Vertical balance operator implementation in BUMP

Benjamin Menetrier, IRIT, Toulouse, France
(benjamin.menetrier@irit.fr)

JEDI meeting - 12th September 2019

Formulation

The control variable $\mathbf{x}$ is multivariate (e.g. with 3 subvectors):

$$\mathbf{x} = \begin{pmatrix} \mathbf{x}_1 \\ \mathbf{x}_2 \\ \mathbf{x}_3 \end{pmatrix} \tag{1}$$

The vertical balance operator $\mathbf{K}$ transforms the unbalanced control variable $\mathbf{v}$ into $\mathbf{x}$:

$$\mathbf{x} = \mathbf{K}\mathbf{v} \tag{2}$$

$\mathbf{K}$ is generally built as a lower diagonal block matrix, with identity diagonal blocks:

$$\mathbf{K} = \begin{pmatrix} \mathbf{I} & 0 & 0 \\ \mathbf{K}_{2,1} & \mathbf{I} & 0 \\ \mathbf{K}_{3,1} & \mathbf{K}_{3,2} & \mathbf{I} \end{pmatrix} \tag{3}$$

Formulation

The adjoint of K is straightforward:

$$\mathsf{K}^{\mathrm{T}} = \begin{pmatrix} \mathsf{I} & \mathsf{K}_{2,1}^{\mathrm{T}} & \mathsf{K}_{3,1}^{\mathrm{T}} \\ 0 & \mathsf{I} & \mathsf{K}_{3,2}^{\mathrm{T}} \\ 0 & 0 & \mathsf{I} \end{pmatrix} \tag{4}$$

and the inverse of K:

$$\mathbf{v} = \mathsf{K}^{-1}\mathbf{x} \tag{5}$$

can be computed recursively:

$$\mathbf{v}_1 = \mathbf{x}_1 \tag{6}$$

$$\mathbf{v}_2 = \mathbf{x}_2 - \mathsf{K}_{2,1}\mathbf{v}_1 \tag{7}$$

$$\mathbf{v}_3 = \mathbf{x}_3 - \mathsf{K}_{3,1}\mathbf{v}_1 - \mathsf{K}_{3,2}\mathbf{v}_2 \tag{8}$$

The inverse of $\mathsf{K}_{i,j}$ blocks is not required!

Estimation

By definition, $\mathbf{v}$ is *unbalanced*, which means that its subvectors have zero cross-covariances:

$$\operatorname{cov}\left(\mathbf{v}_i, \mathbf{v}_j\right) = \begin{cases} \mathbf{C}_i & \text{if} \quad i = j \\ 0 & \text{if} \quad i \neq j \end{cases} \tag{9}$$

Blocks are estimated successively, each row at a time, and the unbalanced subvectors are computed simultaneously:

- First row: $\mathbf{v}_1 = \mathbf{x}_1$
- Second row:

$$\operatorname{cov}\left(\mathbf{v}_1, \mathbf{v}_2\right) = 0 \tag{10}$$

$$\Leftrightarrow \operatorname{cov}\left(\mathbf{v}_1, \mathbf{x}_2 - \mathbf{K}_{2,1}\mathbf{v}_1\right) = 0 \tag{11}$$

$$\Leftrightarrow \mathbf{K}_{2,1} = \operatorname{cov}\left(\mathbf{v}_1, \mathbf{x}_2\right)\left[\operatorname{cov}\left(\mathbf{v}_1, \mathbf{v}_1\right)\right]^{-1} \tag{12}$$

and

$$\mathbf{v}_2 = \mathbf{x}_2 - \mathbf{K}_{2,1}\mathbf{v}_1 \tag{13}$$

Estimation

Three steps are required to compute $\mathbf{K}_{2,1}$:

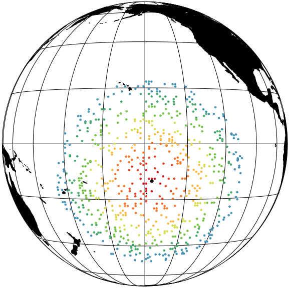
- Step 1: estimate the cross-covariance $\text{cov}(\mathbf{v}_1, \mathbf{x}_2)$ and the auto-covariance $\text{cov}(\mathbf{v}_1, \mathbf{v}_1)$
- Step 2: invert the auto-covariance to get $\left[\text{cov}(\mathbf{v}_1, \mathbf{v}_1)\right]^{-1}$
- Step 3: compute $\mathbf{K}_{2,1} = \text{cov}(\mathbf{v}_1, \mathbf{x}_2) \left[\text{cov}(\mathbf{v}_1, \mathbf{v}_1)\right]^{-1}$

Remarks:

- The nature and order of physical variables in $\mathbf{x}$ is crucial. Several choices are possible (psi/chi/T, vor/div/T, etc.).
- For a *vertical* balance operator, all the matrices are $n \times n$, where n is the number of vertical levels.
- However, it is possible to introduce a horizontal dependency: one specific $\mathbf{K}$ can be computed for each given “bin” (latitude band, local neighborhood, specific area, etc.).

BUMP implementation

- Variables are provided by the user via the **UnstructuredGrid**.
- An ensemble is used to estimate vertical covariances for a subset of columns **Sc1**.
- For another subset **Sc2**, these covariances are averaged locally over a given radius.



- To apply the vertical balance operator at any grid point, regression values of subset **Sc2** are linearly interpolated.
- The subsets can take boundaries and masks into account.
- The auto-covariance inversion uses a Cholesky decomposition.

BUMP implementation

SABER branch **feature/improved_lct_calculation** (PR #24).

Namelist / YAML keys to set:

- **new_vbal** to activate the vertical balance estimation.
- **load_vbal** to load an existing vertical balance operator.
- **write_vbal** to write the vertical balance estimation.
- **check_vbal** to test the operator inverse and adjoint.
- **vbal_block** (vector) to separately activate blocks $\mathbf{K}_{2,1}$, $\mathbf{K}_{3,1}$, $\mathbf{K}_{3,2}$, $\mathbf{K}_{4,1}$, etc.
- **vbal_rad** to set the averaging radius (in m).
- **vbal_diag_auto** (vector) to use a diagonal estimation of the auto-covariance, easily invertible, for given blocks.

Code:

- SABER source code: **saber/bump/type_vbal.F90**
- OOPS interface: **oops/generic/StatsVariableChange.h**

On-going work

To do with the current code:

- Test the scientific relevance of the local neighborhood and interpolation approach.
- Play with the radius/mask possibilities.
- Test the numerical efficiency of the estimation and of the application.

To do in the future:

- Improve the regressions estimation scalability.
- Add an option to get fully diagonal regressions (as for psi/chi in WRFDA).
- Any other idea?