Parallelisation of 4D-Var in the time dimension using a saddlepoint algorithm

Mike Fisher

ECMWF

July 23, 2013
Outline

1. Introduction
2. Weak-Constraint 4D-Var
3. Characteristics of the problem
   - Parallelisation in the time dimension
     - The Saddle Point Formulation
     - Results from a toy system
4. Conclusions
Introduction

- 4D-Var is a statistical estimation method that is widely used for geoscience applications, especially Numerical Weather Prediction (NWP).
- It is used by many of the major NWP Centres (ECMWF, Met Office, Météo France, JMA, Canadian Met Service, etc.), as well as being used for ocean data-assimilation (e.g. NEMOVAR).
- It expresses the estimation problem as an optimisation problem.
- The task is to estimate a sequence of states, defined over a finite time interval (the “analysis window”), given an initial state (the “background” or “prior”) and a set of observations.
In this talk, I will concentrate on Weak-constraint 4D-Var.

Let us define the analysis window as $t_0 \leq t \leq t_{N+1}$

We wish to estimate the sequence of states $x_0 \ldots x_N$ (valid at times $t_0 \ldots t_N$), given:

- A prior $x_b$ (valid at $t_0$).
- A set of observations $y_0 \ldots y_N$ Each $y_k$ is a vector containing, typically, a large number of measurements of a variety of variables distributed spatially and in the time interval $[t_k, t_{k+1})$.

4D-Var is a maximum likelihood method. We define the estimate as the sequence of states that minimizes the cost function:

$$J(x_0 \ldots x_N) = -\log (p(x_0 \ldots x_N|x_b; y_0 \ldots y_N)) + \text{const}.$$
Weak-constraint 4D-Var

Using Bayes’ theorem, and assuming unbiased Gaussian errors, the weak-constraint 4D-Var cost function can be written as:

\[
J(x_0 \ldots x_N) = (x_0 - x_b)^T B^{-1} (x_0 - x_b) \\
+ \sum_{k=0}^{N} (\mathcal{H}_k(x_k) - y_k)^T R_k^{-1} (\mathcal{H}_k(x_k) - y_k) \\
+ \sum_{k=1}^{N} (q_k - \bar{q})^T Q_k^{-1} (q_k - \bar{q}) .
\]

where \( q_k = x_k - \mathcal{M}_k(x_{k-1}) \)

\( B, R_k \) and \( Q_k \) are covariance matrices of background, observation and model error. \( \mathcal{H}_k \) is an operator that maps model variables \( x_k \) to observed variables \( y_k \), and \( \mathcal{M}_k \) represents an integration of the numerical model from time \( t_{k-1} \) to time \( t_k \).
Weak-constraint 4D-Var

- 4D-Var is computationally expensive, and NWP is a real-time activity.
- It is usual to reduce the computational cost of 4D-Var by framing it as a simplified Gauss-Newton iteration in which a sequence of quadratic problems is solved.
- The scale of the problem, and the real-time constraints of weather forecasting require us to solve the 4D-Var problem on highly parallel computers.
- We are reaching the limits of what can be achieved by a purely spatial decomposition of the problem.
- We need a new dimension over which to parallelise the problem.
Weak Constraint 4D-Var: Quadratic Inner Loops

The inner loops of weak-constraint 4D-Var minimise:

\[ J(\delta x_0, \ldots, \delta x_N) = \frac{1}{2} \left( \delta x_0 - b \right)^T B^{-1} \left( \delta x_0 - b \right) \]

\[ + \frac{1}{2} \sum_{k=0}^{N} \left( H_k \delta x_k - d_k \right)^T R_k^{-1} \left( H_k \delta x_k - d_k \right) \]

\[ + \frac{1}{2} \sum_{k=1}^{N} \left( \delta q_k - c_k \right)^T Q_k^{-1} \left( \delta q_k - c_k \right) \]

where \( \delta q_k = \delta x_k - M_k \delta x_{k-1} \),

and where \( b, c_k \) and \( d_k \) come from the outer loop:

\[ b = x_b - x_0 \]

\[ c_k = \bar{q} - q_k \]

\[ d_k = y_k - H_k(x_k) \]
Weak Constraint 4D-Var: Quadratic Inner Loops

We simplify the notation by defining some 4D vectors and matrices:

\[
\delta x = \begin{pmatrix}
\delta x_0 \\
\delta x_1 \\
\vdots \\
\delta x_N
\end{pmatrix}
\quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad
\delta p = \begin{pmatrix}
\delta q_0 \\
\delta q_1 \\
\vdots \\
\delta q_N
\end{pmatrix}
\]

These vectors are related through \( \delta q_k = \delta x_k - M_k \delta x_{k-1} \).

We can write this relationship in matrix form as:

\[
\delta p = L \delta x
\]

where:

\[
L = \begin{pmatrix}
I & & & \\
-M_1 & I & & \\
& -M_2 & I & \\
& & \ddots & \ddots \\
& & & -M_N & I
\end{pmatrix}
\]
We will also define:

\[
R = \begin{pmatrix}
R_0 & 0 & 0 & \cdots & 0 \\
0 & R_1 & 0 & \cdots & 0 \\
0 & 0 & \ddots & \cdots & 0 \\
0 & 0 & \cdots & R_N & 0 \\
0 & 0 & \cdots & 0 & R_N
\end{pmatrix}, \quad D = \begin{pmatrix}
B & 0 & 0 & \cdots & 0 \\
0 & Q_1 & 0 & \cdots & 0 \\
0 & 0 & \ddots & \cdots & 0 \\
0 & 0 & \cdots & Q_N & 0 \\
0 & 0 & \cdots & 0 & Q_N
\end{pmatrix},
\]

\[
H = \begin{pmatrix}
H_0 & 0 & 0 & \cdots & 0 \\
0 & H_1 & 0 & \cdots & 0 \\
0 & 0 & \ddots & \cdots & 0 \\
0 & 0 & \cdots & H_N & 0 \\
0 & 0 & \cdots & 0 & H_N
\end{pmatrix}, \quad b = \begin{pmatrix}
b \\
c_1 \\
\vdots \\
c_N
\end{pmatrix}, \quad d = \begin{pmatrix}
d_0 \\
d_1 \\
\vdots \\
d_N
\end{pmatrix}.
\]
Weak Constraint 4D-Var: Quadratic Inner Loops

With these definitions, we can write the inner-loop cost function either as a function of $\delta x$:

$$J(\delta x) = (L\delta x - b)^T D^{-1} (L\delta x - b) + (H\delta x - d)^T R^{-1} (H\delta x - d)$$

Or as a function of $\delta p$:

$$J(\delta p) = (\delta p - b)^T D^{-1} (\delta p - b) + (HL^{-1}\delta p - d)^T R^{-1} (HL^{-1}\delta p - d)$$
Weak Constraint 4D-Var: Quadratic Inner Loops

\[ L = \begin{pmatrix} I & & & \\ -M_1 & I & & \\ & -M_2 & I & \\ & & \ddots & \ddots \\ & & & -M_N & I \end{pmatrix} \]

\[ \delta p = L \delta x \] can be done in parallel: \[ \delta q_k = \delta x_k - M_k \delta x_{k-1} . \]
We know all the \( \delta x_{k-1} \)'s. We can apply all the \( M_k \)'s simultaneously.
An algorithm involving only \( L \) is time-parallel.

\[ \delta x = L^{-1} \delta p \] is sequential: \[ \delta x_k = M_k \delta x_{k-1} + \delta q_k . \]
We have to generate each \( \delta x_{k-1} \) in turn before we can apply the next \( M_k \).
An algorithm involving \( L^{-1} \) is sequential.
Forcing Formulation

\[ J(\delta p) = (\delta p - b)^T D^{-1}(\delta p - b) + (HL^{-1}\delta p - d)^T R^{-1}(HL^{-1}\delta p - d) \]

- This version of the cost function is **sequential**, since it contains \( L^{-1} \).
- The form of cost function resembles that of strong-constraint 4D-Var, and it can be minimised using techniques that have been developed for strong-constraint 4D-Var.
- In particular, we can precondition it using \( D^{1/2} \) to diagonalise the first term:

\[ J(\chi) = \chi^T \chi + (HL^{-1}\delta p - d)^T R^{-1}(HL^{-1}\delta p - d) \]

where \( \delta p = D^{1/2}\chi + b \).
4D State Formulation

\[ J(\delta x) = (L\delta x - b)^T D^{-1} (L\delta x - b) + (H\delta x - d)^T R^{-1} (H\delta x - d) \]

- This version of the cost function is parallel. It does not contain \( L^{-1} \).
- Unfortunately, it is difficult to precondition.
4D State Formulation

\[ J(\delta x) = (L\delta x - b)^T D^{-1} (L\delta x - b) + (H\delta x - d)^T R^{-1} (H\delta x - d) \]

The usual method of preconditioning used in 4D-Var defines a control variable \( \chi \) that diagonalizes the first term of the cost function

\[ \delta x = L^{-1} (D^{1/2} \chi + b) \]

With this change-of-variable, the cost function becomes:

\[ J(\chi) = \chi^T \chi + (H\delta x - d)^T R^{-1} (H\delta x - d) \]

But, we have introduced a sequential model integration (i.e. \( L^{-1} \)) into the preconditioner.

Replacing \( L^{-1} \) by something cheaper destroys the preconditioning because \( D \) is extremely ill-conditioned.
If we approximate $L$ by $\tilde{L}$ in the preconditioner, the Hessian matrix of the first term of the cost function becomes

$$D^{1/2}\tilde{L}^{-T}LD^{-1}\tilde{L}^{-1}D^{1/2}$$

Because $D$ is highly ill-conditioned, this matrix is not close to the identity matrix unless $\tilde{L}$ is a very good approximation of $L$. 
A third possibility for minimising the cost function is the Lagrangian dual (known as 4D-PSAS in the meteorological community):

$$\delta x = L^{-1}D L^{-T}H^T \delta w$$

where

$$\delta w = \arg \min_{\delta w} F(\delta w)$$

and where

$$F(\delta w) = \frac{1}{2} \delta w^T (R + HL^{-1}DL^{-T}H^T) \delta w + \delta w^T z$$

with $z$ a complicated expression involving $b$ and $d$. Clearly, this is a sequential algorithm, since it contains $L^{-1}$. 
The Saddle Point Formulation

\[ J(\delta x) = (L\delta x - b)^T D^{-1} (L\delta x - b) + (H\delta x - d)^T R^{-1} (H\delta x - d) \]

At the minimum:

\[ \nabla J = L^T D^{-1} (L\delta x - b) + H^T R^{-1} (H\delta x - d) = 0 \]

Define:

\[ \lambda = D^{-1} (b - L\delta x), \quad \mu = R^{-1} (d - H\delta x) \]

Then:

\[ \begin{align*}
    D\lambda + L\delta x &= b \\
    R\mu + H\delta x &= d \\
    L^T\lambda + H^T\mu &= 0
\end{align*} \quad \Rightarrow \quad \begin{pmatrix} D & 0 & L \\ 0 & R & H \\ L^T & H^T & 0 \end{pmatrix} \begin{pmatrix} \lambda \\ \mu \\ \delta x \end{pmatrix} = \begin{pmatrix} b \\ d \\ 0 \end{pmatrix} \]
Saddle Point Formulation

\[
\begin{pmatrix}
D & 0 & L \\
0 & R & H \\
L^T & H^T & 0
\end{pmatrix}
\begin{pmatrix}
\lambda \\
\mu \\
\delta x
\end{pmatrix}
=
\begin{pmatrix}
b \\
d \\
0
\end{pmatrix}
\]

- We call this the **saddle point** formulation of weak-constraint 4D-Var.
- The block $3 \times 3$ matrix is a saddle point matrix.
- The matrix is real, symmetric, indefinite.
- Note that the matrix contains no inverse matrices.
  - We can apply the matrix without requiring multiplication by $L^{-1}$.
- The saddle point formulation is **time parallel**.
Saddle Point Formulation

- Another way to derive the saddle point formulation is to regard the minimisation as a constrained problem:

\[
\min_{\delta p, \delta w} J(\delta p, \delta w) = (\delta p - b)^T D^{-1} (\delta p - b) + (\delta w - d)^T R^{-1} (\delta w - d)
\]

subject to \( \delta p = L \delta x \) and \( \delta w = H \delta x \).

4D-Var solves the primal problem: minimise along AXB.
4D-PSAS solves the Lagrangian dual problem: maximise along CXD.

The saddle point formulation finds the saddle point of \( L \).

The saddle point formulation is neither 4D-Var nor 4D-PSAS.
Saddle Point Formulation

- To solve the saddle point system, we have to precondition it.
- Preconditioning saddle point systems is the subject of much current research.
  
  - See e.g. Benzi and Wathen (2008), Benzi, Golub and Liesen (2005).
- One possibility (c.f. Bergamaschi, et al., 2011) is to approximate the saddle point matrix by:

\[
\tilde{P} = \begin{pmatrix}
D & 0 & \tilde{L} \\
0 & R & 0 \\
\tilde{L}^T & 0 & 0
\end{pmatrix} \quad \Rightarrow \quad \tilde{P}^{-1} = \begin{pmatrix}
0 & 0 & \tilde{L}^{-T} \\
0 & R^{-1} & 0 \\
\tilde{L}^{-1} & 0 & -\tilde{L}^{-1}D\tilde{L}^{-T}
\end{pmatrix}
\]
Saddle Point Formulation

- For $\tilde{L} = L$, we can prove some nice results:
  1. The eigenvalues $\tau$ of $\tilde{P}^{-1}A$ lie on the line $\Re(\tau) = 1$ in the complex plane.
  2. Their distance above/below the real axis is:
     \[
     \pm \sqrt{\frac{\mu_i^T H L^{-1} D L^{-T} H^T \mu_i}{\mu_i^T R \mu_i}}
     \]
     where $\mu_i$ is the $\mu$ component of the $i$th eigenvector.

- The fraction under the square root is the ratio of background+model error variance to observation error variance associated with the pattern $\mu_i$.

- This is the analogue of the eigenvalue estimate in strong constraint 4D-Var.

- For $\tilde{L} \neq L$ the conditioning appears to remain reasonable.
Results from a toy system

- The practical results shown in the next few slides are for a simplified (toy) analogue of a real system.
- The model is a two-level quasi-geostrophic channel model with 1600 gridpoints.
- The model has realistic error-growth and time-to-nonlinearity.
- There are 100 observations of streamfunction every 3 hours, and 100 wind observations every 6 hours.
- The error covariances are assumed to be horizontally isotropic and homogeneous, with a Gaussian spatial structure.
- The analysis window is 24 hours, and is divided into eight 3-hour sub-windows.
- The solution algorithm was GMRES-EN. (A poor choice. GMRES is much better — see Selime Gürol’s poster.)
Saddle Point Formulation

OOPS QG model. 24-hour window with 8 sub-windows.

Ritz Values of $\mathbf{A}$.

Converged Ritz values after 500 Arnoldi iterations are shown in blue. Unconverged values in red.
Saddle Point Formulation

OOPS QG model. 24-hour window with 8 sub-windows.

Ritz Values of $\tilde{P}^{-1}A$ for $\tilde{L} = L$.

Converged Ritz values after 500 Arnoldi iterations are shown in blue. Unconverged values in red.
Saddle Point Formulation

OOPS QG model. 24-hour window with 8 sub-windows.

Ritz Values of $\tilde{P}^{-1}A$ for $\tilde{L} = I$.

Converged Ritz values after 500 Arnoldi iterations are shown in blue. Unconverged values in red.
Saddle Point Formulation

OOPS, QG model, 24-hour window with 8 sub-windows. GMRES-EN

Convergence as a function of iteration. Solid: Forcing formulation; Dashed: saddlepoint $\tilde{L} = L$; Dotted: saddlepoint $\tilde{L} = I$. 
Saddle Point Formulation

OOPS, QG model, 24-hour window with 8 sub-windows. GMRES-EN

Convergence as a function of sequential sub-window integrations.
Conclusions

- The future viability of 4D-Var as an algorithm for Numerical Weather Prediction depends on finding, and exploiting, new dimensions of parallelism.
- The saddle point formulation of weak-constraint 4D-Var allows parallelisation in the time dimension.
- The algorithm is competitive with existing algorithms and has the potential to allow 4D-Var to remain computationally viable on next-generation computer architectures.