# Numerical solution for a time-parallelized formulation of 4DVAR

M. Fisher[1], S. Gratton[2,3], S. Gürol [3]

[1]ECMWF, Reading, UK
[2]ENSEEIHT, Toulouse, France
[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.
  $\rightarrow$ Tangent Linear and Adjoint integrations run one after the other.
  $\rightarrow$ 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.

  $\rightarrow$ 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

$$\boxed{\min_{\mathbf{x}\in\mathbb{R}^n} \frac{1}{2}\|x_0 - x_b\|^2_{\mathbf{B}^{-1}} + \frac{1}{2}\sum_{j=0}^{N}\|\mathcal{H}_j(x_j) - y_j\|^2_{\mathbf{R}_j^{-1}} + \frac{1}{2}\sum_{j=1}^{N}\|\underbrace{x_j - \mathcal{M}_j(x_{j-1})}_{q_j}\|^2_{\mathbf{Q}_j^{-1}}}$$

- $\mathbf{x} = \begin{pmatrix} x_0 \\ x_1 \\ \vdots \\ x_N \end{pmatrix} \in \mathbb{R}^n$ 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
- $\mathcal{M}_j$ represents an integration of the numerical model from time $t_{j-1}$ to $t_j$
- $\mathbf{B}$, $\mathbf{R}_j$ and $\mathbf{Q}_j$ 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:

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

- This system is called the saddle-point formulation of 4D-Var.
- $\mathbf{L} = \begin{pmatrix} I & & & & \\ -M_1 & I & & & \\ & -M_2 & I & & \\ & & \ddots & \ddots & \\ & & & -M_N & I \end{pmatrix}$ is an n-by-n matrix.
- $\mathbf{H} = diag(\mathbf{H_0}, \mathbf{H_1}, \dots, \mathbf{H_N})$ is an n-by-m matrix.
- $\mathbf{D} = diag(\mathbf{B}, \mathbf{Q_1}, \dots, \mathbf{Q_N})$ is an n-by-n matrix.
- $\mathbf{R} = diag(\mathbf{R_0}, \mathbf{R_1}, \dots, \mathbf{R_N})$ is an m-by-m matrix.

# Parallelization in the time dimension

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

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

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

$\rightarrow$ Matrix-vector products with $\mathbf{L}$ can be parallelized in the time dimension

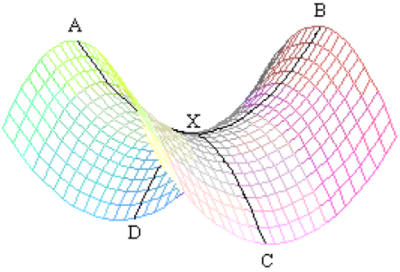
- Note that the matrix contains no inverse matrices.

# Properties of the saddle point system

$$\mathcal{A} = \begin{pmatrix} \mathbf{D} & \mathbf{0} & \mathbf{L} \\ \mathbf{0} & \mathbf{R} & \mathbf{H} \\ \mathbf{L}^{\mathrm{T}} & \mathbf{H}^{\mathrm{T}} & \mathbf{0} \end{pmatrix} = \begin{pmatrix} \mathbf{A} & \mathbf{B}^{\mathrm{T}} \\ \mathbf{B} & \mathbf{0} \end{pmatrix}$$

- $\mathcal{A}$ is a $(2n+m)$-by-$(2n+m)$ indefinite symmetric matrix. $\mathcal{A}$ has negative and positive eigenvalues.
- The solution of this problem is a saddle point.
- $\mathbf{A}$ is symmetric positive definite, i.e. $\mathbf{x}^{\mathbf{T}}\mathbf{A}\mathbf{x} > \mathbf{0}$
- If the schur complement $\mathbf{S} = -\mathbf{B}\mathbf{A}^{-1}\mathbf{B}^{\mathbf{T}}$ is negative definite, then $\mathcal{A}$ 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 $\mathcal{P}$

  - is an approximation to $\mathcal{A}$
  - 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 $\mathcal{P}$
  - is an approximation to $\mathcal{A}$
  - 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?

$$\mathcal{A} = \begin{pmatrix} \mathbf{D} & \mathbf{0} & \mathbf{L} \\ \mathbf{0} & \mathbf{R} & \mathbf{H} \\ \mathbf{L}^{\mathrm{T}} & \mathbf{H}^{\mathrm{T}} & \mathbf{0} \end{pmatrix} = \begin{pmatrix} \mathbf{A} & \mathbf{B}^{\mathrm{T}} \\ \mathbf{B} & \mathbf{0} \end{pmatrix}$$

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

  $\Rightarrow$ Nice review is given by Benzi, Golub and Liesen (2005).

- Most preconditioners in the literature assume that $\mathbf{D}$ and $\mathbf{R}$ are expensive, and $\mathbf{L}$ and $\mathbf{H}$ are cheap.

# How to precondition?

$$\mathcal{A} = \begin{pmatrix} \mathbf{D} & \mathbf{0} & \mathbf{L} \\ \mathbf{0} & \mathbf{R} & \mathbf{H} \\ \mathbf{L}^{\mathrm{T}} & \mathbf{H}^{\mathrm{T}} & \mathbf{0} \end{pmatrix} = \begin{pmatrix} \mathbf{A} & \mathbf{B}^{\mathrm{T}} \\ \mathbf{B} & \mathbf{0} \end{pmatrix}$$

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

  $\Rightarrow$ Nice review is given by Benzi, Golub and Liesen (2005).

- Most preconditioners in the literature assume that $\mathbf{D}$ and $\mathbf{R}$ are expensive, and $\mathbf{L}$ and $\mathbf{H}$ are cheap.

- The opposite is true in our case! $\mathbf{B}$ is the most computationally expensive block and calculations involving $\mathbf{A}$ are relatively cheap.

# How to precondition?

$$\mathcal{A} = \begin{pmatrix} \mathbf{D} & \mathbf{0} & \mathbf{L} \\ \mathbf{0} & \mathbf{R} & \mathbf{H} \\ \mathbf{L}^{\mathrm{T}} & \mathbf{H}^{\mathrm{T}} & \mathbf{0} \end{pmatrix} = \begin{pmatrix} \mathbf{A} & \mathbf{B}^{\mathrm{T}} \\ \mathbf{B} & \mathbf{0} \end{pmatrix}$$

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

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

where
  - $\widetilde{\mathbf{L}}$ is an approximation to the matrix $\mathbf{L}$
  - $\widetilde{\mathbf{B}} = [\widetilde{\mathbf{L}}^{\mathrm{T}} \quad \mathbf{0}]$ is a full row rank approximation of the matrix $\mathbf{B} \in \mathbb{R}^{n\times(m+n)}$

# Second-level preconditioner

$$\underbrace{\begin{pmatrix} \mathbf{D} & \mathbf{0} & \mathbf{L} \\ \mathbf{0} & \mathbf{R} & \mathbf{H} \\ \mathbf{L}^{\mathrm{T}} & \mathbf{H}^{\mathrm{T}} & \mathbf{0} \end{pmatrix}}_{\mathcal{A}_k} \underbrace{\begin{pmatrix} \boldsymbol{\lambda} \\ \boldsymbol{\mu} \\ \delta\mathbf{x} \end{pmatrix}}_{\mathbf{u}} = \underbrace{\begin{pmatrix} \mathbf{b} \\ \mathbf{d} \\ \mathbf{0} \end{pmatrix}}_{\mathbf{f}_k}$$

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 $\mathcal{A}^{-1}$ or its effect on a vector?

# Preconditioning Saddle Point Formulation of 4D-Var

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

$$\mathcal{P}_0 = \begin{pmatrix} \mathbf{A} & \mathbf{B}_0^{\mathrm{T}} \\ \mathbf{B}_0 & \mathbf{0} \end{pmatrix} \quad \Rightarrow \quad \mathcal{P}_0^{-1} \mathcal{A}_1 \, \mathbf{u} = \mathcal{P}_0^{-1} \mathbf{f}_1$$

# Preconditioning Saddle Point Formulation of 4D-Var

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

$$\mathcal{P}_0 = \begin{pmatrix} \mathbf{A} & \mathbf{B}_0^{\mathrm{T}} \\ \mathbf{B}_0 & \mathbf{0} \end{pmatrix} \quad \Rightarrow \quad \mathcal{P}_0^{-1} \mathcal{A}_1 \, \mathbf{u} = \mathcal{P}_0^{-1} \mathbf{f}_1$$

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

$$\mathcal{P}_k = \begin{pmatrix} \mathbf{A} & \mathbf{B}_k^{\mathrm{T}} \\ \mathbf{B}_k & \mathbf{0} \end{pmatrix} + \begin{pmatrix} \mathbf{0} & \Delta\mathbf{B}_k^{\mathrm{T}} \\ \Delta\mathbf{B}_k & \mathbf{0} \end{pmatrix} \Rightarrow \mathcal{P}_k^{-1} \mathcal{A}_{k+1} \, \mathbf{u} = \mathcal{P}_k^{-1} \mathbf{f}_{k+1}$$

# Preconditioning Saddle Point Formulation of 4D-Var

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

$$\mathcal{P}_0 = \begin{pmatrix} \mathbf{A} & \mathbf{B}_0^{\mathrm{T}} \\ \mathbf{B}_0 & \mathbf{0} \end{pmatrix} \quad \Rightarrow \quad \mathcal{P}_0^{-1} \mathcal{A}_1 \, \mathbf{u} = \mathcal{P}_0^{-1} \mathbf{f}_1$$

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

$$\mathcal{P}_k = \begin{pmatrix} \mathbf{A} & \mathbf{B}_k^{\mathrm{T}} \\ \mathbf{B}_k & \mathbf{0} \end{pmatrix} + \begin{pmatrix} \mathbf{0} & \Delta\mathbf{B}_k^{\mathrm{T}} \\ \Delta\mathbf{B}_k & \mathbf{0} \end{pmatrix} \Rightarrow \mathcal{P}_k^{-1} \mathcal{A}_{k+1} \, \mathbf{u} = \mathcal{P}_k^{-1} \mathbf{f}_{k+1}$$

## How to obtain these updates?

$\rightarrow$ GMRES performs matrix-vector products with $\mathcal{A}$ :

$$\underbrace{\begin{pmatrix} \mathbf{A} & \mathbf{B}^{\mathrm{T}} \\ \mathbf{B} & \mathbf{0} \end{pmatrix}}_{\mathcal{A}_k} \underbrace{\begin{pmatrix} \mathbf{v}_j \\ \delta \boldsymbol{x}_j \end{pmatrix}}_{\mathbf{u}_j^{(k)}} = \underbrace{\begin{pmatrix} \mathbf{b}_j \\ \mathbf{c}_j \end{pmatrix}}_{\mathbf{f}_j^{(k)}}$$

$\rightarrow$ We can use the pairs $(\mathbf{u}_j^{(k)}, \mathbf{f}_j^{(k)})$ to find an update $\Delta \mathbf{B}_k$.

# Preconditioning Saddle Point Formulation of 4D-Var

$$\begin{pmatrix} \mathbf{A} & \mathbf{B}^{\mathrm{T}} \\ \mathbf{B} & \mathbf{0} \end{pmatrix} \begin{pmatrix} \mathbf{v} \\ \delta \boldsymbol{x} \end{pmatrix} = \begin{pmatrix} \mathbf{b} \\ \mathbf{c} \end{pmatrix} \quad \Rightarrow \quad \begin{pmatrix} \mathbf{A} & \mathbf{B}_k^{\mathrm{T}} \\ \mathbf{B}_k & \mathbf{0} \end{pmatrix} \begin{pmatrix} \mathbf{v} \\ \delta \boldsymbol{x} \end{pmatrix} + \begin{pmatrix} \mathbf{0} & \Delta\mathbf{B}_k^{\mathrm{T}} \\ \Delta\mathbf{B}_k & \mathbf{0} \end{pmatrix} \begin{pmatrix} \mathbf{v} \\ \delta \boldsymbol{x} \end{pmatrix} = \begin{pmatrix} \mathbf{b} \\ \mathbf{c} \end{pmatrix}$$

$$\Rightarrow \quad \begin{pmatrix} \mathbf{A}\mathbf{v} + \mathbf{B}_k^{\mathrm{T}} \delta \boldsymbol{x} \\ \mathbf{B}_k \mathbf{v} \end{pmatrix} + \begin{pmatrix} \Delta\mathbf{B}_k^{\mathrm{T}} \delta \boldsymbol{x} \\ \Delta\mathbf{B}_k \mathbf{v} \end{pmatrix} = \begin{pmatrix} \mathbf{b} \\ \mathbf{c} \end{pmatrix}$$

$$\Rightarrow \quad \begin{pmatrix} \Delta\mathbf{B}_k^{\mathrm{T}} \delta \boldsymbol{x} \\ \Delta\mathbf{B}_k \mathbf{v} \end{pmatrix} = \begin{pmatrix} \mathbf{b} - \mathbf{A}\mathbf{v} - \mathbf{B}_{\mathbf{k}}^{\mathrm{T}} \delta \boldsymbol{x} \\ \mathbf{c} - \mathbf{B}_{\mathbf{k}} \mathbf{v} \end{pmatrix}$$

# Preconditioning Saddle Point Formulation of 4D-Var

$$\begin{pmatrix} \mathbf{A} & \mathbf{B}^{\mathrm{T}} \\ \mathbf{B} & \mathbf{0} \end{pmatrix} \begin{pmatrix} \mathbf{v} \\ \delta \boldsymbol{x} \end{pmatrix} = \begin{pmatrix} \mathbf{b} \\ \mathbf{c} \end{pmatrix} \quad \Rightarrow \quad \begin{pmatrix} \mathbf{A} & \mathbf{B}_k^{\mathrm{T}} \\ \mathbf{B}_k & \mathbf{0} \end{pmatrix} \begin{pmatrix} \mathbf{v} \\ \delta \boldsymbol{x} \end{pmatrix} + \begin{pmatrix} \mathbf{0} & \Delta\mathbf{B}_k^{\mathrm{T}} \\ \Delta\mathbf{B}_k & \mathbf{0} \end{pmatrix} \begin{pmatrix} \mathbf{v} \\ \delta \boldsymbol{x} \end{pmatrix} = \begin{pmatrix} \mathbf{b} \\ \mathbf{c} \end{pmatrix}$$

$$\Rightarrow \quad \begin{pmatrix} \mathbf{A}\mathbf{v} + \mathbf{B}_k^{\mathrm{T}} \delta \boldsymbol{x} \\ \mathbf{B}_k \mathbf{v} \end{pmatrix} + \begin{pmatrix} \Delta\mathbf{B}_k^{\mathrm{T}} \delta \boldsymbol{x} \\ \Delta\mathbf{B}_k \mathbf{v} \end{pmatrix} = \begin{pmatrix} \mathbf{b} \\ \mathbf{c} \end{pmatrix}$$

$$\Rightarrow \quad \begin{pmatrix} \Delta\mathbf{B}_k^{\mathrm{T}} \delta \boldsymbol{x} \\ \Delta\mathbf{B}_k \mathbf{v} \end{pmatrix} = \begin{pmatrix} \mathbf{b} - \mathbf{A}\mathbf{v} - \mathbf{B}_{\mathbf{k}}^{\mathrm{T}} \delta \boldsymbol{x} \\ \mathbf{c} - \mathbf{B}_{\mathbf{k}} \mathbf{v} \end{pmatrix}$$

- Let's define the vectors $\mathbf{r}_b$ and $\mathbf{r}_c$ as

$$\mathbf{r}_b = \mathbf{b} - \mathbf{A}\mathbf{v} - \mathbf{B}_k^{\mathrm{T}} \delta \boldsymbol{x}$$
$$\mathbf{r}_c = \mathbf{c} - \mathbf{B}_k \mathbf{v}$$

# Preconditioning Saddle Point Formulation of 4D-Var

$$\begin{pmatrix} \mathbf{A} & \mathbf{B}^{\mathrm{T}} \\ \mathbf{B} & \mathbf{0} \end{pmatrix} \begin{pmatrix} \mathbf{v} \\ \delta \boldsymbol{x} \end{pmatrix} = \begin{pmatrix} \mathbf{b} \\ \mathbf{c} \end{pmatrix} \quad \Rightarrow \quad \begin{pmatrix} \mathbf{A} & \mathbf{B}_k^{\mathrm{T}} \\ \mathbf{B}_k & \mathbf{0} \end{pmatrix} \begin{pmatrix} \mathbf{v} \\ \delta \boldsymbol{x} \end{pmatrix} + \begin{pmatrix} \mathbf{0} & \Delta\mathbf{B}_k^{\mathrm{T}} \\ \Delta\mathbf{B}_k & \mathbf{0} \end{pmatrix} \begin{pmatrix} \mathbf{v} \\ \delta \boldsymbol{x} \end{pmatrix} = \begin{pmatrix} \mathbf{b} \\ \mathbf{c} \end{pmatrix}$$

$$\Rightarrow \quad \begin{pmatrix} \mathbf{A}\mathbf{v} + \mathbf{B}_k^{\mathrm{T}} \delta \boldsymbol{x} \\ \mathbf{B}_k \mathbf{v} \end{pmatrix} + \begin{pmatrix} \Delta\mathbf{B}_k^{\mathrm{T}} \delta \boldsymbol{x} \\ \Delta\mathbf{B}_k \mathbf{v} \end{pmatrix} = \begin{pmatrix} \mathbf{b} \\ \mathbf{c} \end{pmatrix}$$

$$\Rightarrow \quad \begin{pmatrix} \Delta\mathbf{B}_k^{\mathrm{T}} \delta \boldsymbol{x} \\ \Delta\mathbf{B}_k \mathbf{v} \end{pmatrix} = \begin{pmatrix} \mathbf{b} - \mathbf{A}\mathbf{v} - \mathbf{B}_{\mathbf{k}}^{\mathrm{T}} \delta \boldsymbol{x} \\ \mathbf{c} - \mathbf{B}_{\mathbf{k}} \mathbf{v} \end{pmatrix}$$

- Let's define the vectors $\mathbf{r}_b$ and $\mathbf{r}_c$ as

$$\mathbf{r}_b = \mathbf{b} - \mathbf{A}\mathbf{v} - \mathbf{B}_k^{\mathrm{T}} \delta \boldsymbol{x}$$
$$\mathbf{r}_c = \mathbf{c} - \mathbf{B}_k \mathbf{v}$$

- Then we have

$$\Delta\mathbf{B}_k^{\mathrm{T}} \delta \boldsymbol{x} = \mathbf{r}_b$$
$$\Delta\mathbf{B}_k \mathbf{v} = \mathbf{r}_c$$

$\rightarrow$ We want to find an update $\Delta\mathbf{B}_k$ satisfying these equations.

# Preconditioning Saddle Point Formulation of 4D-Var

- A rank-1 solution (an update to $\mathbf{B}_k$) for the equations:

$$\Delta\mathbf{B}_k^{\mathrm{T}}\delta\boldsymbol{x} = \mathbf{r}_b$$
$$\Delta\mathbf{B}_k\mathbf{v} = \mathbf{r}_c$$

can be given as:

$$\boxed{\Delta\mathbf{B}_k = \frac{\mathbf{r}_c\mathbf{r}_b^T}{\mathbf{v}\mathbf{r}_c^T\delta\boldsymbol{x}} = \frac{(\mathbf{c}-\mathbf{B}_k\mathbf{v})(\mathbf{b}-\mathbf{A}\mathbf{v}-\mathbf{B}_k^{\mathrm{T}}\delta\boldsymbol{x})^T}{(\mathbf{c}-\mathbf{B}_k\mathbf{v})^T\delta\boldsymbol{x}}}$$

# Preconditioning Saddle Point Formulation of 4D-Var

- A rank-1 solution (an update to $\mathbf{B}_k$) for the equations:

$$\Delta \mathbf{B}_k^{\mathrm{T}} \delta \boldsymbol{x} = \mathbf{r}_b$$
$$\Delta \mathbf{B}_k \mathbf{v} = \mathbf{r}_c$$

can be given as:

$$\boxed{\Delta \mathbf{B}_k = \frac{\mathbf{r}_c \mathbf{r}_b^T}{\mathbf{v}\mathbf{r}_c^T \delta \boldsymbol{x}} = \frac{(\mathbf{c} - \mathbf{B}_k \mathbf{v})(\mathbf{b} - \mathbf{A}\mathbf{v} - \mathbf{B}_k^{\mathrm{T}} \delta \boldsymbol{x})^T}{(\mathbf{c} - \mathbf{B}_k \mathbf{v})^T \delta \boldsymbol{x}}}$$

- This formula can then be used to update:

$$\mathbf{B}_{k+1} = \mathbf{B}_k + \Delta \mathbf{B}_k$$

# Preconditioning Saddle Point Formulation of 4D-Var

- A rank-1 solution (an update to $\mathbf{B}_k$) for the equations:

$$\Delta \mathbf{B}_k^{\mathrm{T}} \delta \boldsymbol{x} = \mathbf{r}_b$$
$$\Delta \mathbf{B}_k \mathbf{v} = \mathbf{r}_c$$

can be given as:

$$\boxed{\Delta \mathbf{B}_k = \frac{\mathbf{r}_c \mathbf{r}_b^T}{\mathbf{v}\mathbf{r}_c^T \delta \boldsymbol{x}} = \frac{(\mathbf{c} - \mathbf{B}_k \mathbf{v})(\mathbf{b} - \mathbf{A}\mathbf{v} - \mathbf{B}_k^{\mathrm{T}} \delta \boldsymbol{x})^T}{(\mathbf{c} - \mathbf{B}_k \mathbf{v})^T \delta \boldsymbol{x}}}$$

- This formula can then be used to update:

$$\mathbf{B}_{k+1} = \mathbf{B}_k + \Delta \mathbf{B}_k$$

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

$$\mathcal{P}_k = \begin{pmatrix} \mathbf{A} & \mathbf{B}_{k+1}^{\mathrm{T}} \\ \mathbf{B}_{k+1} & \mathbf{0} \end{pmatrix} \quad \Rightarrow \quad \mathcal{P}_k^{-1} \mathcal{A}_{k+1} \mathbf{u} = \mathcal{P}_k^{-1} \mathbf{f}_{k+1}$$

- $\mathcal{P}_k^{-1}$ can be obtained by using Sherman-Morrison-Woodbury formula.

# 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.

# 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.

# 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.

# 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

# 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:

$$\begin{aligned} &\min_{\Delta B_k} \|\mathbf{W}_1^{-1} \Delta \mathbf{B}_k \mathbf{W}_2^{-1}\|_F \\ \text{s.t. } &\Delta \mathbf{B}_k^T \delta \boldsymbol{x} = \mathbf{r}_b, \\ &\Delta \mathbf{B}_k \, \mathbf{v} = \mathbf{r}_c, \end{aligned}$$

# Least-Frobenius norm update

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

$$\begin{aligned} &\min_{\Delta B_k} \|\mathbf{W}_1^{-1} \Delta \mathbf{B}_k \mathbf{W}_2^{-1}\|_F \\ \text{s.t. } &\Delta \mathbf{B}_k^T \delta \boldsymbol{x} = \mathbf{r}_b, \\ &\Delta \mathbf{B}_k \, \mathbf{v} = \mathbf{r}_c, \end{aligned}$$

where $\mathbf{W}_1$ is any $m$-by-$m$ nonsingular matrix such that $\mathbf{W}_1 \mathbf{W}_1^T \, \delta \boldsymbol{x} = \mathbf{c}$, and $\mathbf{W}_2$ is any $n$-by-$n$ nonsingular matrix such that $\mathbf{W}_2^T \mathbf{W}_2 \, \mathbf{v} = \mathbf{s}$.

# Least-Frobenius norm update

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

$$\min_{\Delta B_k} \|\mathbf{W}_1^{-1} \Delta \mathbf{B}_k \mathbf{W}_2^{-1}\|_F$$
$$\text{s.t. } \Delta \mathbf{B}_k^T \delta \boldsymbol{x} = \mathbf{r}_b,$$
$$\Delta \mathbf{B}_k \, \mathbf{v} = \mathbf{r}_c,$$

  where $\mathbf{W}_1$ is any $m$-by-$m$ nonsingular matrix such that $\mathbf{W}_1 \mathbf{W}_1^T \, \delta \boldsymbol{x} = \mathbf{c}$, and $\mathbf{W}_2$ is any $n$-by-$n$ nonsingular matrix such that $\mathbf{W}_2^T \mathbf{W}_2 \, \mathbf{v} = \mathbf{s}$.

- The solution is given by

$$\Delta \mathbf{B}_k = \frac{\mathbf{c} \, \mathbf{r}_b^T}{\delta \mathbf{x}^T \mathbf{c}} + \frac{r_c \, \mathbf{s}^T}{\mathbf{v}^T \mathbf{s}} - \frac{\mathbf{c} \, \delta \mathbf{x}^T \mathbf{r}_c \, \mathbf{s}^T}{\delta \mathbf{x}^T \mathbf{c} \, \mathbf{v}^T \mathbf{s}}$$

  which is equivalent to

$$\Delta \mathbf{B}_k = \frac{\mathbf{c} \, (\mathbf{b} - \mathbf{A}\mathbf{v} - \mathbf{B}_k^T \delta \boldsymbol{x})^T}{\delta \mathbf{x}^T \mathbf{c}} + \frac{(\mathbf{c} - \mathbf{B}_k \mathbf{v}) \, \mathbf{s}^T}{\mathbf{v}^T \mathbf{s}} - \frac{\mathbf{c} \, \delta \mathbf{x}^T (\mathbf{c} - \mathbf{B}_k \mathbf{v}) \mathbf{s}^T}{\delta \mathbf{x}^T \mathbf{c} \, \mathbf{v}^T \mathbf{s}}$$

# 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
   $\rightarrow$ Solution method is preconditioned conjugate-gradients

## Methods

1. Standard weak-constrained 4D-Var formulation
   $\rightarrow$ Solution method is preconditioned conjugate-gradients
2. Saddle point formulation with an updated inexact constraint preconditioner
   $\rightarrow$ Solution method is GMRES
   $\rightarrow$ The initial preconditioner is chosen as

$$\mathcal{P}_0 = \begin{pmatrix} \mathbf{D} & \mathbf{0} & \widetilde{\mathbf{L}} \\ \mathbf{0} & \mathbf{R} & \mathbf{0} \\ \widetilde{\mathbf{L}}^{\mathrm{T}} & \mathbf{0} & \mathbf{0} \end{pmatrix} \quad \text{with} \quad \widetilde{\mathbf{L}} = \begin{pmatrix} \mathbf{I} & & & \\ -\mathbf{I} & \mathbf{I} & & \\ & \ddots & \ddots & \\ & & -\mathbf{I} & \mathbf{I} \end{pmatrix}.$$

## Methods

1. Standard weak-constrained 4D-Var formulation
   $\rightarrow$ Solution method is preconditioned conjugate-gradients
2. Saddle point formulation with an updated inexact constraint preconditioner
   $\rightarrow$ Solution method is GMRES
   $\rightarrow$ The initial preconditioner is chosen as

$$\mathcal{P}_0 = \begin{pmatrix} \mathbf{D} & \mathbf{0} & \widetilde{\mathbf{L}} \\ \mathbf{0} & \mathbf{R} & \mathbf{0} \\ \widetilde{\mathbf{L}}^{\mathrm{T}} & \mathbf{0} & \mathbf{0} \end{pmatrix} \quad \text{with} \quad \widetilde{\mathbf{L}} = \begin{pmatrix} \mathbf{I} & & & \\ -\mathbf{I} & \mathbf{I} & & \\ & \ddots & \ddots & \\ & & -\mathbf{I} & \mathbf{I} \end{pmatrix}.$$

$$\mathcal{P}_0^{-1} = \begin{pmatrix} \mathbf{0} & \mathbf{0} & \widetilde{\mathbf{L}}^{-\mathrm{T}} \\ \mathbf{0} & \mathbf{R}^{-1} & \mathbf{0} \\ \widetilde{\mathbf{L}}^{-1} & \mathbf{0} & -\widetilde{\mathbf{L}}^{-1}\mathbf{D}\widetilde{\mathbf{L}}^{-\mathrm{T}} \end{pmatrix} \quad \text{and} \quad \widetilde{\mathbf{L}}^{-1} = \begin{pmatrix} \mathbf{I} & & & \\ \mathbf{I} & \mathbf{I} & & \\ \vdots & \ddots & \ddots & \\ \mathbf{I} & \dots & \mathbf{I} & \mathbf{I} \end{pmatrix}.$$

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

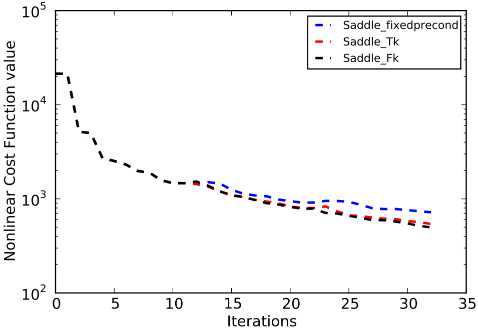



Figure: Nonlinear cost function values along iterations

- 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

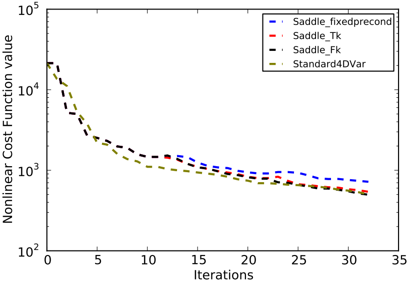



Figure: Nonlinear cost function values along iterations

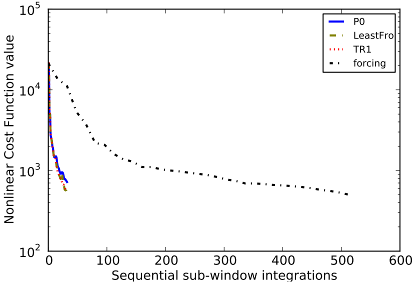



Figure: Nonlinear cost function values along sequential subwindow integrations

- At each iteration the standard 4DVar formulation requires one application of $\mathbf{L}^{-1}$, followed by one application of $\mathbf{L}^{-\mathrm{T}}$ (16 sequential subwindow integrations)
- At each iteration of saddle point formulation require one subwindow integration (provided that $\mathbf{L}^{-1}$ and $\mathbf{L}^{-\mathrm{T}}$ 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

$$\mathcal{P}_{j+1} = \mathcal{P}_j + \begin{pmatrix} \mathbf{0} & \Delta\mathbf{B}^T \\ \Delta\mathbf{B} & \mathbf{0} \end{pmatrix} = \mathcal{P}_j + \begin{pmatrix} \mathbf{0} & \alpha\mathbf{w}\mathbf{v}^{\mathrm{T}} \\ \alpha\mathbf{v}\mathbf{w}^{\mathrm{T}} & \mathbf{0} \end{pmatrix},$$

  where $\mathbf{w} = \mathbf{r}_b$, $\mathbf{v} = \mathbf{r}_c$ and $\alpha = 1/\mathbf{v}^{\mathrm{T}}\delta\boldsymbol{x}$.

- We can rewrite this formula as

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

  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

$$\boxed{\mathcal{P}_{j+1}^{-1} = \mathcal{P}_j^{-1} - \mathcal{P}_j^{-1}\mathbf{F}(\mathbf{I}_2 + \mathbf{G}\mathcal{P}_j^{-1}\mathbf{F})^{-1}\mathbf{G}\mathcal{P}_j^{-1}}$$

# Preconditioning Saddle Point Formulation of 4D-Var

- Remember that we want to find an update such that

$$\Delta\mathbf{B}^{\mathrm{T}}\mathbf{v} = \mathbf{r}_b \tag{1}$$

$$\Delta\mathbf{B}\,\delta\boldsymbol{x} = \mathbf{r}_c \tag{2}$$

# Preconditioning Saddle Point Formulation of 4D-Var

- Remember that we want to find an update such that

$$\Delta \mathbf{B}^{\mathrm{T}} \mathbf{v} = \mathbf{r}_b \tag{1}$$

$$\Delta \mathbf{B}\, \delta \mathbf{x} = \mathbf{r}_c \tag{2}$$

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

$$\Delta \mathbf{B}^{\mathrm{T}} = \mathbf{r}_b {\mathbf{u}_2}^{\dagger} + \mathbf{S}(\mathbf{I} - \mathbf{u}_2 \mathbf{u}_2^{\dagger}),$$

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

$${\mathbf{u}_2}^{\mathrm{T}\dagger} {\mathbf{r}_b}^{\mathrm{T}} \mathbf{u}_1 + (\mathbf{I} - {\mathbf{u}_2}^{\mathrm{T}\dagger} {\mathbf{u}_2}^{\mathrm{T}}) \mathbf{S}^{\mathrm{T}} \mathbf{u}_1 = \mathbf{r}_c.$$

# Preconditioning Saddle Point Formulation of 4D-Var

- Remember that we want to find an update such that

$$\Delta\mathbf{B}^{\mathrm{T}}\mathbf{v} = \mathbf{r}_b \tag{1}$$

$$\Delta\mathbf{B}\,\delta\boldsymbol{x} = \mathbf{r}_c \tag{2}$$

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

$$\Delta\mathbf{B}^{\mathrm{T}} = \mathbf{r}_b\mathbf{u}_2{}^{\dagger} + \mathbf{S}(\mathbf{I} - \mathbf{u}_2\mathbf{u}_2^{\dagger}),$$

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

$$\mathbf{u}_2{}^{\mathrm{T}\dagger}\mathbf{r}_b{}^{\mathrm{T}}\mathbf{u}_1 + (\mathbf{I} - \mathbf{u}_2{}^{\mathrm{T}\dagger}\mathbf{u}_2{}^{\mathrm{T}})\mathbf{S}^{\mathrm{T}}\mathbf{u}_1 = \mathbf{r}_c.$$

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

$$\min_{\mathbf{S}^{\mathrm{T}}\in\mathbb{R}^{m\times n}} \|(\mathbf{r}_c - \mathbf{u}_2{}^{\mathrm{T}\dagger}\mathbf{r}_b^{\mathrm{T}}\mathbf{u}_1) - (\mathbf{I} - \mathbf{u}_2{}^{\mathrm{T}\dagger}\mathbf{u}_2{}^{\mathrm{T}})\mathbf{S}^{\mathrm{T}}\mathbf{u}_1\|_F,$$

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

$$(\mathbf{S}^{\mathrm{T}})^* = (\mathbf{I} - \mathbf{u}_2{}^{\mathrm{T}\dagger}\mathbf{u}_2{}^{\mathrm{T}})^{\dagger}(\mathbf{r}_c - \mathbf{u}_2{}^{\mathrm{T}\dagger}\mathbf{r}_b^{\mathrm{T}}\mathbf{u}_1)\mathbf{u}_1{}^{\dagger}.$$

# Preconditioning Saddle Point Formulation of 4D-Var

- Substituting the solution for $\mathbf{S}$ into $\Delta\mathbf{B}$ yields that

$$\boxed{\Delta\mathbf{B}^{*} = {\mathbf{u}_2}^{\mathrm{T}\dagger}\mathbf{r}_b^{\mathrm{T}} + (\mathbf{I} - {\mathbf{u}_2}^{\mathrm{T}\dagger}{\mathbf{u}_2}^{\mathrm{T}})\mathbf{r}_c{\mathbf{u}_1}^{\dagger}}$$

# Preconditioning Saddle Point Formulation of 4D-Var

- Substituting the solution for $\mathbf{S}$ into $\Delta\mathbf{B}$ yields that

$$\boxed{\Delta\mathbf{B}^* = {\mathbf{u}_2}^{\mathrm{T}\dagger}\mathbf{r}_b^{\mathrm{T}} + (\mathbf{I} - {\mathbf{u}_2}^{\mathrm{T}\dagger}{\mathbf{u}_2}^{\mathrm{T}})\mathbf{r}_c{\mathbf{u}_1}^{\dagger}}$$

- This formula can be rewritten as

$$\Delta\mathbf{B}^* = \begin{bmatrix}\delta\boldsymbol{x}^{\mathrm{T}\dagger} & \mathbf{r}_c & -\delta\boldsymbol{x}^{\mathrm{T}\dagger}\end{bmatrix}\begin{bmatrix}{\mathbf{r_b}}^{\mathrm{T}} \\ \mathbf{v}^{\dagger} \\ \delta\boldsymbol{x}^{\mathrm{T}}\mathbf{r}_c\mathbf{v}^{\dagger}\end{bmatrix} = \mathbf{V}\mathbf{W}^{\mathrm{T}},$$

where $\mathbf{V}$ is an $m$-by-3 matrix and $\mathbf{W}$ is an $2n$-by-3 matrix.

# Preconditioning Saddle Point Formulation of 4D-Var

- Substituting the solution for $\mathbf{S}$ into $\Delta\mathbf{B}$ yields that

$$\boxed{\Delta\mathbf{B}^* = \mathbf{u}_2{}^{\mathrm{T}\dagger}\mathbf{r}_b^{\mathrm{T}} + (\mathbf{I} - \mathbf{u}_2{}^{\mathrm{T}\dagger}\mathbf{u}_2{}^{\mathrm{T}})\mathbf{r}_c\mathbf{u}_1{}^{\dagger}}$$

- This formula can be rewritten as

$$\Delta\mathbf{B}^* = \begin{bmatrix} \delta\boldsymbol{x}^{\mathrm{T}\dagger} & \mathbf{r}_c & -\delta\boldsymbol{x}^{\mathrm{T}\dagger} \end{bmatrix} \begin{bmatrix} \mathbf{r_b}^{\mathrm{T}} \\ \mathbf{v}^{\dagger} \\ \delta\boldsymbol{x}^{\mathrm{T}}\mathbf{r}_c\mathbf{v}^{\dagger} \end{bmatrix} = \mathbf{V}\mathbf{W}^{\mathrm{T}},$$

where $\mathbf{V}$ is an $m$-by-3 matrix and $\mathbf{W}$ is an $2n$-by-3 matrix.

- The preconditioner can be updated by using the following formula

$$\mathcal{P}_1 = \mathcal{P}_0 + \begin{pmatrix} \mathbf{0} & \mathbf{W}\mathbf{V}^T \\ \mathbf{V}\mathbf{W}^T & \mathbf{0} \end{pmatrix} = \mathcal{P}_0 + \underbrace{\begin{pmatrix} \mathbf{0} & \mathbf{W} \\ \mathbf{V} & \mathbf{0} \end{pmatrix}}_{\mathbf{F}} \underbrace{\begin{pmatrix} \mathbf{W}^T & \mathbf{0} \\ \mathbf{0} & \mathbf{V}^T \end{pmatrix}}_{\mathbf{G}}$$

# Preconditioning Saddle Point Formulation of 4D-Var

- The inverse formula is then given by

$$\boxed{\mathcal{P}_F^{-1} = \mathcal{P}_0^{-1} - \mathcal{P}_0^{-1}\mathbf{F}(\mathbf{I}_4 + \mathbf{G}\mathcal{P}_0^{-1}\mathbf{F})^{-1}\mathbf{G}\mathcal{P}_0^{-1}}$$

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

# Preconditioning Saddle Point Formulation of 4D-Var

- The inverse formula is then given by

$$\boxed{\mathcal{P}_F^{-1} = \mathcal{P}_0^{-1} - \mathcal{P}_0^{-1}\mathbf{F}(\mathbf{I}_4 + \mathbf{G}\mathcal{P}_0^{-1}\mathbf{F})^{-1}\mathbf{G}\mathcal{P}_0^{-1}}$$

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

- Let's remember the first formula:

$$\boxed{\mathcal{P}_T^{-1} = \mathcal{P}_0^{-1} - \mathcal{P}_0^{-1}\mathbf{F}(\mathbf{I}_2 + \mathbf{G}\mathcal{P}_0^{-1}\mathbf{F})^{-1}\mathbf{G}\mathcal{P}_0^{-1}}$$

- The least Frobenius norm update is slightly more expensive than the first update however it is more stable.
$\rightarrow$ It can be shown that $\|\mathcal{P}_T^{-1}\|_F$ can be arbitrarily larger than $\|\mathcal{P}_F^{-1}\|_F$