Numerical solution for a time-parallelized formulation of 4DVAR

M. Fisher\textsuperscript{1}, S. Gratton\textsuperscript{2,3}, S. Gürol \textsuperscript{3}

\textsuperscript{1}ECMWF, Reading, UK
\textsuperscript{2}ENSEEIHT, Toulouse, France
\textsuperscript{3}CERFACS, Toulouse, France

Workshop on Meteorological Sensitivity Analysis and Data Assimilation

Roanoke, West Virginia

2 June 2015
Outline

- Saddle point approach of 4D-Var
- Preconditioning of saddle point formulation
- Numerical results
- Conclusions
Why saddle-point formulation?

- 4D-Var is a sequential algorithm.
  → Tangent Linear and Adjoint integrations run one after the other.
  → Model timesteps follow each other.

- Parallelization of 4D-Var in the spatial domain has been performed by a spatial decomposition, and distribution over processors of the model grid.
  → The number of grid points (associated with each processor) are independent of the resolution of the model.

- BUT, increasing the resolution of the model, increases the work per processor since higher resolutions require shorter timesteps.

- In order to keep the work per processor constant, parallelization in the time dimension is required.

M. Fisher shows that saddle-point formulation allows parallelization in the time dimension.
Weak-constraint 4D-Var

\[
\min_{x \in \mathbb{R}^n} \frac{1}{2} \|x_0 - x_b\|_B^{-1}^2 + \frac{1}{2} \sum_{j=0}^{N} \|\mathcal{H}_j(x_j) - y_j\|_{R_j}^{-1}^2 + \frac{1}{2} \sum_{j=1}^{N} \|x_j - M_j(x_{j-1})\|_{Q_j}^{-1}^2
\]

- \( x = \begin{pmatrix} x_0 \\ x_1 \\ \vdots \\ x_N \end{pmatrix} \in \mathbb{R}^n \) is the control variable where \( x_j = x(t_j) \) defined at the start of each of a set of sub-windows that span the analysis window.

- \( x_b \) is the background given at the initial time \( (t_0) \).

- \( y_j \in \mathbb{R}^{m_j} \) is the observation vector over a given time interval

- \( \mathcal{H}_j \) maps the state vector \( x_j \) from model space to observation space

- \( M_j \) represents an integration of the numerical model from time \( t_{j-1} \) to \( t_j \)

- \( B, R_j \) and \( Q_j \) are the covariance matrices of background, observation and model error.
Formulation

Let us consider the linearized subproblem of the weak-constraint 4D-Var as a constrained problem and write its Lagrangian function. Then the stationary point of $\mathcal{L}$ satisfies the system of equations that can be written in a matrix form as:

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

This system is called the saddle-point formulation of 4D-Var.

$L = \begin{pmatrix}
I \\
-M_1 & I \\
& -M_2 & I \\
& & \ddots & \ddots \\
& & & -M_N & I \\
\end{pmatrix}$ is an n-by-n matrix.

$H = \text{diag}(H_0, H_1, \ldots, H_N)$ is an n-by-m matrix.

$D = \text{diag}(B, Q_1, \ldots, Q_N)$ is an n-by-n matrix.

$R = \text{diag}(R_0, R_1, \ldots, R_N)$ is an m-by-m matrix.
Parallelization in the time dimension

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

- We can apply the matrix \(A\) without requiring a sequential model integration (i.e. we can parallelise over sub-windows).

\[
L \delta x = \begin{pmatrix}
I & & \\
-M_1 & I & \\
& -M_2 & I \\
& & \ddots & \ddots \\
& & & -M_N & I
\end{pmatrix}
\begin{pmatrix}
\delta x_0 \\
\delta x_1 \\
\delta x_2 \\
\vdots \\
\delta x_N
\end{pmatrix}
= \begin{pmatrix}
\delta x_0 \\
\delta x_1 - M_1 \delta x_0 \\
\delta x_2 - M_2 \delta x_1 \\
\vdots \\
\delta x_N - M_N \delta x_{N-1}
\end{pmatrix}
\]

→ Matrix-vector products with \(L\) can be parallelized in the time dimension

- Note that the matrix contains no inverse matrices.
Properties of the saddle point system

\[ A = \begin{pmatrix}
D & 0 & L \\
0 & R & H \\
L^T & H^T & 0
\end{pmatrix} = \begin{pmatrix}
A & B \\
B^T & 0
\end{pmatrix} \]

- \( A \) is a \((2n + m)\)-by-\((2n + m)\) indefinite symmetric matrix. \( A \) has negative and positive eigenvalues.

- The solution of this problem is a saddle point.

- \( A \) is symmetric positive definite, i.e. \( x^T A x > 0 \)

- If the schur complement \( S = -BA^{-1}B^T \) is negative definite, then \( A \) is invertible and saddle point system has a unique solution.
Properties of the saddle point system

- 4D-Var solves the primal problem: minimise along AXB.
- Dual algorithms (PSAS, RPCG) solves the Lagrangian dual problem: maximise along CXD.
- The saddle point formulation finds the saddle point of the Lagrangian problem

ref: Mike’s presentation
Numerical solution of the saddle point system

- MINRES or GMRES Krylov subspace methods can be used to solve iteratively the symmetric indefinite saddle point system.
Numerical solution of the saddle point system

- MINRES or GMRES Krylov subspace methods can be used to solve iteratively the symmetric indefinite saddle point system.

- When using iterative methods, it is crucial to find an efficient preconditioner which attempts to improve the spectral properties of the system.

**Efficient preconditioner \( P \)

- is an approximation to \( A \)

- the cost of constructing and applying the preconditioner should be less than the gain in computational cost

- exploits the block structure of the problem for saddle point systems
Numerical solution of the saddle point system

- MINRES or GMRES Krylov subspace methods can be used to solve iteratively the symmetric indefinite saddle point system.

- When using iterative methods, it is crucial to find an efficient preconditioner which attempts to improve the spectral properties of the system.

  Efficient preconditioner $P$
  - is an approximation to $A$
  - the cost of constructing and applying the preconditioner should be less than the gain in computational cost
  - exploits the block structure of the problem for saddle point systems

- We focus on GMRES since it allows us to use more general preconditioners.
How to precondition?

\[ A = \begin{pmatrix} D & 0 & L \\ 0 & R & H \\ L^T & H^T & 0 \end{pmatrix} = \begin{pmatrix} A & B \\ B^T & 0 \end{pmatrix} \]

- Preconditioning saddle point systems is the subject of much current research!

  ⇒ Nice review is given by Benzi, Golub and Liesen (2005).

- Most preconditioners in the literature assume that \( D \) and \( R \) are expensive, and \( L \) and \( H \) are cheap.
How to precondition?

\[ A = \begin{pmatrix} D & 0 & L \\ 0 & R & H \\ L^T & H^T & 0 \end{pmatrix} = \begin{pmatrix} A & B \\ B^T & 0 \end{pmatrix} \]

- Preconditioning saddle point systems is the subject of much current research!

⇒ Nice review is given by Benzi, Golub and Liesen (2005).

- Most preconditioners in the literature assume that \( D \) and \( R \) are expensive, and \( L \) and \( H \) are cheap.

- The opposite is true in our case! \( B \) is the most computationally expensive block and calculations involving \( A \) are relatively cheap.
How to precondition?

\[ \mathcal{A} = \begin{pmatrix} D & 0 & L \\ 0 & R & H \\ L^T & H^T & 0 \end{pmatrix} = \begin{pmatrix} A & B^T \\ \tilde{B} & 0 \end{pmatrix} \]

- The inexact constraint preconditioner proposed by (Bergamaschi et. al. 2005) is promising for our application. The preconditioner can be chosen as:

\[ \mathcal{P} = \begin{pmatrix} A & \tilde{B}^T \\ \tilde{B} & 0 \end{pmatrix} = \begin{pmatrix} D & 0 & \tilde{L} \\ 0 & R & 0 \\ \tilde{L}^T & 0 & 0 \end{pmatrix} \Rightarrow \mathcal{P}^{-1} = \begin{pmatrix} 0 & 0 & \tilde{L}^{-T} \\ 0 & R^{-1} & 0 \\ \tilde{L}^{-1} & 0 & -\tilde{L}^{-1}DL\tilde{L}^{-T} \end{pmatrix} \]

where
- \( \tilde{L} \) is an approximation to the matrix \( L \)
- \( \tilde{B} = [\tilde{L}^T \ 0] \) is a full row rank approximation of the matrix \( B \in \mathbb{R}^{n \times (m+n)} \)
When solving a sequence of saddle point systems, can we further improve the preconditioning for the outer loops $k > 1$?

Can we find low-rank updates for the inexact constraint preconditioner that approximates $A^{-1}$ or its effect on a vector?
Preconditioning Saddle Point Formulation of 4D-Var

For $k = 1$, we have the inexact constraint preconditioner:

$$\mathcal{P}_0 = \begin{pmatrix} A & B_0^T \\ B_0 & 0 \end{pmatrix} \Rightarrow \mathcal{P}_0^{-1} A_1 u = \mathcal{P}_0^{-1} f_1$$
Preconditioning Saddle Point Formulation of 4D-Var

For $k = 1$, we have the inexact constraint preconditioner:

$$P_0 = \begin{pmatrix} A & B^T_0 \\ B_0 & 0 \end{pmatrix} \Rightarrow P_0^{-1} A_1 u = P_0^{-1} f_1$$

For $k > 1$, we want to find a low-rank update $\Delta B_k = B_{k+1} - B_k$ and use the updated preconditioner:

$$P_k = \begin{pmatrix} A & B_k^T \\ B_k & 0 \end{pmatrix} + \begin{pmatrix} 0 & \Delta B_k^T \\ \Delta B_k & 0 \end{pmatrix} \Rightarrow P_k^{-1} A_{k+1} u = P_k^{-1} f_{k+1}$$
Preconditioning Saddle Point Formulation of 4D-Var

- For $k = 1$, we have the inexact constraint preconditioner:

$$
\mathcal{P}_0 = \begin{pmatrix}
A & B_0^T \\
B_0 & 0
\end{pmatrix} \Rightarrow \mathcal{P}_0^{-1} A_1 u = \mathcal{P}_0^{-1} f_1
$$

- For $k > 1$, we want to find a low-rank update $\Delta B_k = B_{k+1} - B_k$ and use the updated preconditioner:

$$
\mathcal{P}_k = \begin{pmatrix}
A & B_k^T \\
B_k & 0
\end{pmatrix} + \begin{pmatrix}
0 & \Delta B_k^T \\
\Delta B_k & 0
\end{pmatrix} \Rightarrow \mathcal{P}_k^{-1} A_{k+1} u = \mathcal{P}_k^{-1} f_{k+1}
$$

How to obtain these updates?

- GMRES performs matrix-vector products with $A$:

$$
\begin{pmatrix}
A & B_k^T \\
B & 0
\end{pmatrix}
\begin{pmatrix}
\delta x_j \\
u_j^{(k)}
\end{pmatrix}
= 
\begin{pmatrix}
b_j \\
c_j
\end{pmatrix}
\Rightarrow
\begin{pmatrix}
A_k \\
u_j^{(k)}
\end{pmatrix}
= 
\begin{pmatrix}
\delta x_j \\
f_j^{(k)}
\end{pmatrix}
$$

- We can use the pairs $(u_j^{(k)}, f_j^{(k)})$ to find an update $\Delta B_k$. 
Preconditioning Saddle Point Formulation of 4D-Var

\[
\begin{pmatrix}
A & B^T \\
B & 0 \\
\end{pmatrix}
\begin{pmatrix}
v \\
\delta x \\
\end{pmatrix}
= 
\begin{pmatrix}
b \\
c \\
\end{pmatrix}
\Rightarrow
\begin{pmatrix}
A & B_k^T \\
B_k & 0 \\
\end{pmatrix}
\begin{pmatrix}
v \\
\delta x \\
\end{pmatrix}
+ 
\begin{pmatrix}
0 & \Delta B_k^T \\
\Delta B_k & 0 \\
\end{pmatrix}
\begin{pmatrix}
v \\
\delta x \\
\end{pmatrix}
= 
\begin{pmatrix}
b \\
c \\
\end{pmatrix}
\Rightarrow
\begin{pmatrix}
Av + B_k^T \delta x \\
B_k \delta x \\
\end{pmatrix}
+ 
\begin{pmatrix}
\Delta B_k^T \delta x \\
\Delta B_k \delta x \\
\end{pmatrix}
= 
\begin{pmatrix}
b \\
c \\
\end{pmatrix}
\Rightarrow
\begin{pmatrix}
\Delta B_k^T \delta x \\
\Delta B_k \delta x \\
\end{pmatrix}
= 
\begin{pmatrix}
b - Av - B_k^T \delta x \\
c - B_k \delta x \\
\end{pmatrix}
Preconditioning Saddle Point Formulation of 4D-Var

\[
\begin{pmatrix}
A & B^T \\
B & 0
\end{pmatrix}
\begin{pmatrix}
v \\
\delta x
\end{pmatrix} =
\begin{pmatrix}
b \\
c
\end{pmatrix}
\Rightarrow
\begin{pmatrix}
A & B_k^T \\
B_k & 0
\end{pmatrix}
\begin{pmatrix}
v \\
\delta x
\end{pmatrix} +
\begin{pmatrix}
0 & \Delta B_k^T \\
\Delta B_k & 0
\end{pmatrix}
\begin{pmatrix}
v \\
\delta x
\end{pmatrix} =
\begin{pmatrix}
b \\
c
\end{pmatrix}
\Rightarrow
\begin{pmatrix}
Av + B_k^T \delta x \\
B_k \delta x
\end{pmatrix} +
\begin{pmatrix}
\Delta B_k^T \delta x \\
\Delta B_k \delta x
\end{pmatrix} =
\begin{pmatrix}
b \\
c
\end{pmatrix}
\Rightarrow
\begin{pmatrix}
\Delta B_k^T \delta x \\
\Delta B_k \delta x
\end{pmatrix} =
\begin{pmatrix}
b - Av - B_k^T \delta x \\
c - B_k \delta x
\end{pmatrix}
\]

Let’s define the vectors \( r_b \) and \( r_c \) as

\[
\begin{align*}
\mathbf{r}_b &= \mathbf{b} - \mathbf{A} \mathbf{v} - \mathbf{B}_k^T \delta \mathbf{x} \\
\mathbf{r}_c &= \mathbf{c} - \mathbf{B}_k \mathbf{v}
\end{align*}
\]
Preconditioning Saddle Point Formulation of 4D-Var

\[
\begin{pmatrix}
A & B^T \\
B & 0
\end{pmatrix}
\begin{pmatrix}
v \\
\delta x
\end{pmatrix}
= 
\begin{pmatrix}
b \\
c
\end{pmatrix}
\Rightarrow
\begin{pmatrix}
A & B_k^T \\
B_k & 0
\end{pmatrix}
\begin{pmatrix}
v \\
\delta x
\end{pmatrix}
+ 
\begin{pmatrix}
0 & \Delta B_k^T \\
\Delta B_k & 0
\end{pmatrix}
\begin{pmatrix}
v \\
\delta x
\end{pmatrix}
= 
\begin{pmatrix}
b \\
c
\end{pmatrix}
\Rightarrow
\begin{pmatrix}
Av + B_k^T \delta x \\
B_k \delta x
\end{pmatrix}
+ 
\begin{pmatrix}
\Delta B_k^T \delta x \\
\Delta B_k \delta x
\end{pmatrix}
= 
\begin{pmatrix}
b \\
c
\end{pmatrix}
\Rightarrow
\begin{pmatrix}
\Delta B_k^T \delta x \\
\Delta B_k \delta x
\end{pmatrix}
= 
\begin{pmatrix}
b - Av - B_k^T \delta x \\
c - B_k \delta x
\end{pmatrix}
\]

- Let’s define the vectors \( r_b \) and \( r_c \) as
  \[
  r_b = b - Av - B_k^T \delta x
  \]
  \[
  r_c = c - B_k v
  \]

- Then we have
  \[
  \Delta B_k^T \delta x = r_b
  \]
  \[
  \Delta B_k \delta x = r_c
  \]
  \[
  \rightarrow \text{We want to find an update } \Delta B_k \text{ satisfying these equations.}
  \]
Preconditioning Saddle Point Formulation of 4D-Var

- A rank-1 solution (an update to $B_k$) for the equations:
  \[
  \Delta B_k^T \delta x = r_b \\
  \Delta B_k v = r_c
  \]
  
  can be given as:

  \[
  \Delta B_k = \frac{r_c r_b^T}{v r_c^T \delta x} = \frac{(c - B_k v)(b - Av - B_k^T \delta x)^T}{(c - B_k v)^T \delta x}
  \]
Preconditioning Saddle Point Formulation of 4D-Var

- A rank-1 solution (an update to $B_k$) for the equations:
  \[
  \Delta B_k^T \delta x = r_b \\
  \Delta B_k v = r_c
  \]
  
  can be given as:
  \[
  \Delta B_k = \frac{r_c r_b^T}{v r_c^T \delta x} = \frac{(c - B_k v)(b - Av - B_k^T \delta x)^T}{(c - B_k v)^T \delta x}
  \]

- This formula can then be used to update:
  \[
  B_{k+1} = B_k + \Delta B_k
  \]
Preconditioning Saddle Point Formulation of 4D-Var

- A rank-1 solution (an update to $B_k$) for the equations:

  $$\Delta B_k^T \delta x = r_b$$
  $$\Delta B_k v = r_c$$

  can be given as:

  $$\Delta B_k = \frac{r_c r_b^T}{v \delta x} = \frac{(c - B_k v) (b - A v - B_k^T \delta x)^T}{(c - B_k v)^T \delta x}$$

- This formula can then be used to update:

  $$B_{k+1} = B_k + \Delta B_k$$

- How we will use this update in a GMRES iteration?

  $$\mathcal{P}_k = \begin{pmatrix} A & B_{k+1}^T \\ B_{k+1} & 0 \end{pmatrix} \Rightarrow \mathcal{P}_k^{-1} A_{k+1} u = \mathcal{P}_k^{-1} f_{k+1}$$

  $\mathcal{P}_k^{-1}$ can be obtained by using Sherman-Morrison-Woodbury formula.
We have shown that it is possible to find a **low-cost low-rank update** for the inexact constraint preconditioner.

This update amounts to the **two-sided-rank-one (TR1)** update proposed by Griewank and Walther (2002). They used to update Jacobian matrix in a constrained optimisation problem.
Preconditioning Saddle Point Formulation of 4D-Var

- We have shown that it is possible to find a low-cost low-rank update for the inexact constraint preconditioner.

- This update amounts to the two-sided-rank-one (TR1) update proposed by Griewank and Walther (2002). They used to update Jacobian matrix in a constrained optimization problem.

**TR1 update:**

- It generalizes the classical symmetric rank-one (SR1) update.
Preconditioning Saddle Point Formulation of 4D-Var

- We have shown that it is possible to find a low-cost low-rank update for the inexact constraint preconditioner.

- This update amounts to the two-sided-rank-one (TR1) update proposed by Griewank and Walther (2002). They used to update Jacobian matrix in a constrained optimisation problem.

**TR1 update:**

- It generalizes the classical symmetric rank-one (SR1) update.
- It maintains the validity of all previous secant conditions.
We have shown that it is possible to find a low-cost low-rank update for the inexact constraint preconditioner.

This update amounts to the two-sided-rank-one (TR1) update proposed by Griewank and Walther (2002). They used to update Jacobian matrix in a constrained optimisation problem.

**TR1 update:**

- It generalizes the classical symmetric rank-one (SR1) update.
- It maintains the validity of all previous secant conditions.
- It is invariant with respect to linear transformations.
Preconditioning Saddle Point Formulation of 4D-Var

- We have shown that it is possible to find a low-cost low-rank update for the inexact constraint preconditioner.

- This update amounts to the two-sided-rank-one (TR1) update proposed by Griewank and Walther (2002). They used to update Jacobian matrix in a constrained optimisation problem.

**TR1 update:**

- It generalizes the classical symmetric rank-one (SR1) update.
- It maintains the validity of all previous secant conditions.
- It is invariant with respect to linear transformations
- It has no least change characterization in terms of any particular matrix norm.
Least-Frobenius norm update

We are interested with the solution of the following problem:

\[
\min_{\Delta B_k} \| W_1^{-1} \Delta B_k W_2^{-1} \|_F
\]

s.t. \( \Delta B_k^T \delta x = r_b, \)
\( \Delta B_k v = r_c, \)
Least-Frobenius norm update

We are interested with the solution of the following problem:

$$\min_{\Delta B_k} \|W_1^{-1} \Delta B_k W_2^{-1}\|_F$$

s.t. $\Delta B_k^T \delta x = r_b,$
\hspace{1cm} $\Delta B_k v = r_c,$

where $W_1$ is any $m$-by-$m$ nonsingular matrix such that $W_1 W_1^T \delta x = c$, and $W_2$ is any $n$-by-$n$ nonsingular matrix such that $W_2^T W_2 v = s$. 
Least-Frobenius norm update

- We are interested with the solution of the following problem:

\[
\min_{\Delta B_k} \| W_1^{-1} \Delta B_k W_2^{-1} \|_F
\]

s.t. \( \Delta B_k^T \delta x = r_b, \)

\( \Delta B_k v = r_c, \)

where \( W_1 \) is any \( m \)-by-\( m \) nonsingular matrix such that \( W_1 W_1^T \delta x = c, \) and \( W_2 \) is any \( n \)-by-\( n \) nonsingular matrix such that \( W_2^T W_2 v = s. \)

- The solution is given by

\[
\Delta B_k = \frac{c r_b^T}{\delta x^T c} + \frac{r_c s^T}{v^T s} - \frac{c \delta x^T r_c}{\delta x^T c} s^T
\]

which is equivalent to

\[
\Delta B_k = \frac{c (b - Av - B_k^T \delta x)^T}{\delta x^T c} + \frac{(c - B_k v) s^T}{v^T s} - \frac{c \delta x^T (c - B_k v)s^T}{\delta x^T c} v^T s
\]
Least-Frobenius norm update

- It is a new update that can be used for Jacobian updates
- It has least change characterization in terms of weighted Frobenius norm
- It generalizes the classical Davidon-Fletcher-Powell (DFP) update
- It maintains the validity of all previous secant conditions (when the block formula is used)
- It is invariant with respect to linear transformations
Numerical Results

Implementation platform

- We used the Object Oriented Prediction System (OOPS) developed by ECMWF
- OOPS consists of simplified models of a real-system

The model

- It is a two-layer quasi-geostraphic model with 1600 grid-points

Implementation details

- There are 100 observations of stream function every 3 hours, 100 wind observations plus 100 wind-speed observations every 6 hours
- The error covariance matrices are assumed to be horizontally isotropic and homogeneous, with Gaussian spatial structure
- The analysis window is 24 hours, and is divided into 8 subwindows
- 3 outer loops with 10 inner loops each are performed
Methods

1. Standard weak-constrained 4D-Var formulation
   → Solution method is preconditioned conjugate-gradients

\[
P_0 = \begin{bmatrix} D_0 & \tilde{L}_0 \end{bmatrix} \begin{bmatrix} 0 & 0 \end{bmatrix} = \begin{bmatrix} \tilde{L}_0 \end{bmatrix} \begin{bmatrix} 0 & 0 \end{bmatrix} \begin{bmatrix} D_0 & \tilde{L}_0 \end{bmatrix}
\]

Second-level preconditioners:

1. $T_k$: The preconditioner obtained by using the TR1 update
2. $F_k$: The preconditioner obtained by using the least-Frobenius update
Methods

1. Standard weak-constrained 4D-Var formulation
   → Solution method is preconditioned conjugate-gradients

2. Saddle point formulation with an updated inexact constraint preconditioner
   → Solution method is GMRES
   → The initial preconditioner is chosen as

\[
P_0 = \begin{pmatrix}
  D & 0 & \tilde{L}
  \\
  0 & R & 0
  \\
  \tilde{L}^T & 0 & 0
\end{pmatrix}
\]

with

\[
\tilde{L} = \begin{pmatrix}
  I & -I & I \\
  -I & \ddots & \ddots \\
  -I & \ddots & I
\end{pmatrix}
\]
Methods

1. Standard weak-constrained 4D-Var formulation
   → Solution method is preconditioned conjugate-gradients

2. Saddle point formulation with an updated inexact constraint preconditioner
   → Solution method is GMRES
   → The initial preconditioner is chosen as

\[
P_0 = \begin{pmatrix}
D & 0 & \tilde{L} \\
0 & R & 0 \\
\tilde{L}^T & 0 & 0
\end{pmatrix}
\]  
with  
\[
\tilde{L} = \begin{pmatrix}
I \\
-1 & I \\
& \ddots & \ddots \\
& & -1 & I
\end{pmatrix}
\]

\[
P_0^{-1} = \begin{pmatrix}
0 & 0 & \tilde{L}^{-T} \\
0 & R^{-1} & 0 \\
\tilde{L}^{-1} & 0 & -\tilde{L}^{-1}D\tilde{L}^{-T}
\end{pmatrix}
\]  
and  
\[
\tilde{L}^{-1} = \begin{pmatrix}
I \\
I \\
\vdots & \ddots & \ddots \\
& & I & I
\end{pmatrix}
\]

Second-level preconditioners:

1. \( \mathcal{T}_k \): The preconditioner obtained by using the TR1 update

2. \( \mathcal{F}_k \): The preconditioner obtained by using the least-Frobenius update
The performance of the second level preconditioners

Last 8 pairs were used to construct the preconditioner

Figure: Nonlinear cost function values along iterations

- Second-level preconditioners obtained by using updates accelerate the convergence
- The performance of the least-Frobenius and TR1 update are very similar.
Overall performance compared with the standard 4DVar formulation

At each iteration the standard 4DVar formulation requires one application of $L^{-1}$, followed by one application of $L^{-T}$ (16 sequential subwindow integrations)

At each iteration of saddle point formulation require one subwindow integration (provided that $L^{-1}$ and $L^{-T}$ are applied simultaneously)
Conclusions

- The saddle point formulation of weak-constraint 4D-Var allows parallelisation in the time dimension.

- Finding an effective preconditioner is a key issue in solving the saddle point systems.

- The inexact constraint preconditioner can be used to precondition the saddle point formulation of 4D-Var.

- When solving a sequence of saddle point systems, a low-rank low-cost update formulas can be found to further improve preconditioning.

- The preconditioned GMRES algorithm for saddle point formulation is competitive with the existing algorithms and has the potential to allow 4D-Var to remain computationally viable on next-generation computer architectures.
Thank you for your attention!
Preconditioning Saddle Point Formulation of 4D-Var

- As a result, an inexact constraint preconditioner $\mathcal{P}$ can be updated from

$$
\mathcal{P}_{j+1} = \mathcal{P}_j + \begin{pmatrix}
0 & \Delta B^T \\
\Delta B & 0
\end{pmatrix} = \mathcal{P}_j + \begin{pmatrix}
0 & \alpha wv^T \\
\alpha vw^T & 0
\end{pmatrix},
$$

where $w = r_b$, $v = r_c$ and $\alpha = 1/\mathbf{v}^T\delta\mathbf{x}$.

- We can rewrite this formula as

$$
\mathcal{P}_{j+1} = \mathcal{P}_j + \left( \begin{pmatrix}
w \\
v
\end{pmatrix} \right) \left( \begin{pmatrix}
\alpha w^T & 0 \\
0 & \alpha v^T
\end{pmatrix} \\
\mathbf{F} & \mathbf{G}
\right)
$$

where $\mathbf{F}$ is an $(2n + m)$-by-2 matrix and $\mathbf{G}$ is an 2-by-$(2n + m)$ matrix.

- Using the Sherman-Morrison-Woodbury formula on this equation gives the inverse update as

$$
\mathcal{P}_{j+1}^{-1} = \mathcal{P}_j^{-1} - \mathcal{P}_j^{-1} \mathbf{F} (\mathbf{I}_2 + \mathbf{G} \mathcal{P}_j^{-1} \mathbf{F})^{-1} \mathbf{G} \mathcal{P}_j^{-1}
$$
Preconditioning Saddle Point Formulation of 4D-Var

Remember that we want to find an update such that

\[
\Delta B^T v = r_b \quad \quad \quad (1) \\
\Delta B \delta x = r_c \quad \quad \quad (2)
\]

Any solution \( \Delta B \) satisfying Equation (1) can be written as \( \text{[Lemma 2.1]} \) (Sun 1999)

\[
\Delta B = r_b u_2^\dagger + S (I - u_2^T u_2^*)^\dagger,
\]

where \( \dagger \) denotes the pseudo-inverse and \( S \) is an \((n+m)\times n\) matrix. Inserting this relation into (2) yields

\[
u_2^T r_b^T u_1 + (I - u_2^T u_2^*) S^T u_1 = r_c.
\]

If this equation admits one solution, its least Frobenius norm solution, \( \min_{S^T \in \mathbb{R}^{m \times n}} \| (r_c - u_2^T r_b^T u_1 ) - (I - u_2^T u_2^*) S^T u_1 \|_F \), can be written as \( \text{[Lemma 2.3]} \) (Sun 1999)

\[
(S^T)^* = (I - u_2^T u_2^*)^\dagger (r_c - u_2^T r_b^T u_1 ) u_1^\dagger.
\]
Preconditioning Saddle Point Formulation of 4D-Var

Remember that we want to find an update such that

\[ \Delta B^T v = r_b \]  
\[ \Delta B \delta x = r_c \]  

Any solution \( \Delta B \) satisfying Equation (1) can be written as [Lemma 2.1](Sun 1999)

\[ \Delta B^T = r_b u_2^\dagger + S(I - u_2 u_2^\dagger), \]

where \( \dagger \) denotes the pseudo-inverse and \( S \) is an \((n + m) \times n\) matrix. Inserting this relation into (2) yields

\[ u_2^T r_b^T u_1 + (I - u_2^T u_2^T) S^T u_1 = r_c. \]
Preconditioning Saddle Point Formulation of 4D-Var

Remember that we want to find an update such that

\[ \Delta B^T v = r_b \]  \hspace{1cm} (1)
\[ \Delta B \delta x = r_c \]  \hspace{1cm} (2)

Any solution \( \Delta B \) satisfying Equation (1) can be written as [Lemma 2.1](Sun 1999)

\[ \Delta B^T = r_b u_2^\dagger + S(I - u_2 u_2^\dagger), \]

where \( \dagger \) denotes the pseudo-inverse and \( S \) is an \( (n + m) \times n \) matrix. Inserting this relation into (2) yields

\[ u_2^T r_b^T u_1 + (I - u_2^T u_2^T) S^T u_1 = r_c. \]

If this equation admits one solution, its least Frobenius norm solution,

\[ \min_{S^T \in \mathbb{R}^{m \times n}} \| (r_c - u_2^T r_b^T u_1) - (I - u_2^T u_2^T) S^T u_1 \|_F, \]

can be written as [Lemma 2.3](Sun 1999)

\[ (S^T)^* = (I - u_2^T u_2^T)^\dagger (r_c - u_2^T r_b^T u_1) u_1^\dagger. \]
Substituting the solution for $S$ into $\Delta B$ yields that

$$\Delta B^* = u_2^T r_b^T + (I - u_2^T u_2^T) r_c u_1^T$$
Substituting the solution for $S$ into $\Delta B$ yields that

$$\Delta B^* = u_2^{T\dagger} r_b^T + (I - u_2^{T\dagger} u_2^T) r_c u_1^{\dagger}$$

This formula can be rewritten as

$$\Delta B^* = \begin{bmatrix} \delta x^{T\dagger} & r_c & -\delta x^{T\dagger} \end{bmatrix} \begin{bmatrix} r_b^T \newline v^{\dagger} \newline \delta x^T r_c v^{\dagger} \end{bmatrix} = VW^T,$$

where $V$ is an $m$-by-3 matrix and $W$ is an $2n$-by-3 matrix.
Substituting the solution for $S$ into $\Delta B$ yields that

$$\Delta B^* = u_2^T r_b + (I - u_2^T u_2^T) r_c u_1^T$$

This formula can be rewritten as

$$\Delta B^* = \begin{bmatrix} \delta x^T & r_c & -\delta x^T \end{bmatrix} \begin{bmatrix} r_b^T \\ v^T \\ \delta x^T r_c v^T \end{bmatrix} = VW^T,$$

where $V$ is an $m$-by-3 matrix and $W$ is an $2n$-by-3 matrix.

The preconditioner can be updated by using the following formula

$$P_1 = P_0 + \begin{bmatrix} 0 & WV^T \\ VW & 0 \end{bmatrix} = P_0 + \begin{bmatrix} 0 & W \\ V & 0 \end{bmatrix} \begin{bmatrix} W^T & 0 \\ 0 & V^T \end{bmatrix} \tag{F, G}.$$
The inverse formula is then given by

\[
P_F^{-1} = P_0^{-1} - P_0^{-1}F(I_4 + G P_0^{-1} F)^{-1} G P_0^{-1}
\]

where \( F \) is an \((2n + m)\)-by-4 matrix and \( G \) is an 4-by-\((2n + m)\) matrix.
The inverse formula is then given by

\[
\mathcal{P}_F^{-1} = \mathcal{P}_0^{-1} - \mathcal{P}_0^{-1}F(I_4 + GP_0^{-1}F)^{-1}GP_0^{-1}
\]

where \( F \) is an \((2n + m)\)-by-4 matrix and \( G \) is an 4-by-(\(2n + m\)) matrix.

Let’s remember the first formula:

\[
\mathcal{P}_T^{-1} = \mathcal{P}_0^{-1} - \mathcal{P}_0^{-1}F(I_2 + GP_0^{-1}F)^{-1}GP_0^{-1}
\]

The least Frobenius norm update is slightly more expensive than the first update however it is more stable.

→ It can be shown that \( \|\mathcal{P}_T^{-1}\|_F \) can be arbitrarily larger than \( \|\mathcal{P}_F^{-1}\|_F \)