Stabilized approximate Kalman filter and its extension towards parallel implementation

An example of two-layer Quasi-Geostrophic model + CUDA-accelerated shallow water

Alex Bibov, Heikki Haario

10/2014
Contents

• Data assimilation at glance
• Approximating Extended Kalman filter using BFGS: instability
• Stabilizing correction for approximate EKF
• Combined state space and parallel filtering
• Current test case: the Two-Layer Quasi-Geostrophic model
• Experimental results

• The next test case: large-scale Shallow Water model
• CUDA accelerated implementation
• Example runs
Consider coupled system of stochastic equations:

\[ x_{k+1} = M_k(x_k) + \varepsilon_k, \]
\[ y_{k+1} = H_{k+1}(x_{k+1}) + \eta_{k+1}, \]

where \( x_k \in \mathbb{R}^n \) describes system state at time instance \( k \), \( y_{k+1} \in \mathbb{R}^m \) is observed data obtained at time instance \( k + 1 \), \( M_k \) is state transition operator, and \( H_{k+1} \) is observation mapping describing how system state relates to the observed data at a certain time instance, \( \varepsilon_k \) and \( \eta_{k+1} \) are random terms that model prediction and observation uncertainties.

The task: given the estimate \( x_k^{est} \) of state \( x_k \) and observation \( y_{k+1} \) derive estimate \( x_{k+1}^{est} \).
Approximating the EKF

- Denote $C_k = \text{Cov}(x_k)$, $C_{\varepsilon k} = \text{Cov}(\varepsilon_k)$, $C_{\eta_{k+1}} = \text{Cov}(\eta_{k+1})$

- Recall formulation of the Extended Kalman filter:
  1. Run the forecast model: $x_{k+1}^p = M_k(x_k)$,
  2. Estimate forecast covariance: $C_{k+1}^p = \text{Cov}(x_{k+1}^p) = M_k^T C_k M_k^{AD} + C_{\varepsilon_k}$,
  3. Compute the Kalman gain: $G_{k+1} = C_{k+1}^p H_{k+1}^{AD} (H_{k+1}^{TL} C_{k+1}^p H_{k+1}^{AD} + C_{\eta_{k+1}})^{-1}$,
  4. Compute state estimate: $x_{k+1}^{est} = x_{k+1}^p + G_{k+1}(y_{k+1} - H_{k+1}^{TL} x_{k+1}^p)$,
  5. Find covariance of the estimate: $C_{k+1}^{est} = C_{k+1}^p - G_{k+1} H_{k+1}^{TL} C_{k+1}^p$.

- Problem: Large dimension of state $x_k$ induces issues at covariance matrix storage
- Solution: approximate problematic matrices the same way as it is done for Hessians of large-scale optimization problems
EKF approximation based on BFGS*

1. Run forecast model: \( x_{k+1}^p = \mathcal{M}_k(x_k) \),
2. At the code level define operator implementing forecast covariance matrix:
   \[ C_{k+1}^p = M_k^T L x_{k+1}^p M_k^A D + C_{\varepsilon_k} \],
3. Apply L-BFGS minimization to auxiliary quadratic cost function:
   \[ f(x) = x^T A x - x^T b, \]
   where \( A = H_{k+1}^T L C_{k+1}^p H_{k+1}^A D + C_{\eta_{k+1}} \), and \( b = y_{k+1} - H_{k+1}^T L x_{k+1}^p \),
4. Assign \( x^* \) to the minimizer of \( f(x) \) and \( B^* \) to approximation of Hessian matrix \( A \) produced as part of output from L-BFGS
5. Compute state estimate: \( x_{k+1}^{est} = x_{k+1}^p + C_{k+1} H_{k+1}^A D x^* \)
6. Approximate covariance matrix of the estimate by applying L-BFGS minimization to a quadratic cost function with Hessian defined as follows:
   \[ C_{k+1}^p - C_{k+1}^p H_{k+1}^A D B^* H_{k+1}^T L C_{k+1}^p \]

*See H. Auvinen et. al. “The variational Kalman filter and an efficient implementation using limited memory BFGS”
BFGS EKF: Instability problem

- Approximate estimate covariance matrix $C^{p}_{k+1} - C^{p}_{k+1}H^{AD}_{k+1}B^{*}H^{TL}_{k+1}C^{p}_{k+1}$ may have “non-physical” negative eigenvalues as $B^{*}$ is itself approximation of prior covariance projected onto the observation space:

$$B^{*} \approx \left(H^{TL}_{k+1}C^{p}_{k+1}H^{AD}_{k+1} + C_{\eta_{k+1}}\right)^{-1}$$

- L-BFGS on the other hand relies on the eigenvalues of Hessian being non-negative.

- We correct this problem by injecting “stabilizing correction”, i.e. we replace $B^{*}$ by $(2I - B^{*}A)B^{*}$.

- Let us denote $C^{p}_{k+1} - C^{p}_{k+1}H^{AD}_{k+1}(2I - B^{*}A)B^{*}H^{TL}_{k+1}C^{p}_{k+1}$ as $\hat{C}^{est}_{k+1}$.

**Lemma.** For any symmetric matrix $B^{*}$, the matrix $\hat{C}^{p}_{k+1}$ is non-negative. Moreover, as $B^{*} \rightarrow A^{-1}$ necessarily $\hat{C}^{est}_{k+1} \rightarrow C^{est}_{k+1}$ and the following inequalities hold:

$$\|\hat{C}^{est}_{k+1} - C^{est}_{k+1}\|_{Fr} \leq \|A\|\|H^{TL}_{k+1}C^{p}_{k+1}\|_{Fr}^{2} \|B^{*} - A^{-1}\|^{2},$$

$$\|\hat{C}^{est}_{k+1} - C^{est}_{k+1}\| \leq \|A\|\|H^{TL}_{k+1}C^{p}_{k+1}\|^{2} \|B^{*} - A^{-1}\|^{2}.$$
Current toy-case: the QG-model*

- The current test case for DA testing purposes is provided by Two-Layer Quasi-Geostrophic model:
  - Simulates “slow” wind motions
  - Resides on cylindrical surface vertically divided into two layers
  - The boundary conditions are periodic in zonal direction and fixed at the top and at the bottom of the cylinder
- The model is chaotic, dimension can be adjusted by changing resolution of the spatial grid
- Provides a neat toy-case, which can be run with no special hardware

*See C.Fandry and L.Leslie, “A two-layer quasi-geostrophic model of summer trough formation in the Australian subtropical easterlies”.
Current toy-case: the QG-model

- Governing equations with respect to unknown stream function $\psi_i(x, y)$
  
  $q_1 = \nabla^2 \psi_1 - F_1(\psi_1 - \psi_2) + \beta y,$
  
  $q_2 = \nabla^2 \psi_2 - F_2(\psi_2 - \psi_1) + \beta y + R_s,$
  
  $\frac{D_1 q_1}{Dt} = \frac{D_2 q_2}{Dt} = 0,$

  where $R_s = R_s(x, y)$ is orography surface,

  $\frac{D_i}{Dt} = \frac{\partial}{\partial t} + u_i \frac{\partial}{\partial x} + v_i \frac{\partial}{\partial y}$ and $\nabla \psi_i = (v_i, -u_i)$.

- The equations are numerically solved by combining finite-difference approximation of derivatives with semi-Lagrangian advection
Current toy-case: the QG-model

Layer interaction interface
QG-model: chaotic behavior
Numerical experiments: the QG-model

- Data assimilation performance was tested in emulated environment: we ran two instances of the qg-model at different resolutions and used one to emulate observations and the other to make predictions.
- Observations were collected from a sparse subset of the state vector elements.
- Predictions were made at lower resolution than the “truth” and the values of the depths of the model layers were biased.
- Sources of incoming observations were interpolated onto the spatial grid of lower-resolution model by bilinear interpolation.
- Estimation quality was measured by root mean square error.
- We run several experiments at different resolutions and with different number of observations employing stabilized BFGS EKF, usual uncorrected BFGS EKF, weak-constraint 4D-VAR and the parallel filter.
Convergence with and without the stabilizing correction

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>10-by-10</td>
<td>9-by-9</td>
<td>50</td>
<td>162</td>
</tr>
<tr>
<td>2</td>
<td>15-by-15</td>
<td>12-by-12</td>
<td>85</td>
<td>288</td>
</tr>
<tr>
<td>3</td>
<td>80-by-40</td>
<td>40-by-20</td>
<td>500</td>
<td>1600</td>
</tr>
</tbody>
</table>

Table 1: Benchmark options depending on model resolution.

<table>
<thead>
<tr>
<th>Benchmark</th>
<th>I</th>
<th>II</th>
<th>III</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>BFGS-EKF</td>
<td>SA-EKF</td>
<td>BFGS-EKF</td>
</tr>
<tr>
<td>BFGS vectors</td>
<td>5</td>
<td>0.5860</td>
<td>0.5508</td>
</tr>
<tr>
<td></td>
<td>10</td>
<td>0.5573</td>
<td>0.5445</td>
</tr>
<tr>
<td></td>
<td>15</td>
<td>0.4959</td>
<td>0.4788</td>
</tr>
<tr>
<td></td>
<td>20</td>
<td>0.4686</td>
<td>0.4695</td>
</tr>
<tr>
<td>EKF ref.</td>
<td>0.4247</td>
<td></td>
<td>0.4346</td>
</tr>
</tbody>
</table>

Table 2: Mean values of RMS error obtained from the benchmarks using varied capacity of the BFGS storage.
Parallel filter

- Consider combined state and observation vectors
  \[ \bar{x}_k = (x_{k-P+1}, x_{k-P+2}, \ldots, x_k), \]
  \[ \bar{y}_k = (y_{k-P+2}, y_{k-P+3}, \ldots, y_{k+1}). \]

- We extend transition and observation operators onto combined state space:
  \[ \bar{M}_k(\bar{x}_k) = (M_{k-P+1}(x_{k-P+1}), M_{k-P+2}(x_{k-P+2}), \ldots, M_k(x_k)), \]
  \[ \bar{H}_{k+1}(\bar{y}_k) = (H_{k-P+2}(x_{k-P+2}), H_{k-P+3}(x_{k-P+3}), \ldots, H_{k+1}(x_{k+1})). \]

- We call the data assimilation problem formulated for \( \bar{M}_k \) and \( \bar{H}_{k+1} \) the parallel filtering task.
Parallel filter: additional comments

- Model error covariance $C_{\varepsilon_k}$ and observation error covariance $C_{\eta_{k+1}}$ can be extended to combined state and observation spaces as follows:

$$
\bar{C}_{\varepsilon_k} = \begin{pmatrix} 
C_{\varepsilon_{k-P+1}} & \ldots & O \\
\vdots & \ddots & \vdots \\
0 & \ldots & C_{\varepsilon_k} 
\end{pmatrix},
$$

$$
\bar{C}_{\eta_{k+1}} = \begin{pmatrix} 
C_{\eta_{k-P+2}} & \ldots & O \\
\vdots & \ddots & \vdots \\
0 & \ldots & C_{\eta_{k+1}} 
\end{pmatrix}.
$$

- Adding non zero off-diagonal terms into definition of $\bar{C}_{\varepsilon_k}$ and $\bar{C}_{\eta_{k+1}}$ allows to account for time-correlated prediction and observation errors, which relaxes one of the classical assumptions used by derivation of the Kalman filter formulae.
Parallel filter: additional comments

- Allows to account for cross-time correlations between the states included into analysis
- Combines observations from several time steps, which should help in case of deficient observations
- Enables natural parallel implementation, as model propagations within combined state are executed independently
- Retrospective analysis of the older states are computed as part of the normal algorithm’s output with no extra outlay

**Main problem:** parallel filtering task is extremely large scale, which means that a highly-compressed packaging of covariance data is required.

**Solution:** Use L-BFGS approximation with stabilization introduced earlier.
Relation to the Weak-Constraint 4D-Var*

- Consider combined transition operator $\tilde{M}_k$ and combined observation mapping $\tilde{H}_{k+1}$. Assume that $x^b$ is a prior state estimate at time instance $k - P + 1$. Then weak-constraint 4D-Var estimate is calculated by minimizing the following cost function with respect to $x_k$:

$$l(\bar{x}_k | \bar{y}_k, x^b) = R_1(\bar{x}_k, \bar{y}_k) + R_2(\bar{x}_k) + R_3(x_{k-P+1}, x^b)$$

- $R_1(\bar{x}_k, \bar{y}_k)$ defines measure for observation discrepancy:

$$R_1(\bar{x}_k, \bar{y}_{k+1}) = \sum_{i=0}^{P-1} \|y_{k-P+1+i} - H_{k-P+2+i}(x_{k-P+1+i})\|_{C_{\eta_{k-P+1+i}}}^2.$$

- $R_2(\bar{x}_k)$ smoothing part, accounts for prediction errors:

$$R_2(\bar{x}_k) = \sum_{i=1}^{P-1} \|x_{k-P+1+i} - M_{k-P+i}(x_{k-P+i})\|_{Q_{k-P+i+1}}^2.$$

- $R_3(x_{k-P+1}, x^b)$ penalizes discrepancy with the prior:

$$R_3(x_{k-P+1}, x^b) = \|x_{k-P+1} - x^b\|_{B^{-1}}^2.$$

*See Y. Trémolet “Accounting for an imperfect model in 4D-Var”
Relation to the Weak-Constraint 4D-Var

- Weak-constraint 4D-Var employs the concept of time window composed of a few consequent states.
- Propagations of each state over the time are performed independently from each other and thus can be executed in parallel.
- It is allowed to have a “jump” $q_i$ between prediction $\mathcal{M}_i(x_i)$ and the next state $x_{i+1}$. This accounts for prediction error.
- Forecast is defined by prediction made from the state located at the end of the window.
Relation to the Weak-Constraint 4D-Var

- Estimation task of the parallel filter can be reformulated in terms of the following cost function, which should be minimized with respect to $\tilde{x}_k$:

$$l(\tilde{x}_k|\tilde{y}_k, \tilde{x}_k^p) = R_1(\tilde{x}_k, \tilde{y}_k) + R_2(\tilde{x}_k, \tilde{x}_k^p).$$

- $R_1(\tilde{x}_k, \tilde{y}_k)$ penalizes discrepancy between observation and the estimate:

$$R_1(\tilde{x}_k, \tilde{y}_k) = \sum_{i=0}^{P-1} \|y_{k-P+1+i} - H_{k-P+1+i}(x_{k-P+1+i})\|_C^{-1},$$

- $R_2(\tilde{x}_k, \tilde{x}_k^p)$ penalizes discrepancy between the estimate and the forecast:

$$R_2(\tilde{x}_k, \tilde{x}_k^p) = \|\tilde{x}_k - \tilde{x}_k^p\|_{\eta_k}^{-1},$$

where $\tilde{x}_k^p = \tilde{M}_{k-1}(\tilde{x}_{k-1}^\text{est})$.

- If $C_{k+1}^\text{est}$ is block-diagonal (it is usually not in practice), then $R_2(\tilde{x}_k, \tilde{x}_k^p)$ can be reduced to the following sum:

$$R_1(\tilde{x}_k, \tilde{y}_k) = \sum_{i=0}^{P-1} \|x_{k-P+1+i} - x_{k-P+1+i}^p\|_{(C_{k-P+1+i}^\text{est})^{-1}}.$$
Relation to the Weak-Constraint 4D-Var

- If $\tilde{C}^{est}_{k+1}$ is block-diagonal then parallel filtering effectively reduces to weak-constraint 4D-Var with fixed predictions $x^p_i = M_{i-1}(x_{i-1})$.
- If parameter $x^p_i$ in the parallel filtering likelihood function is allowed to vary during minimization and $\tilde{C}^{est}_{k+1}$ is block-diagonal, then parallel filtering becomes equivalent to the weak-constraint 4D-Var.
- In parallel filtering we do not need to assume block-diagonal approximations of covariance matrices, which enables cross-correlations between time sub-windows. In Weak-Constraint 4D-Var the same effect is achieved by unfixed value of $x^p_i$.
- Dimension of the data assimilation problem defined by parallel filtering can be effectively treated by low-memory approaches provided by L-BFGS EKF approximation with stabilizing correction.
Numerical experiments: the QG-model

- The total window comprised three 6-hour sub-windows (18-hour analysis)
- Dimension of combined state for 18-hour window was 4800
- BFGS storage capacity was set to 20 vectors
- Quality of obtained estimates was measured by root mean square error
- The results were compared against usual single-state SA-EKF and weak-constraint 4D-VAR
- Model used to simulate observations had spatial grid resolution 40-by-80 points in both layers
- Prediction model used 4-times smaller resolution of 20-by-40 points in both layers
- Integration time step was set to one hour of model time
Test of concept: 10 observations
Test of concept: 20 observations

Data assimilation

Retrospective analysis

Stabilized L-BFGS-EKF

Retrospective analysis 1
Retrospective analysis 2
Data assimilation
Stabilized L-BFGS-EKF
Test of concept: 30 observations

Retrospective analysis
Retrospective analysis 2
Data assimilation
Data assimilation
Stabilized L-BFGS-EKF
Test of concept: 200 observations
Future case: Large-Scale Shallow Water

\[
\begin{align*}
(\rho \frac{h}{t} + (hu)_x + (hv)_y) &= 0, \\
(hu)_t + \left( hu^2 + \frac{1}{2} gh^2 \right)_x + (huv)_y &= -ghB_x - gu\sqrt{u^2 + v^2}/C_z^2, \\
(hu)_t + (huv)_x + \left( hu^2 + \frac{1}{2} gh^2 \right)_y &= -ghB_y - gv\sqrt{u^2 + v^2}/C_z^2,
\end{align*}
\]

Here \( h \) denotes water elevation, \( u \) and \( v \) are horizontal and vertical velocity components, \( B_x \) and \( B_y \) denote gradient direction of the surface implementing topography, \( g \) is acceleration of gravity, \( C_z \) is the Chézy coefficient.

- It is possible to account for additional phenomena (e.g. wind stresses, friction etc.) by adjusting the right-hand-side part of the equations.

*See [http://www.sintef.no/Projectweb/Heterogeneous-Computing/Research-TOPICS/Shallow-Water/](http://www.sintef.no/Projectweb/Heterogeneous-Computing/Research-TOPICS/Shallow-Water/) for details on practical application of the model*
Numerics: Discretization by finite volumes

- Numerics: Kurganov-Petrova second-order well-balanced positivity preserving central-upwind scheme
- The problem is solved for a huge set of discretization cells that form a staggered grid.
Numerics: fitting with GPU architecture

Thread\((j,k)\)

\[
\begin{align*}
U_{j,k}^N & \quad U_{j,k}^W \\
U_{j,k}^E & \quad U_{j,k+1}^N \\
U_{j,k}^S & \quad U_{j+1,k}^N
\end{align*}
\]

Thread\((j+1,k)\)

\[
\begin{align*}
U_{j+1,k}^N & \quad U_{j+1,k}^W \\
U_{j+1,k}^E & \quad U_{j+1,k+1}^N \\
U_{j+1,k}^S & \quad U_{j+1,k+1}^S
\end{align*}
\]

Thread\((j,k+1)\)

\[
\begin{align*}
U_{j,k+1}^E & \quad U_{j+1,k}^E \\
U_{j,k+1}^W & \quad U_{j+1,k+1}^W \\
U_{j,k+1}^S & \quad U_{j+1,k+1}^S
\end{align*}
\]

Thread\((j+1,k+1)\)
Roadmap of the GPU implementation

- Single call to cudaMalloc(...) to allocate a huge linear block of memory. The needed part is then accessed by the offsets.
- Extensive use of the shared memory: neighboring cells propagate their “boundary conditions” between each other through the CUDA shared memory.
- No intermediate transfers to the host: all computations are done on the GPU-side.
- The grid is horizontally divided between all available GPUs. Pinned memory is used for data exchange to minimize the I/O workload (albeit, this part needs more testing).
- The serial part of the code is reduced to data initialization, hence the impact of the Amdahl’s law is minimal → the code scales very good with growth of the spatial resolution (one can run up to 3 000 000 dimensional shallow water in very this laptop!)
- Under certain conditions we were able to reach 100x performance boost over CPU-hosted implementation based on intel MKL routines.
Conclusion

- Presented an algorithm based on Kalman filter approximation, which is able to preserve stability when applied to large-scale dynamics.
- A further improvement for the approach based on parallelization is introduced.
- Both concepts are tested with a toy-case chaotic model, which can be made fairly large-scale by increasing spatial discretization.
- A new test model, which can be run at a very high resolution on widely available hardware is implemented (thanks to CUDA!)
Thank you for attention!