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

Mike Fisher

ECMWF

July 23, 2013

Outline

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.

Weak-constraint 4D-Var

- 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:

$$\begin{aligned} J(x_0 \ldots x_N) &= -\log\left(p(x_0 \ldots x_N | x_b; y_0 \ldots y_N)\right) \\ &\quad + const. \end{aligned}$$

Weak-constraint 4D-Var

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

$$\begin{aligned} J(x_0 \ldots x_N) \quad = \quad & (x_0 - x_b)^{\mathrm{T}} B^{-1} (x_0 - x_b) \\ & + \sum_{k=0}^{N} (\mathcal{H}_k(x_k) - y_k)^{\mathrm{T}} R_k^{-1} (\mathcal{H}_k(x_k) - y_k) \\ & + \sum_{k=1}^{N} (q_k - \bar{q})^{\mathrm{T}} Q_k^{-1} (q_k - \bar{q}) \,. \end{aligned}$$

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:

$$
\begin{aligned}
J(\delta x_0, \ldots, \delta x_N) \quad = \quad & \frac{1}{2} (\delta x_0 - b)^{\mathrm{T}} B^{-1} (\delta x_0 - b) \\
& + \frac{1}{2} \sum_{k=0}^{N} (H_k \delta x_k - d_k)^{\mathrm{T}} R_k^{-1} (H_k \delta x_k - d_k) \\
& + \frac{1}{2} \sum_{k=1}^{N} (\delta q_k - c_k)^{\mathrm{T}} Q_k^{-1} (\delta q_k - c_k)
\end{aligned}
$$

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:

$$
\begin{aligned}
b &= x_b - x_0 \\
c_k &= \bar{q} - q_k \\
d_k &= y_k - \mathcal{H}_k(x_k)
\end{aligned}
$$

Weak Constraint 4D-Var: Quadratic Inner Loops

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

$$\delta \mathbf{x} = \begin{pmatrix} \delta x_0 \\ \delta x_1 \\ \vdots \\ \delta x_N \end{pmatrix} \qquad \delta \mathbf{p} = \begin{pmatrix} \delta x_0 \\ \delta q_1 \\ \vdots \\ \delta q_N \end{pmatrix}$$

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:

$$\boxed{\delta \mathbf{p} = \mathbf{L} \delta \mathbf{x}}$$

where:

$$\mathbf{L} = \begin{pmatrix} I & & & & \\ -M_1 & I & & & \\ & -M_2 & I & & \\ & & \ddots & \ddots & \\ & & & -M_N & I \end{pmatrix}$$

Weak Constraint 4D-Var: Quadratic Inner Loops

We will also define:

$$
\mathbf{R} = \begin{pmatrix} R_0 & & & \\ & R_1 & & \\ & & \ddots & \\ & & & R_N \end{pmatrix}, \qquad
\mathbf{D} = \begin{pmatrix} B & & & \\ & Q_1 & & \\ & & \ddots & \\ & & & Q_N \end{pmatrix},
$$

$$
\mathbf{H} = \begin{pmatrix} H_0 & & & \\ & H_1 & & \\ & & \ddots & \\ & & & H_N \end{pmatrix}, \qquad
\mathbf{b} = \begin{pmatrix} b \\ c_1 \\ \vdots \\ c_N \end{pmatrix} \qquad
\mathbf{d} = \begin{pmatrix} d_0 \\ d_1 \\ \vdots \\ d_N \end{pmatrix}.
$$

Weak Constraint 4D-Var: Quadratic Inner Loops

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

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

Or as a function of $\delta\mathbf{p}$:

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

Weak Constraint 4D-Var: Quadratic Inner Loops

$$\mathbf{L} = \begin{pmatrix} I & & & & \\ -M_1 & I & & & \\ & -M_2 & I & & \\ & & \ddots & \ddots & \\ & & & -M_N & I \end{pmatrix}$$

$\delta\mathbf{p} = \mathbf{L}\delta\mathbf{x}$ 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 $\mathbf{L}$ is time-parallel.

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

Forcing Formulation

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

- This version of the cost function is sequential, since it contains $\mathbf{L}^{-1}$.
- 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-constrint 4D-Var.
- In particular, we can precondition it using $\mathbf{D}^{1/2}$ to diagonalise the first term:

  $$J(\chi) = \chi^{\mathrm{T}}\chi + (\mathbf{H}\mathbf{L}^{-1}\delta\mathbf{p} - \mathbf{d})^{\mathrm{T}}\mathbf{R}^{-1}(\mathbf{H}\mathbf{L}^{-1}\delta\mathbf{p} - \mathbf{d})$$

  where $\delta\mathbf{p} = \mathbf{D}^{1/2}\chi + \mathbf{b}$.

4D State Formulation

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

- This version of the cost function is parallel. It does not contain $\mathbf{L}^{-1}$.
- Unfortunately, it is difficult to precondition.

4D State Formulation

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

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

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

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

$$J(\chi) = \chi^{\mathrm{T}}\chi + (\mathbf{H}\delta\mathbf{x} - \mathbf{d})^{\mathrm{T}}\mathbf{R}^{-1}(\mathbf{H}\delta\mathbf{x} - \mathbf{d})$$

- But, we have introduced a sequential model integration (i.e. $\mathbf{L}^{-1}$) into the preconditioner.
- Replacing $\mathbf{L}^{-1}$ by something cheaper destroys the preconditioning because $\mathbf{D}$ is extremely ill-conditioned.

4D State Formulation

If we approximate $\mathbf{L}$ by $\tilde{\mathbf{L}}$ in the preconditioner, the Hessian matrix of the first term of the cost function becomes

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

Because $\mathbf{D}$ is highly ill-conditioned, this matrix is not close to the identity matrix unless $\tilde{\mathbf{L}}$ is a very good approximation of $\mathbf{L}$.

Lagrangian Dual (4D-PSAS)

A third possibility for minimising the cost function is the Lagrangian dual (known as 4D-PSAS in the meteorological community):

$$
\begin{aligned}
\delta\mathbf{x} &= \mathbf{L}^{-1}\mathbf{D}\mathbf{L}^{-\mathrm{T}}\mathbf{H}^{\mathrm{T}}\delta\mathbf{w} \\
\text{where} \quad \delta\mathbf{w} &= \arg\min_{\delta\mathbf{w}} F(\delta\mathbf{w}) \\
\text{and where} \quad F(\delta\mathbf{w}) &= \frac{1}{2}\delta\mathbf{w}^{\mathrm{T}}(\mathbf{R} + \mathbf{H}\mathbf{L}^{-1}\mathbf{D}\mathbf{L}^{-\mathrm{T}}\mathbf{H}^{\mathrm{T}})\delta\mathbf{w} + \delta\mathbf{w}^{\mathrm{T}}\mathbf{z}
\end{aligned}
$$

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

The Saddle Point Formulation

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

At the minimum:

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

Define:

$$\lambda = \mathbf{D}^{-1}(\mathbf{b} - \mathbf{L}\delta\mathbf{x}), \qquad \mu = \mathbf{R}^{-1}(\mathbf{d} - \mathbf{H}\delta\mathbf{x})$$

Then:

$$\left.\begin{array}{rcl} \mathbf{D}\lambda + \mathbf{L}\delta\mathbf{x} & = & \mathbf{b} \\ \mathbf{R}\mu + \mathbf{H}\delta\mathbf{x} & = & \mathbf{d} \\ \mathbf{L}^{\mathrm{T}}\lambda + \mathbf{H}^{\mathrm{T}}\mu & = & \mathbf{0} \end{array}\right\} \Longrightarrow \begin{pmatrix} \mathbf{D} & \mathbf{0} & \mathbf{L} \\ \mathbf{0} & \mathbf{R} & \mathbf{H} \\ \mathbf{L}^{\mathrm{T}} & \mathbf{H}^{\mathrm{T}} & \mathbf{0} \end{pmatrix} \begin{pmatrix} \lambda \\ \mu \\ \delta\mathbf{x} \end{pmatrix} = \begin{pmatrix} \mathbf{b} \\ \mathbf{d} \\ \mathbf{0} \end{pmatrix}$$

Saddle Point Formulation

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

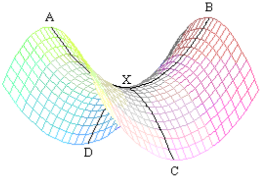
- We call this the saddle point formulation of weak-constraint 4D-Var.
- The block 3×3 matrix is a saddle point matrix.
- The matrix is real, symmetric, indefinite.
- Note that the matrix contains no inverse matrices.
  - We can apply the matrix without requiring multiplication by $\mathbf{L}^{-1}$.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

Saddle Point Formulation

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

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

$$\text{subject to } \delta\mathbf{p} = \mathbf{L}\delta\mathbf{x} \text{ and } \delta\mathbf{w} = \mathbf{H}\delta\mathbf{x}.$$



Lagrangian: $\mathcal{L}(\delta\mathbf{x}, \delta\mathbf{p}, \delta\mathbf{w}, \lambda, \mu)$

- 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 $\mathcal{L}$.
- 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:

$$\tilde{\mathcal{P}} = \begin{pmatrix} \mathbf{D} & \mathbf{0} & \tilde{\mathbf{L}} \\ \mathbf{0} & \mathbf{R} & \mathbf{0} \\ \tilde{\mathbf{L}}^{\mathrm{T}} & \mathbf{0} & \mathbf{0} \end{pmatrix} \quad \Rightarrow \quad \tilde{\mathcal{P}}^{-1} = \begin{pmatrix} \mathbf{0} & \mathbf{0} & \tilde{\mathbf{L}}^{-\mathrm{T}} \\ \mathbf{0} & \mathbf{R}^{-1} & \mathbf{0} \\ \tilde{\mathbf{L}}^{-1} & \mathbf{0} & -\tilde{\mathbf{L}}^{-1}\mathbf{D}\tilde{\mathbf{L}}^{-\mathrm{T}} \end{pmatrix}$$

Saddle Point Formulation

- For $\tilde{\mathbf{L}} = \mathbf{L}$, we can prove some nice results:
  1. The eigenvalues τ of $\tilde{\mathcal{P}}^{-1}\mathcal{A}$ lie on the line $\Re(\tau) = 1$ in the complex plane.
  2. Their distance above/below the real axis is:

$$\pm\sqrt{\frac{\mu_i^{\mathrm{T}}\mathbf{H}\mathbf{L}^{-1}\mathbf{D}\mathbf{L}^{-\mathrm{T}}\mathbf{H}^{\mathrm{T}}\mu_i}{\mu_i^{\mathrm{T}}\mathbf{R}\mu_i}}$$

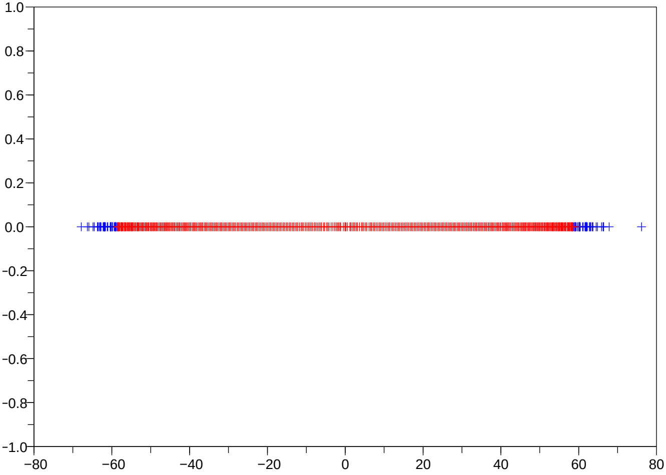
  where μ_i is the μ component of the ith eigenvector.
- The fraction under the square root is the ratio of background+model error variance to observation error variance associated with the pattern μ_i.
- This is the analogue of the eigenvalue estimate in strong constraint 4D-Var.
- For $\tilde{\mathbf{L}} \neq \mathbf{L}$ 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.

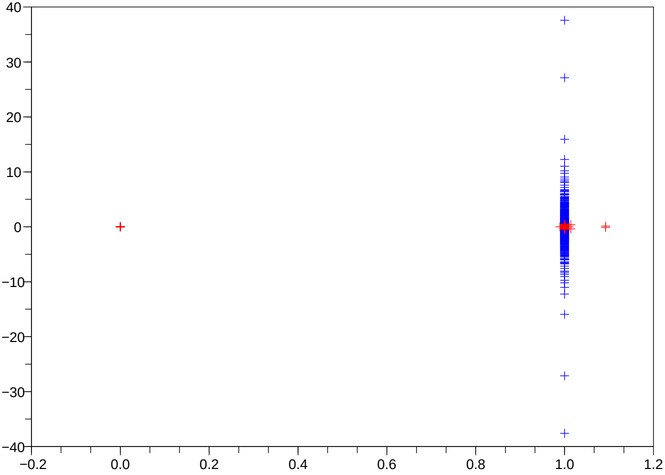


Ritz Values of $\mathcal{A}$.

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.

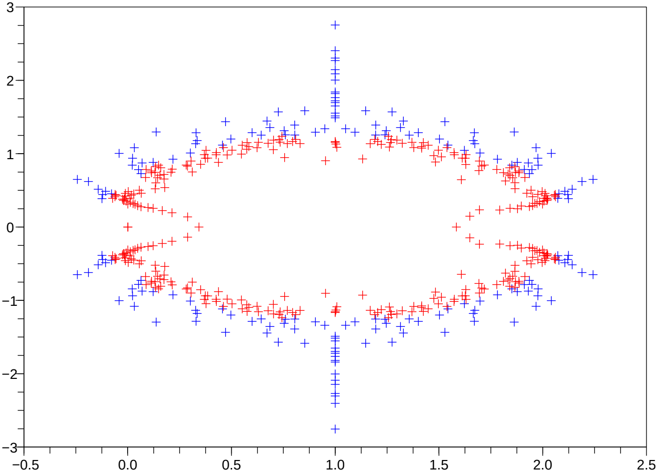


Ritz Values of $\tilde{\mathcal{P}}^{-1}\mathcal{A}$ for $\tilde{\mathbf{L}} = \mathbf{L}$.

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.

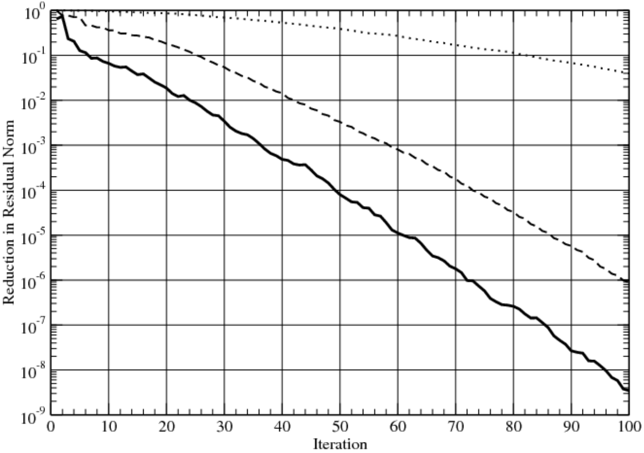


Ritz Values of $\tilde{\mathcal{P}}^{-1}\mathcal{A}$ for $\tilde{\mathbf{L}} = \mathbf{I}$.

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

Saddle Point Formulation

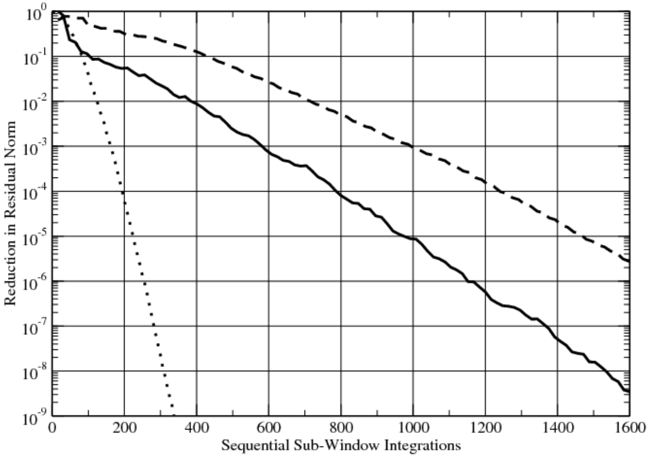
OOPS, QG model, 24-hour window with 8 sub-windows. GMRES-EN



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

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.