 and 1*

# The Variational Ensemble Kalman Filter (VEnKF)

## Algorithm

**Iterate in time**

**Step 0:** *Initialize the background state* $\mathbf{x}_{apr}(0)$ *and the ensemble members* $\mathbf{x}_{est}^{(i)}(0)$ *for* $i = 1, \ldots, n$

**Step 1:** *Compute* $\mathbf{x}_a^{(i)}(t) = \mathcal{M}_t(\mathbf{x}_{est}^{(i)}(t-1))$ *and* $\mathbf{x}_{apr}(t) = \mathcal{M}_t(\mathbf{x}_{apr}(t-1))$;

**Step 2:** *Perturb the members* $\mathbf{x}_a^{(i)}(t)$ *and assemble them in matrix* $\Psi$;

**Step 3:** *Compute the matrix* $X_a(t) : X_a^{(i)}(t) = \mathbf{x}_{apr}(t) - \Psi^{(i)}$;

**Step 4:** *Compute the matrix* $Y_a(t) : Y_a^{(i)}(t) = \mathcal{K}(\mathbf{x}_a^{(i)}(t)) - \mathcal{K}(\mathbf{x}_{apr}(t))$;

# The Variational Ensemble Kalman Filter (VEnKF)

## Algorithm

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

**Step 6:** *Compute the analysis* $\mathbf{x}_{apr}(t) = \mathbf{x}_{apr}(t) + X_a(t)\mathbf{w}$

**Step 7:** *Compute the background ensemble* $X_{est}(t) : X_{est}^{(i)}(t) = X_a^{(i)}(t) + X_a(t)\mathbf{w}$

**Step 8:** *Update* $t := t + 1$ *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*

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

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

# 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* $\mathbf{Se}_t = 0.15\sigma_{clim}\mathbf{I}$ *and*
- *prediction error covariance* $\mathbf{SE}_t = 0.5\sigma_{clim}\mathbf{I}$.

# 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*

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

# The Lorenz '95 Model

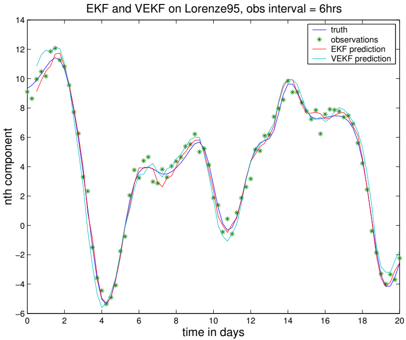
## Where

- *$\mathcal{M}_n$ 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*

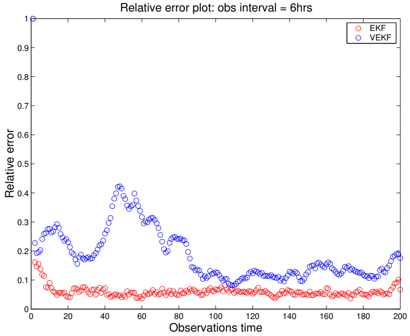
$$[\textbf{forecast\_skill}]_i = \frac{1}{\sigma_{\text{clim}}} \sqrt{\frac{1}{100} \sum_{j \in \mathcal{I}} [\textbf{forecast\_error}_j]_i},$$

*i=1,. . . ,20,*

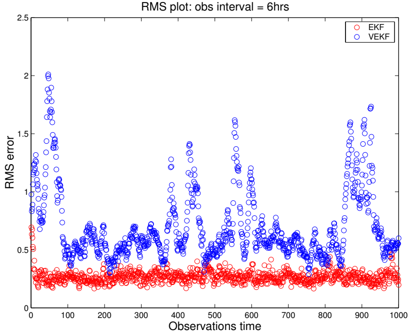
# Computational Results



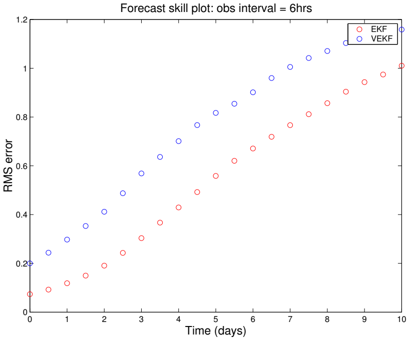
# Computational Results



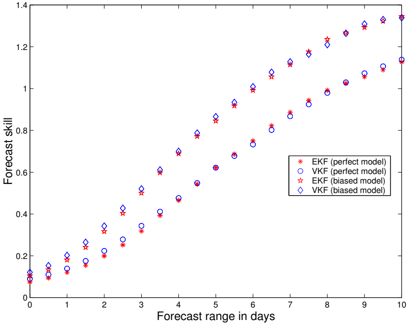
# Computational Results



# Computational Results



# Computational Results



# Overview

# Conclusions

- VKF performs as well as EKF, with a computational cost comparable to 4D-Var, on Lorentz '95
- VEnKF is less good than EKF or VKF in forecast skill, but can be run without an adjoint code
- VEnKF is embarrassingly parallel
- Another version of VKF also parallelizes well, but has a higher serial complexity
- VKF and VEnKF are attractive candidates to replace 4D-Var and Optimum Interpolation, respectively, in operational weather data assimilation

# Thank You

**Thank You!**