Long Window 4D-Var

Mike Fisher[1]

ECMWF

September 7, 2011

[1] Thanks: Harri Auvinen, Lappeenranta University of Technology, Finland

Outline

Part I: Long Window 4D-Var.

Part II: Parallel Algorithms for Weak-Constraint 4D-Var.

Part I: Long Window 4D-Var.

Persistence of Past Information

- All analysis systems combine:
  - Information from the past.
  - Current information.
- Typically, the past information takes the form of a forecast from an earlier analysis.
  - It may also include covariance information, as in the Kalman Filter.

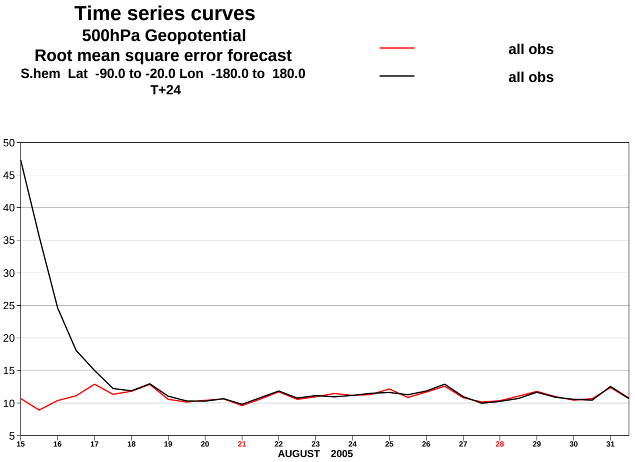
Persistence of Past Information

- All analysis systems combine:
  - Information from the past.
  - Current information.
- Typically, the past information takes the form of a forecast from an earlier analysis.
  - It may also include covariance information, as in the Kalman Filter.
- For how long does past information remain useful?

Persistence of Past Information

- All analysis systems combine:
  - Information from the past.
  - Current information.
- Typically, the past information takes the form of a forecast from an earlier analysis.
  - It may also include covariance information, as in the Kalman Filter.
- For how long does past information remain useful?
- One way to evaluate this is a data-reinstatement experiment:
  - Run the two analysis system experiments for a few weeks: one with satellite data, one without.
  - The system without satellite data will have larger analysis errors than the system that includes the data.
  - Reinstate the satellite data in the "no-satellite" experiment, and see how quickly it converges back towards the control.

Persistence of Past Information



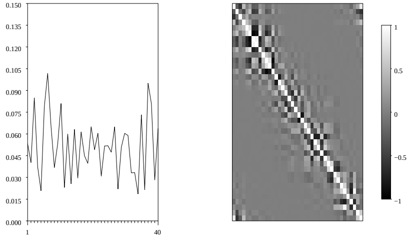
Persistence of Past Information

- What about covariance information?
- In a Kalman filter, how much influence does the covariance matrix from a few days ago have on the current covariance matrix?

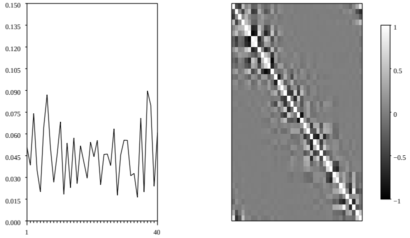
Persistence of Past Information

- What about covariance information?
- In a Kalman filter, how much influence does the covariance matrix from a few days ago have on the current covariance matrix?

Variance and correlation matrices for an EKF for the Lorenz 1996 model:



Spun up for 5 days from a static covariance matrix.



Spun up for 230 days from a static covariance matrix.

From: Fisher, Leutbecher and Kelly 2005, QJRMS.

Equivalence of 4D-Var and KF

For a linear system, 4D-Var is algebraically equivalent to the Kalman smoother:

Weak Constraint 4D-Var

$$
\begin{aligned}
J \;=\; & \frac{1}{2}(x_0 - x_b)^{\mathrm{T}} B^{-1} (x_0 - x_b) \\
& + \frac{1}{2}\sum_{k=0}^{N} (y_k - H_k x_k)^{\mathrm{T}} R_k^{-1} (y_k - H_k x_k) \\
& + \frac{1}{2}\sum_{k=1}^{N} q_k^{\mathrm{T}} Q_k^{-1} q_k
\end{aligned}
$$

where $q_k = x_k - M_k x_{k-1}$.

$\overset{\text{algebra}}{\longleftrightarrow}$

Kalman Smoother

$$
\begin{aligned}
\tilde{x}_k^a &= \tilde{x}_k^f + K_k \left(y_k - H_k \tilde{x}_k^f \right) \\
\tilde{P}_k^a &= \left[\left(\tilde{P}_k^f \right)^{-1} + H_k^{\mathrm{T}} R_k^{-1} H_k \right]^{-1} \\
\tilde{x}_{k+1}^f &= M_{k+1} \tilde{x}_k^a \\
\tilde{P}_{k+1}^f &= M_{k+1} \tilde{P}_k^a M_{k+1}^{\mathrm{T}} + Q_{k+1}
\end{aligned}
$$

$$
\begin{aligned}
x_N^a &= \tilde{x}_N^a \\
P_N^a &= \tilde{P}_N^a \\
x_k &= \tilde{x}_k^a + A_k \left(x_{k+1}^a - x_{k+1}^f \right) \\
P_k^a &= \tilde{P}_k^a + A_k \left(P_{k+1}^a - P_{k+1}^f \right) A_k^{\mathrm{T}}
\end{aligned}
$$

where

$$
\begin{aligned}
K_k &= \tilde{P}_k^a H_k^{\mathrm{T}} R_k^{-1} \\
A_k &= \tilde{P}_k^a M_{k+1} \left(\tilde{P}_{k+1}^f \right)^{-1} \\
\tilde{x}_0^f &= x_b \qquad \tilde{P}_0^f = B
\end{aligned}
$$

Equivalence of 4D-Var and KF

- 4D-Var and the Kalman Smoother produce exactly the same sequence of states $x_0, \ldots, x_N$, given the same initial state x_b and covariance matrix B.
- At the end of the analysis window (x_N), both are equivalent to the Kalman Filter.

Equivalence of 4D-Var and KF

- 4D-Var and the Kalman Smoother produce exactly the same sequence of states $x_0, \ldots, x_N$, given the same initial state x_b and covariance matrix B.
- At the end of the analysis window (x_N), both are equivalent to the Kalman Filter.
- If x_b is far enough in the past, then x_N will be insensitive to old information: i.e. to x_b and B.

Equivalence of 4D-Var and KF

- 4D-Var and the Kalman Smoother produce exactly the same sequence of states $x_0, \ldots, x_N$, given the same initial state x_b and covariance matrix B.
- At the end of the analysis window (x_N), both are equivalent to the Kalman Filter.
- If x_b is far enough in the past, then x_N will be insensitive to old information: i.e. to x_b and B.
- In this case, 4D-Var will give the same analysis x_N as a Kalman Filter that has been running indefinitely.

Equivalence of 4D-Var and KF

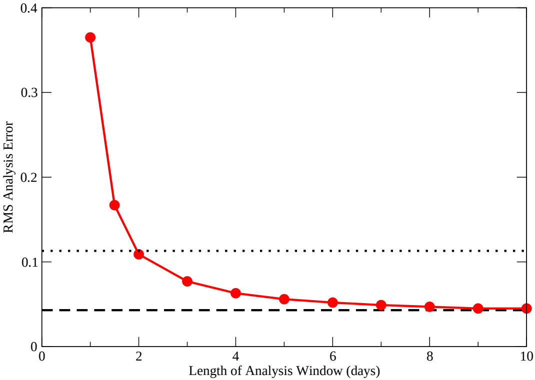
- 4D-Var and the Kalman Smoother produce exactly the same sequence of states $x_0, \ldots, x_N$, given the same initial state x_b and covariance matrix B.
- At the end of the analysis window (x_N), both are equivalent to the Kalman Filter.
- If x_b is far enough in the past, then x_N will be insensitive to old information: i.e. to x_b and B.
- In this case, 4D-Var will give the same analysis x_N as a Kalman Filter that has been running indefinitely.
- Long-Window 4D-Var is an algorithm for solving the Kalman Filter equations.

Equivalence of 4D-Var and KF

- 4D-Var and the Kalman Smoother produce exactly the same sequence of states $x_0, \ldots, x_N$, given the same initial state x_b and covariance matrix B.
- At the end of the analysis window (x_N), both are equivalent to the Kalman Filter.
- If x_b is far enough in the past, then x_N will be insensitive to old information: i.e. to x_b and B.
- In this case, 4D-Var will give the same analysis x_N as a Kalman Filter that has been running indefinitely.
- Long-Window 4D-Var is an algorithm for solving the Kalman Filter equations.
- Strictly, this equivalence holds only for a linear system. This is not fundamental to the argument. We have to decide how to linearise.
    - The quadratic inner loop of 4D-Var has an equivalent Kalman Smoother formulation. (c.f. Iterated EKF — Wishner *et al.*, 1969; Bell, 1994.)

Equivalence of 4D-Var and KF

Fisher, Leutbecher and Kelly (2005) demonstrated the equivalence of 4D-Var and the Kalman Filter for a Lorenz-1996 system:



Mean analysis error at the end of the 4D-Var window. (Mean OI and EKF analysis error are shown by dotted and dashed lines, respectively.)

Equivalence of 4D-Var and KF

Fisher, Leutbecher and Kelly's results were for a somewhat unrealistic system:

- The Lorenz-1996 system has some shortcomings as an analogue for an NWP model.
- The model was perfect.
- The analysis system did not have a B matrix.
- The system was rather well observed (60% of gridpoints observed).

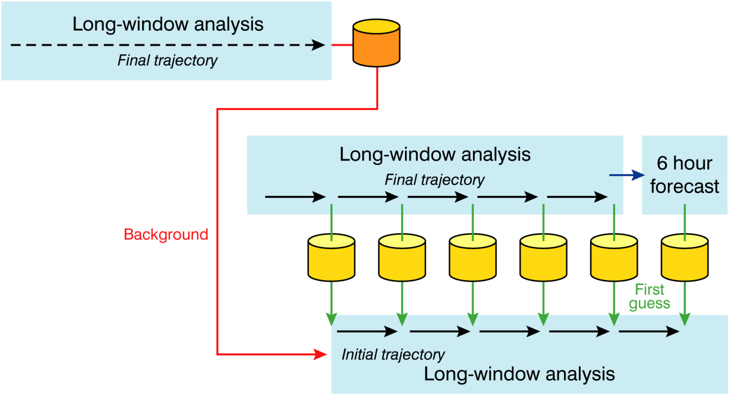
Equivalence of 4D-Var and KF

Recent work carried out by Harri Auvinen (Univ Lappeenranta, Finland) during a visit to ECMWF has addressed the shortcomings of the earlier work:

- A more realistic model: two-level quasi-geostrophic channel.
- The model has realistic error-growth and nonlinearity.
  - Error doubling time $\approx$ 30 hours.
  - Nonlinearity index (Gilmour *et al.*,2001) reaches 0.7 after $\approx$ 60 hours.
- Realistic model error, produced by perturbing model parameters (layer depths). The resulting error is:
  - flow-dependent
  - time-correlated
  - strongly anisotropic and inhomogeneous
  - contains a significant systematic component
  - poorly described by the analysis system's Q matrix
- The analysis system incudes a B matrix.
- Only 1.25% of gridpoints observed every 6 hours.

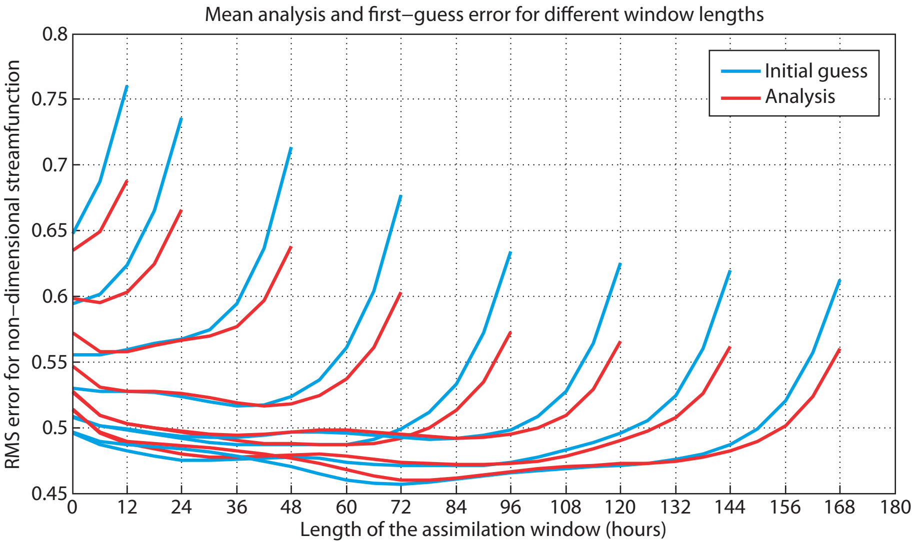
Equivalence of 4D-Var and KF

The cycling scheme is key to the success of long-window 4D-Var:



The overlap between analyses ensures that each analysis starts from a very good first-guess. Only very small adjustments to the first guess are required. The linear approximation is accurate.

Equivalence of 4D-Var and KF



Conclusions to Part I

- Analyses are insensitive to information more than a few days old.
  - Data-reinstatement experiments show that observations and backgrounds retain their usefulness for ≈ 3 days.
  - Experiments with an EKF in a simple model suggest covariance information remains useful for a similar period.
- Weak constraint 4D-Var is algebraically equivalent to the Kalman smoother.
- For sufficiently long windows, weak constraint 4D-Var analyses are identical to those of the (un-approximated) Kalman smoother.
- Long window 4D-Var can be seen as an algorithm for solving the Kalman filter equations.
- Extending the analysis window can improve the analysis even if the model error is imperfectly described.
- Time-to-nonlinearity is not a barrier on window length.

Part II: Parallel Algorithms for Weak-Constraint 4D-Var.

Can we afford (Long Window) 4D-Var?

- 4D-Var is a highly sequential algorithm:
  - Iterations of the minimisation algorithm are sequential.
  - TL and Adjoint integrations run one after the other.
  - Model timesteps follow each other.

Can we afford (Long Window) 4D-Var?

- 4D-Var is a highly sequential algorithm:
  - Iterations of the minimisation algorithm are sequential.
  - TL and Adjoint integrations run one after the other.
  - Model timesteps follow each other.
- Computers are becoming ever more parallel, but processors are not getting faster.
- Unless we do something to make 4D-Var more parallel, we will soon find that 4D-Var becomes un-affordable (even with a 12-hour window).

Can we afford (Long Window) 4D-Var?

- 4D-Var is a highly sequential algorithm:
  - Iterations of the minimisation algorithm are sequential.
  - TL and Adjoint integrations run one after the other.
  - Model timesteps follow each other.
- Computers are becoming ever more parallel, but processors are not getting faster.
- Unless we do something to make 4D-Var more parallel, we will soon find that 4D-Var becomes un-affordable (even with a 12-hour window).
- We cannot make the model more parallel.
  - The inner loops of 4D-Var run with a few 10's of grid columns per processor.
  - This is barely enough to mask inter-processor communication costs.

Can we afford (Long Window) 4D-Var?

- 4D-Var is a highly sequential algorithm:
  - Iterations of the minimisation algorithm are sequential.
  - TL and Adjoint integrations run one after the other.
  - Model timesteps follow each other.
- Computers are becoming ever more parallel, but processors are not getting faster.
- Unless we do something to make 4D-Var more parallel, we will soon find that 4D-Var becomes un-affordable (even with a 12-hour window).
- We cannot make the model more parallel.
  - The inner loops of 4D-Var run with a few 10's of grid columns per processor.
  - This is barely enough to mask inter-processor communication costs.
- We have to use more parallel algorithms.

Incremental Weak Constraint 4D-Var

- Weak Constraint 4D-Var splits the analysis window into a set of sub-windows.
- The cost function is a function of the states $x_0, x_1, \ldots, x_N$ defined at the start of each sub-window:

$$
\begin{aligned}
J(x_0, x_1, \ldots, x_N) = \quad & \frac{1}{2}(x_0 - x_b)^{\mathrm{T}} B^{-1} (x_0 - x_b) \\
& + \frac{1}{2} \sum_{k=0}^{N} (y_k - \mathcal{H}_k(x_k))^{\mathrm{T}} R_k^{-1} (y_k - \mathcal{H}_k(x_k)) \\
& + \frac{1}{2} \sum_{k=1}^{N} (q_k - \bar{q})^{\mathrm{T}} Q_k^{-1} (q_k - \bar{q})
\end{aligned}
$$

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

ECMWF

Incremental Weak Constraint 4D-Var

- We use an incremental algorithm to reduce the computational cost.

Incremental Weak Constraint 4D-Var

- We use an incremental algorithm to reduce the computational cost.
- In weak-constraint 4D-Var, the inner loop produces a four-dimensional increment, $\delta x_0, \ldots, \delta x_N$, which is used to update the four-dimensional state $x_0, x_1, \ldots, x_N$ at the outer loop:

$$x_k^{(n)} = x_k^{(n-1)} + \delta x_k^{(n-1)}$$

where (n) is the outer loop index.

Incremental Weak Constraint 4D-Var

- We use an incremental algorithm to reduce the computational cost.
- In weak-constraint 4D-Var, the inner loop produces a four-dimensional increment, $\delta x_0, \ldots, \delta x_N$, which is used to update the four-dimensional state $x_0, x_1, \ldots, x_N$ at the outer loop:

$$x_k^{(n)} = x_k^{(n-1)} + \delta x_k^{(n-1)}$$

  where (n) is the outer loop index.
- Outer-loop model integrations are required to calculate

$$\begin{aligned} d_k^{(n)} &= y_k - \mathcal{H}(x_k^{(n)}) \\ c_k^{(n)} &= \bar{q} - x_k + \mathcal{M}_k(x_{k-1}) \end{aligned}$$

- These integrations can be performed in parallel for each sub-window.

Incremental Weak Constraint 4D-Var

- We use an incremental algorithm to reduce the computational cost.
- In weak-constraint 4D-Var, the inner loop produces a four-dimensional increment, $\delta x_0, \ldots, \delta x_N$, which is used to update the four-dimensional state $x_0, x_1, \ldots, x_N$ at the outer loop:

$$x_k^{(n)} = x_k^{(n-1)} + \delta x_k^{(n-1)}$$

where (n) is the outer loop index.

- Outer-loop model integrations are required to calculate

$$\begin{aligned} d_k^{(n)} &= y_k - \mathcal{H}(x_k^{(n)}) \\ c_k^{(n)} &= \bar{q} - x_k + \mathcal{M}_k(x_{k-1}) \end{aligned}$$

- These integrations can be performed in parallel for each sub-window.
- Parallelising the outer loop is (in principle) easy.

Incremental Weak Constraint 4D-Var

The inner loop minimises:

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

$\delta q_k = \delta x_k - M_k^{(n)}\delta x_{k-1}$,

and where $b^{(n)}$, $c_k^{(n)}$ and $d_k^{(n)}$ come from the outer loop:

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

Weak Constraint 4D-Var: Inner Loop

- Parallelising the inner loop is not trivial!
- We have two options:
  - Parallelise the minimiser (compute multiple cost-function gradients at each iteration, in the hope that this will reduce the number of iterations required).
  - Parallelise the computations within a gradient calculation.
- In my view, it is unlikely that parallel minimisation will help us much.
- We have to parallelise the computations within each iteration.

Parallelising within an Iteration

- The model is already parallel in both horizontal directions.
- The modellers tell us that it will be hard to parallelise in the vertical (and we already have too little work per processor).
- We are left with parallelising in the time direction.
- Weak-constraint 4D-Var offers some interesting possibilities for parallelisation in the time direction.
  - We managed to parallelise over sub-windows at the outer loop of incremental 4D-Var.
- Can we do the same for the inner loop?

Weak Constraint 4D-Var: Inner Loop

Dropping the outer loop index (n), the inner loop of weak-constraints 4D-Var minimises:

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

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: Inner Loop

We can simplify this further 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: Inner Loop

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

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

$\delta\mathbf{x} = \mathbf{L}^{-1}\delta\mathbf{p}$ 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.

Weak Constraint 4D-Var: Inner Loop

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: Inner Loop

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 $\delta\mathbf{p}$:

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

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.
  - It contains $\mathbf{L}^{-1}$.
- It closely resembles 3D-Var and strong-constraint 4D-Var.
- In particular, we can precondition it using $\mathbf{D}^{1/2}$:

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

  where $\delta\mathbf{p} = \mathbf{D}^{1/2}\chi + \mathbf{b}$.
- We understand how to minimise this.

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 $\mathbf{L}^{-1}$.
- We could precondition it using $\delta\mathbf{x} = \mathbf{L}^{-1}(\mathbf{D}^{1/2}\chi + \mathbf{b})$.
- This would give exactly the same $J(\chi)$ as before.
- But, we have introduced a sequential model integration (i.e. $\mathbf{L}^{-1}$) into the preconditioner.

Plan A: State Formulation, Approximate Preconditioner

- In the forcing ($\delta\mathbf{p}$) formulation (and in 4D-PSAS) $\mathbf{L}^{-1}$ appears in the cost function.
  - These formulations are inherently sequential.
  - We cannot modify the cost function without changing the problem.
- In the 4D-state ($\delta\mathbf{x}$) formulation, $\mathbf{L}^{-1}$ appears in the preconditioner.
  - We are free to modify the preconditioner as we wish.
- This suggests we replace $\mathbf{L}^{-1}$ by a cheap approximation:

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

- If we do this, we can no longer write $J_b + J_q = \chi^{\mathrm{T}}\chi$.
- We have to calculate $\delta\mathbf{x}$, and explicity evaluate

$$J_b + J_q = (\mathbf{L}\delta\mathbf{x} - \mathbf{b})^{\mathrm{T}}\mathbf{D}^{-1}(\mathbf{L}\delta\mathbf{x} - \mathbf{b})$$

- This is where we run into problems...

Plan A: State Formulation, Approximate Preconditioner

- When we approximate $\mathbf{L}^{-1}$ in the preconditioner, the Hessian of $J_b + J_q$ (with respect to χ) is no longer the identity matrix, but:

$$(J_b + J_q)'' = \mathbf{D}^{\mathrm{T}/2}\tilde{\mathbf{L}}^{-\mathrm{T}}\mathbf{L}^{\mathrm{T}}\mathbf{D}^{-1}\mathbf{L}\tilde{\mathbf{L}}^{-1}\mathbf{D}^{1/2}$$

- Unfortunately, the matrix $\mathbf{D}^{-1}$, has some enormous eigenvalues.
  - Large spatial scales have near-zero variances.
- This makes the preconditioning extremely sensitive to the accuracy with which $\tilde{\mathbf{L}}$ approximates $\mathbf{L}$.
- I have tried a number of different approximations $\mathbf{L}$. They all gave condition numbers for the minimisation of O(10^9), and the minimisation failed to converge.
- It seems we need to avoid algorithms that rely on a cancellation between $\mathbf{D}$ and $\mathbf{D}^{-1}$.

Plan A: State Formulation, Approximate Preconditioner

- When we approximate $\mathbf{L}^{-1}$ in the preconditioner, the Hessian of $J_b + J_q$ (with respect to χ) is no longer the identity matrix, but:

$$(J_b + J_q)'' = \mathbf{D}^{\mathrm{T}/2}\tilde{\mathbf{L}}^{-\mathrm{T}}\mathbf{L}^{\mathrm{T}}\mathbf{D}^{-1}\mathbf{L}\tilde{\mathbf{L}}^{-1}\mathbf{D}^{1/2}$$

- Unfortunately, the matrix $\mathbf{D}^{-1}$, has some enormous eigenvalues.
  - Large spatial scales have near-zero variances.
- This makes the preconditioning extremely sensitive to the accuracy with which $\tilde{\mathbf{L}}$ approximates $\mathbf{L}$.
- I have tried a number of different approximations $\mathbf{L}$. They all gave condition numbers for the minimisation of O(10^9), and the minimisation failed to converge.
- It seems we need to avoid algorithms that rely on a cancellation between $\mathbf{D}$ and $\mathbf{D}^{-1}$.
- We need a Plan B!

Plan B: 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:

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

Plan B: 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:

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

Plan B: 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

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

- This is called the saddle point formulation of 4D-Var.
- The 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 a sequential model integration (i.e. we can parallelise over sub-windows).
- We can hope that the problem is well conditioned (since we don't multiply by $\mathbf{D}^{-1}$).

Saddle Point Formulation

Alternative derivation:

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

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

Saddle Point Formulation

Alternative derivation:

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

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

$$\begin{aligned}\mathcal{L}(\delta\mathbf{x},\delta\mathbf{p},\delta\mathbf{w},\lambda,\mu) = \; & (\delta\mathbf{p}-\mathbf{b})^{\mathrm{T}}\mathbf{D}^{-1}(\delta\mathbf{p}-\mathbf{b}) + (\delta\mathbf{w}-\mathbf{d})^{\mathrm{T}}\mathbf{R}^{-1}(\delta\mathbf{w}-\mathbf{d}) \\ & +\lambda^{\mathrm{T}}(\delta\mathbf{p}-\mathbf{L}\delta\mathbf{x}) + \mu^{\mathrm{T}}(\delta\mathbf{w}-\mathbf{H}\delta\mathbf{x})\end{aligned}$$

Saddle Point Formulation

Alternative derivation:

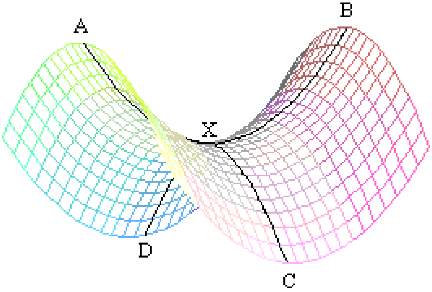
$$\min_{\delta\mathbf{p},\delta\mathbf{w}} J(\delta\mathbf{p},\delta\mathbf{w}) = (\delta\mathbf{p}-\mathbf{b})^{\mathrm{T}}\mathbf{D}^{-1}(\delta\mathbf{p}-\mathbf{b}) + (\delta\mathbf{w}-\mathbf{d})^{\mathrm{T}}\mathbf{R}^{-1}(\delta\mathbf{w}-\mathbf{d})$$

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

$$\begin{aligned}\mathcal{L}(\delta\mathbf{x},\delta\mathbf{p},\delta\mathbf{w},\lambda,\mu) = \; & (\delta\mathbf{p}-\mathbf{b})^{\mathrm{T}}\mathbf{D}^{-1}(\delta\mathbf{p}-\mathbf{b}) + (\delta\mathbf{w}-\mathbf{d})^{\mathrm{T}}\mathbf{R}^{-1}(\delta\mathbf{w}-\mathbf{d}) \\ & + \lambda^{\mathrm{T}}(\delta\mathbf{p}-\mathbf{L}\delta\mathbf{x}) + \mu^{\mathrm{T}}(\delta\mathbf{w}-\mathbf{H}\delta\mathbf{x})\end{aligned}$$

- $\frac{\partial\mathcal{L}}{\partial\lambda} = \mathbf{0} \Rightarrow \quad \delta\mathbf{p} = \mathbf{L}\delta\mathbf{x}$
- $\frac{\partial\mathcal{L}}{\partial\mu} = \mathbf{0} \Rightarrow \quad \delta\mathbf{w} = \mathbf{H}\delta\mathbf{x}$
- $\frac{\partial\mathcal{L}}{\partial\delta\mathbf{p}} = \mathbf{0} \Rightarrow \quad \mathbf{D}^{-1}(\delta\mathbf{p}-\mathbf{b}) + \lambda = \mathbf{0}$
- $\frac{\partial\mathcal{L}}{\partial\delta\mathbf{w}} = \mathbf{0} \Rightarrow \quad \mathbf{R}^{-1}(\delta\mathbf{w}-\mathbf{d}) + \mu = \mathbf{0}$
- $\frac{\partial\mathcal{L}}{\partial\delta\mathbf{x}} = \mathbf{0} \Rightarrow \quad \mathbf{L}^{\mathrm{T}}\lambda + \mathbf{H}^{\mathrm{T}}\mu = \mathbf{0}$

Saddle Point Formulation



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. It is something of a black art!
  - See e.g. Benzi and Wathen (2008), Benzi, Golub and Liesen (2005).
- One possibility is (c.f. Bergamaschi, *et al.*, 2011):

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

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

- Note that $\tilde{\mathcal{P}}^{-1}$ does not contain $\mathbf{D}^{-1}$.

Saddle Point Formulation

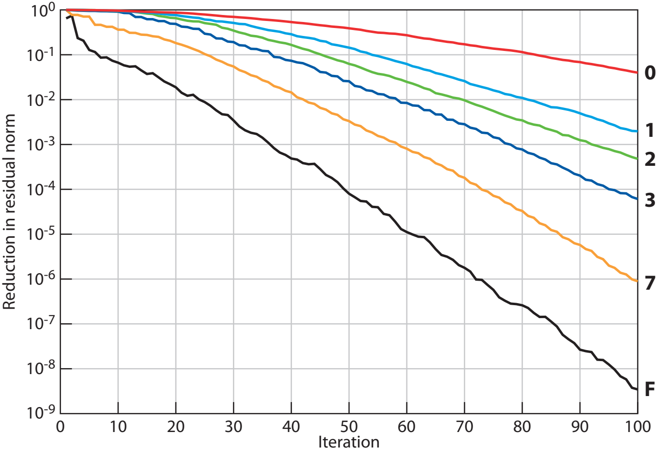
- We still need an approximate inverse of $\mathbf{L}$.
- One approach is to use the following identity (exercise for the reader!):

$$\boxed{\mathbf{L}^{-1} = \mathbf{I} + (\mathbf{I} - \mathbf{L}) + (\mathbf{I} - \mathbf{L})^2 + \ldots + (\mathbf{I} - \mathbf{L})^{N-1}}$$

- Since this is a power series expansion, it suggests truncating the series at some order $< N - 1$.

Saddle Point Formulation

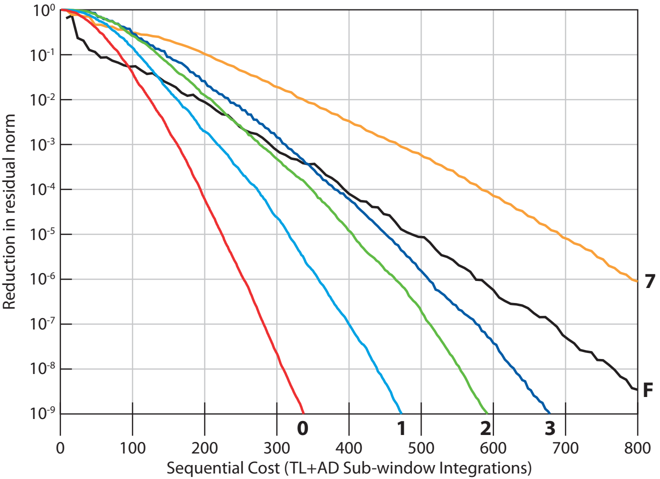
OOPS, QG model, 24-hour window with 8 sub-windows.



Convergence as a function of iteration for different truncations of the series expansion for **L**. ("F" = Forcing formulation.)

Saddle Point Formulation

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

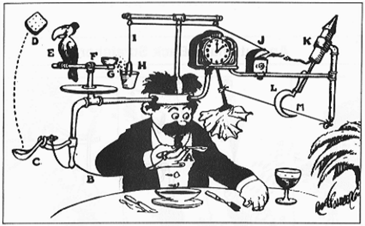


Convergence as a function of sequential sub-window integrations for different truncations of the series expansion for **L**.

Conclusions to Part II

- 4D-Var is not dead yet.
  - Beware (parallel) doom-mongers.
  - c.f. the long-predicted death of spectral models.
- In principle, the 4D-state and saddle point formulations allow parallelisation over sub-windows.
- 4D-PSAS and the forcing formulation are inherently sequential.
- Ill-conditioning of $\mathbf{D}^{-1}$ is a problem for the 4D-state formulation.
- The saddle point formulation is already fast enough to be useful.
  - Better preconditioners may make it even faster.
- Experiments with the QG model were conducted using the Object-Oriented Prediction System (OOPS).
  - OOPS lived up to its billing as an easy to use, flexible framework for work on data assimilation algorithms.

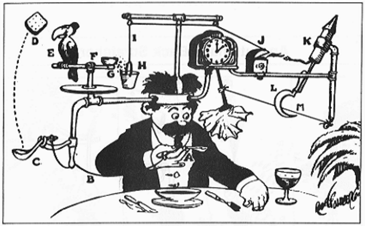
The ECMWF Assimilation System in 2020?



Hybrid Ensemble 4D Particle Ensemble Weak-Constraint

Saddle-Point Long-Window KF Var

The ECMWF Assimilation System in 2020?



Hybrid Ensemble 4D Particle Ensemble Weak-Constraint

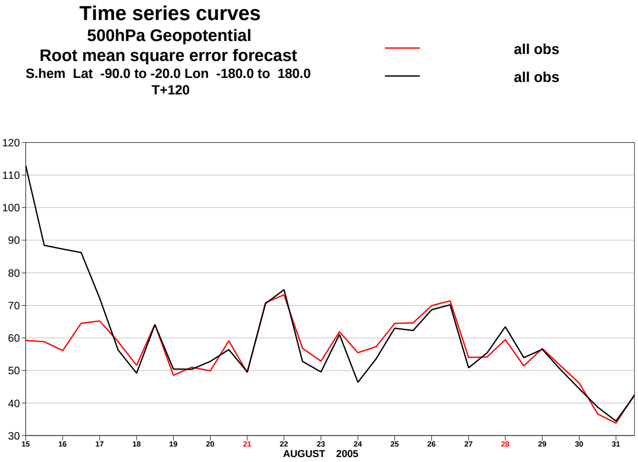
Saddle-Point Long-Window KF Var

What We Really Want

?

Backup Slides

Persistence of Past Information



Parallel Minimisation

- The minimisation algorithms used in the inner loop of 4D-Var are based on Krylov methods: conjugate gradients, quasi-Newton.

Parallel Minimisation

- The minimisation algorithms used in the inner loop of 4D-Var are based on Krylov methods: conjugate gradients, quasi-Newton.
- A Krylov method solves a linear equation $Ax = b$ in the sub-space generated by b:

$$\{b, Ab, A^2 b, \ldots, A^K b\}$$

Parallel Minimisation

- The minimisation algorithms used in the inner loop of 4D-Var are based on Krylov methods: conjugate gradients, quasi-Newton.
- A Krylov method solves a linear equation $Ax = b$ in the sub-space generated by b:

$$\{b, Ab, A^2b, \ldots, A^K b\}$$

- The reason for this is that the inverse of A can be expressed as a polynomial in A (Cayley-Hamilton theorem):

$$A^{-1} = \alpha_0 I + \alpha_1 A + \alpha_2 A^2 + \ldots \alpha_K A^K$$

Parallel Minimisation

- The minimisation algorithms used in the inner loop of 4D-Var are based on Krylov methods: conjugate gradients, quasi-Newton.
- A Krylov method solves a linear equation $Ax = b$ in the sub-space generated by b:

$$\{b, Ab, A^2 b, \ldots, A^K b\}$$

- The reason for this is that the inverse of A can be expressed as a polynomial in A (Cayley-Hamilton theorem):

$$\begin{aligned} A^{-1} &= \alpha_0 I + \alpha_1 A + \alpha_2 A^2 + \ldots \alpha_K A^K \\ \Rightarrow \quad x = A^{-1} b &= \alpha_0 b + \alpha_1 Ab + \alpha_2 A^2 b + \ldots \alpha_K A^K b \end{aligned}$$

Parallel Minimisation

- The minimisation algorithms used in the inner loop of 4D-Var are based on Krylov methods: conjugate gradients, quasi-Newton.
- A Krylov method solves a linear equation $Ax = b$ in the sub-space generated by b:

$$\{b, Ab, A^2 b, \ldots, A^K b\}$$

- The reason for this is that the inverse of A can be expressed as a polynomial in A (Cayley-Hamilton theorem):

$$\begin{aligned} A^{-1} &= \alpha_0 I + \alpha_1 A + \alpha_2 A^2 + \ldots \alpha_K A^K \\ \Rightarrow \quad x = A^{-1} b &= \alpha_0 b + \alpha_1 Ab + \alpha_2 A^2 b + \ldots \alpha_K A^K b \end{aligned}$$

- The right-hand side, b, is very important to the success of the method. A sub-space generated by a different vector c will not contain a good approximation to the solution of $Ax = b$.

Parallel Minimisation

- In our case, the minimisation solves the equation $\nabla J = 0$ using Newton's method:

$$J'' \delta x = -\nabla J|_{\delta x = 0}$$

- That is: $A \longrightarrow J''$ and $b \longrightarrow -\nabla J|_{\delta x=0}$
- To minimise the cost function, we generate the Krylov space from the initial gradient $\nabla J|_{\delta x=0}$ by repeated sequential applications of J''.

Parallel Minimisation

- In our case, the minimisation solves the equation $\nabla J = 0$ using Newton's method:

$$J''\delta x = -\nabla J|_{\delta x=0}$$

- That is: $A \longrightarrow J''$ and $b \longrightarrow -\nabla J|_{\delta x=0}$
- To minimise the cost function, we generate the Krylov space from the initial gradient $\nabla J|_{\delta x=0}$ by repeated sequential applications of J''.
- Generating gradients in parallel, from other starting vectors, produces Krylov spaces that are not relevant to the problem we are trying to solve.
- Computing gradients in parallel does not significantly reduce the number of iterations required to minimise the cost function.

4D-PSAS

With our notation, 4D-PSAS is:

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

$F(\delta\mathbf{w})$ contains $\mathbf{L}^{-1}$, so 4D-PSAS is a sequential algorithm.

Saddle Point Formulation

- There is a large and growing literature on the numerical solution of saddle point systems.
- A good review paper:

  Benzi M, Golub G H, and Liesen J, 2005: Numerical Solution of Saddle Point Systems, *Acta Numerica*, 1–137

  This paper has 29 pages of references.
- See also:

  Benzi M and Wathen A J, 2008: Some Preconditioning Techniques for Saddle Point Problems, *in W. Schilders, H. A. van der Vorst and J. Rommes, eds., Model Order Reduction: Theory, Research Aspects and Applications, Springer-Verlag (Series: Mathematics in Industry)*, 195–211.

Both papers are easy to find online — or ask me for a copy.

Saddle Point Formulation

A very wide range of problems can be cast in saddle point form.
Benzi Golub and Liesen (2005) give the following list:

- computational fluid dynamics (Glowinski 1984, Quarteroni and Valli 1994, Temam 1984, Turek 1999, Wesseling 2001)
- constrained and weighted least squares estimation (Bjorck 1996, Golub and Van Loan 1996)
- constrained optimisation (Gill, Murray and Wright 1981, Wright 1992, Wright 1997)
- economics (Arrow, Hurwicz and Uzawa 1958, Duchin and Szyld 1979, Leontief, Duchin and Szyld 1985, Szyld 1981)
- electrical circuits and networks (Bergen 1986, Chua, Desoer and Kuh 1987, Strang 1986, Tropper 1962)
- electromagnetism (Bossavit 1998, Perugia 1997, Perugia, Simoncini and Arioli 1999)
- finance (Markowitz 1959, Markowitz and Perold 1981)
- image reconstruction (Hall 1979)
- image registration (Haber and Modersitzki 2004, Modersitzki 2003)
- interpolation of scattered data (Lyche, Nilssen and Winther 2002, Sibson and Stone 1991)
- linear elasticity (Braess 2001, Ciarlet 1988)
- mesh generation for computer graphics (Liesen, de Sturler, Sheffer, Aydin and Siefert 2001)
- mixed finite element approximations of elliptic PDEs (Brezzi 1974, Brezzi and Fortin 1991, Quarteroni and Valli 1994)
- model order reduction for dynamical systems (Freund 2003, Heres and Schilders 2005, Stykel 2005)
- optimal control (Battermann and Heinkenschloss 1998, Battermann and Sachs 2001, Betts 2001, Biros and Ghattas 2000, Nguyen 2004)
- parameter identification problems (Burger and Muhlhuber 2002, Haber and Ascher 2001, Haber, Ascher and Oldenburg 2000).

Saddle Point Formulation

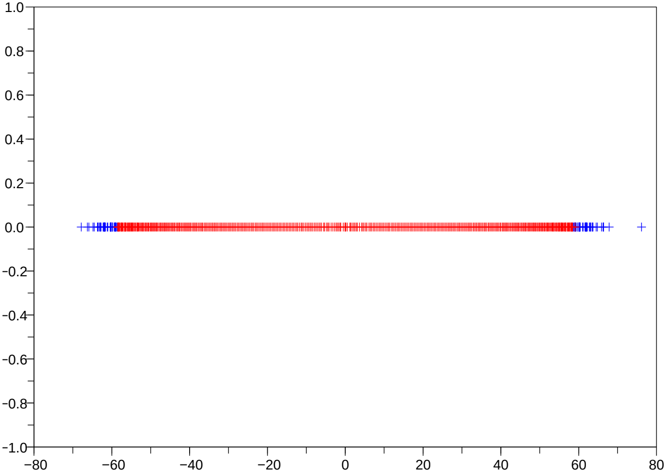
- With this preconditioner, we can prove some nice results for the case $\tilde{\mathbf{L}} = \mathbf{L}$
  1. The eigenvalues τ of $\tilde{\mathcal{P}}^{-1}\mathcal{A}$ lie on the line $\Re(\tau) = 1$ in the complex plane.
  2. Their distance above/below the real axis is:

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

  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.

Saddle Point Formulation

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

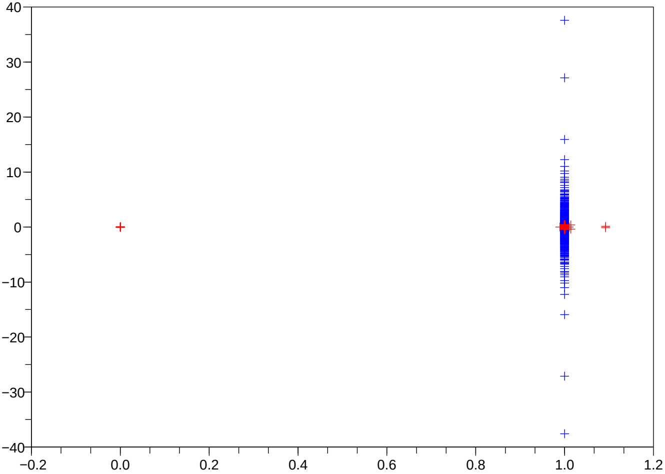


Ritz Values of $\mathcal{A}$.

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.

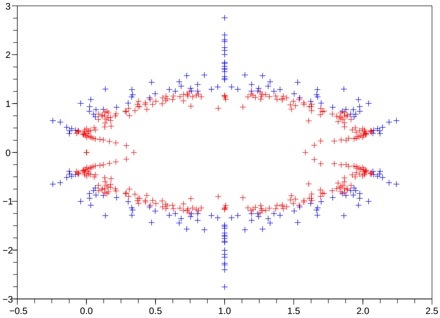


Ritz Values of $\tilde{\mathcal{P}}^{-1}\mathcal{A}$ for $\tilde{\mathbf{L}} = \mathbf{L}$.

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

Saddle Point Formulation

- It is much harder to prove results for the case $\tilde{\mathbf{L}} \neq \mathbf{L}$.
- Experimentally, it seems that many eigenvalues continue to lie on $\Re(\tau) = 1$, with the remainder forming a cloud around $\tau = 1$.



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.