EnsAugm

Ensemble Augmentation by Schur products with large-scale patterns

User's guide

Jean-Michel Brankart

http://pp.ige-grenoble.fr/pageperso/brankarj/

Institut des Géosciences de l'Environnement
Université Grenoble Alpes, CNRS, France

The purpose of EnsAugm is to provide tools to augment an ensemble simulation with new members. The objective is to mitigate the effects of undersampling by an artifical increase of the ensemble size.

All variables from the input ensemble are assumed to be normally distributed with zero mean and unit standard deviation [N(0,1)], which can be obtained by anamorphosis transformation (see EnsAnam modules). The new members can then be obtained by combining Schur products of one of the ensemble member by the large-scale pattern of other members. This operation preserves the local correlation structure of the original ensemble and reduce to zero the long-range correlations.

The tools are provided as a library of modules, which can be easily plugged in any existing software. This library includes:

- the computation of the Schur product of two vectors with N(0,1) marginal distribution;
- the computation of a multiple Schur product by random combination of ensemble members at different scales;
- the sampling of the augmented ensemble.

1 Description of the method

A major difficulty with ensemble methods is that large ensembles are expensive to produce, while the accuracy of the statistics improves quite slowly with the ensemble size. Methods to artificially increase the ensemble size at low numerical cost can thus be very helpful, providing that the statistics can be improved. The approach that is used here to generate an augmented ensemble with imrpoved statistics is to localize the correlation structure of the original ensemble. The localization of correlations is done implicitly (in the sense that correlations are never computed explicitly in the algorithm) by computing Schur products of ensemble members with large-scale patterns of other members.

Our basic assumptions about the original ensemble are that:

- the marginal probability distribution of all variables is a normal distribution with zero mean and unit standard deviation (which can be obtained by anamorphosis transformation, see EnsAnam modules);
- the large-scale component of every member of the ensemble can be computed by separating scales (using for instance the TranSpHO modules, in the case of spherical coordinates); each of them is also renormalized to have a zero mean and a unit standard deviation.

The characteristics of the localizing correlation matrix depend on the scales that are kept in the large-scale components provided to the algorithm.

1.1 Computation of Schur products

Let $\mathbf{x_1}$ and $\mathbf{x_2}$ be two independent random vectors, with:

$$< \mathbf{x_1} >= 0 \qquad \text{and} \qquad < \mathbf{x_1}{\mathbf{x_1}}^T >= \mathbf{C}_1 \tag{1}$$

$$< \mathbf{x_2} >= 0 \qquad \text{and} \qquad < \mathbf{x_2}{\mathbf{x_2}}^T >= \mathbf{C}_2 \tag{2}$$

and let $\mathbf{x}$ be the Schur product of $\mathbf{x_1}$ and $\mathbf{x_2}$:

$$\mathbf{x} = \mathbf{x_1} \circ \mathbf{x_2} \tag{3}$$

Then, it can be shown that:

$$< \mathbf{x} >= 0 \quad \text{and} \quad < \mathbf{x}\mathbf{x}^T >= \mathbf{C}_1 \circ \mathbf{C}_2 \tag{4}$$

This is the basic property used in the augmentation algorithm to localize correlation using Schur products.

One possible difficulty with this method is that the product of two independent Gaussian variable is not a Gaussian variable. If necessary, this problem can be solved since all components of $\mathbf{x_1}$ and $\mathbf{x_2}$ have been assumed to have the same marginal distribution:

$$x_1^{(k)},\, x_2^{(k)} \;\sim\; \mathcal{N}(0,1) \quad \forall k \tag{5}$$

so that all components of $\mathbf{x}$ have also the same marginal distribution:

$$x^{(k)} = x_1^{(k)}\, x_2^{(k)} \;\sim\; \mathcal{P}(0,1) \quad \forall k \tag{6}$$

where $\mathcal{P}(0,1)$ is the distribution of the product of two independent Gaussian variables with zero mean and unit variance. The pdf of this distribution can be computed explicitly (Nadarajah and Pogány, 2015):

$$p(x) = \frac{1}{\pi} K_0(|x|) \tag{7}$$

where K_0 is the order 0 modified Bessel function of the second kind. The x variable can then be easily transformed back to Gaussian by the function:

$$\tilde{x} = G^{-1}\left[P(x)\right] \tag{8}$$

where $P(x)$ is the cdf corresponding to $\mathcal{P}(0,1)$ and $G(\tilde{x})$ is the cdf corresponding to $\mathcal{N}(0,1)$.

The computation of the Schur product with Eq. (3) and the renormalization with Eq. (8) can be peerformed with the module `schurprod` of the library. The renormalization is performed in the module using an existing implementation of P, which is available online[1], following the algorithm proposed by Meeker and Escobar (1994).

1.2 Localizing correlations by Schur products with large-scale patterns

Let us suppose that every ensemble member $\mathbf{x}_i$, $i = 1, \ldots, m$ (where m is the size of the original ensemble) is associated to its corresponding large-scale component $\mathbf{x}_i^{(j)}$, for several cutting wave lengths $j = 2, \ldots, s$ (where s is the number of available scales for each ensemble member, including the full-scale member itself for $s = 1$). Then, we can construct multiple Schur products like:

$$\mathbf{x} = \mathbf{x}_\alpha \circ \left(\mathbf{x}_\beta^{(2)} \circ \ldots \circ \mathbf{x}_\gamma^{(2)}\right) \circ \ldots \circ \left(\mathbf{x}_\psi^{(s)} \circ \ldots \circ \mathbf{x}_\omega^{(s)}\right) \tag{9}$$

combining one member of the original ensemble with several members at each available scale. In computing this product, it is assumed that the member indices $\alpha, \beta, \ldots \gamma \ldots \psi \ldots \omega$ are all different so that the same member is never used twice in the same product.

The covariance of $\mathbf{x}$ is then:

$$< \mathbf{x}\mathbf{x}^T >= \mathbf{C} \circ \left(\mathbf{C}^{(2)} \circ \ldots \circ \mathbf{C}^{(2)}\right) \circ \ldots \circ \left(\mathbf{C}^{(s)} \circ \ldots \circ \mathbf{C}^{(s)}\right) \tag{10}$$

where $\mathbf{C}$ is the correlation matrix of the original ensemble, and the rest of the product is the localizing correlation. One important condition on the localizing correlation matrix is that all elements must be positive (to avoid changing the sign of correlation coefficients in $\mathbf{C}$). In Eq. (10), this condition is easily verified by using an even Schur-power for each of the $\mathbf{C}^{(j)}$, $j = 2, \ldots, s$. In this way, by using an even number of vector in each parenthesis of the product in Eq. (9), we can be sure that we (implicitly) localize the ensemble covariance with a positive correlation matrix. In addition to this, as in the previous section, it is also possible to take care of renormalizing the product so that all variables in $\mathbf{x}$ are still marginally distributed as $\mathcal{N}(0,1)$. However, it must be noted that this operation can modify the linear correlation structure, even if it preserves rank correlations.

The key property of Eq. (9) for augmenting the original ensemble is the very large number of linearly independent $\mathbf{x}$-vector that can be generated by different combinations of the original members. With p Schur products (i.e. by combining p large-scale patterns to one original member), the number of possible combinations is:

$$N = \frac{m!}{(m-p-1)! \prod_i q_i!} \qquad \text{with} \qquad p \leq m-1 \qquad \text{and} \qquad \sum_i q_i = p \tag{11}$$

where m is the size of the original ensemble, q_i is the multiplicity of every scale in the product, and N is the number of products that can be generated. For instance, for $m = 20$, if the positive condition must be verified, the maximum number of $\mathbf{x}$-vectors that can be generated is

[1] https://jblevins.org/mirror/amiller/

by setting $p = 18$ and all q_i ($i = 1, \ldots, 9$) equal to 2, so that the maximum size of the augmented ensemble is as large as $20!/2^9 \simeq 5 \times 10^{15}$. More reasonably, if the size of the state vector is about $n \sim 10^7 - 10^8$, a full rank ensemble could already be obtained with $p = 6$ Schur products, which leads to a possible ensemble size as large as $20!/(13! \times 2^3) \simeq 5 \times 10^7$. Could we explore the state space by linear combination of these vectors, we would be able to solve inverse problems globally without rank approximation.

1.3 Sampling of the augmented ensemble

From the Schur products in Eq. (9), members of the augmented ensemble can then be obtained by random linear combinations:

$$\hat{\mathbf{x}}_i = \frac{1}{\sqrt{N}} \sum_{k=1}^{N} w_{ik}\, \mathbf{x}_{\alpha(k)} \tag{12}$$

where $\mathbf{x}_{\alpha(k)}$ is the Schur product obtained with combination $\alpha(k)$ of one original member and several large-scale patterns, w_{ik} are independent random coefficients with $\mathcal{N}(0,1)$ distribution, and $\hat{\mathbf{x}}_i$ is one member of the augmented ensemble. By construction, the marginal distribution of the $\hat{\mathbf{x}}_i$ is $\mathcal{N}(0,1)$, at the only condition that the variance of the $\mathbf{x}_{\alpha(k)}$ is equal to 1, which follows directly from Eq. (9). In this case, the renormalization of the $\mathbf{x}_{\alpha(k)}$ using Eq. (8) is thus unnecessary.

In practice, the sum in Eq. (12) is computed iteratively, as the result of the sequence:

$$\hat{\mathbf{x}}_i^{(0)} = 0\,; \quad \hat{\mathbf{x}}_i^{(k+1)} = \frac{\sqrt{k-1}}{\sqrt{k}}\, \hat{\mathbf{x}}_i^{(k)} + \frac{1}{\sqrt{k}}\, w_{ik}\, \mathbf{x}_{\alpha(k)} \tag{13}$$

In this way, the augmented members are constructed progressively with more and more Schur products, and can be viewed as an ensemble of MCMC chains converging towards the requested sample.

2 Description of the modules

In this section, the modules are described one by one, giving for each of them: the method that has been implemented, the list of public variables and public routines (with a description of input and output data), the MPI parallelization, and an estimation of the computational cost as a function of the size of the problem.

2.1 Module: schurprod

The purpose of this module is to perform the Schur product of two vectors/variables with N(0,1) marginal distribution and restore a marginal N(0,1) distribution.

Method

The method is to loop on the vector variables, compute the product, and renormalize the result using the transformation in Eq. 8. The transformation is either computed explicitly, or interpolated from a precomputed table. Interpolation is less expensive if the size of the vector is larger than the size of the precomputed table.

Public variables

schurprod_precompute: precompute the renormalization function (default=.TRUE.);

schurprod_gpmin: min Gaussian product in precomputed table (default=-12.);

schurprod_gpmax: max Gaussian product in precomputed table (default=12.);

schurprod_tablesize: size of precomputed table (default=10000).

Public routines

schurprod: perform Schur product:

`prod` **(input/output)** : Schur product (overwritten on 1st input vector/variable);

`vct` **(input)** : second input vector/variable with which performing the Schur product.

MPI parallelization

For Schur product of large-size vectors, MPI parallelization is easily obtained by making each processor work on a different part of the vectors. Since the operations performed on different variables are independent, no MPI operations needed to be implemented inside the modules.

Computational cost

The computational complexity of the algorithm can be written:

$$C_p \sim k_1 n \tag{14}$$

where n is the size of the input vectors, and k_1 is an order 1 constant. The Schur product itself is just one mutliplication for each component of the input vector, while renormalization is reduced to a few operations by interpolating in a precomputed table (4 additions, 2 multiplications, and 1 division). In total, this means that $k_1 \sim 4$.

2.2 Module: ensaugm

The purpose of this module is to generate new ensemble members by Schur product with large scale patterns.

Method

The method is to sample random combinations of members to compute the multiple Schur product and then to perform random linear combination of these multiple Schur products to build augmented members. This random linear combination is performed iteratively, by generating the multiple Schur products one by one, and add progressively their contribution to the random linear combination. The Schur product can optionally be renormalized using the `schurprod` module.

Public variables

ensaugm_with_renormalization: Renormalize the Schur product with the `schurprod` module (default=.FALSE.);

ensaugm_chain_index: Current iteration index in MCMC chain (intialized to 1);

mpi_comm_ensaugm: MPI communicator to use (default=mpi_comm_world).

Public routines

sample_augmented_ensemble: iterate the Markov chains to sample the augmented ensemble:

- `maxchain` **(input)** : number of additional iterations to perform;
- `augens` **(input/output)** : current version of augmented ensemble;
- `ens` **(input)** : input ensemble to be augmented, assumed available at several resolutions (the first one at full resolution, and the next ones at lower resolution), all marginal distributions must be N(0,1);
- `multiplicity` **(input)** : multiplicity of each resolution in the construction of the new member;

newproduct: compute new multiple Schur product (with random combination of input members):

- `new` **(output)** : new multiple Schur product;
- `ens` **(input)** : ensemble to be augmented, at full resolution and at several lower resolutions;
- `multiplicity` **(input)** : multiplicity of each resolution in the construction of the new member; and at several lower resolutions;
- `sample` **(output)** : member indices that have been used to build the new multiple Schur product.

getproduct: compute new multiple Schur product (with specified combination of input members):

- `new` **(output)** : new multiple Schur product;
- `ens` **(input)** : ensemble to be augmented, at full resolution and at several lower resolutions;
- `multiplicity` **(input)** : multiplicity of each resolution in the construction of the new member;
- `sample` **(input)** : member indices that must be used to build the new multiple Schur product.

MPI parallelization

For large-size vectors, MPI parallelization is easily obtained by making each processor work on a different part of the vectors. MPI communications are only needed to share the random member indices that are used to compute the multiple Schur products, and the random coefficients that are used to linearly combine these multiple Schur products.

Computational cost

The computational complexity of the computation of one multiple product is:

$$C_{mp} = p \times C_p \sim k_1 pn \tag{15}$$

where n is the size of the input vectors, p, the number of vectors used in each multiple product (this is the sum of the multiplicities used for each resolution), and $k_1 = 1$ (without renormalization) or $k_1 \sim 4$ (with renormalization).

The computational complexity of the sampling of the augmented ensemble can be written:

$$C_{\text{aug}} = Nm \times C_{mp} \sim k_1 pnmN \tag{16}$$

where m is the size of the augmented ensemble (the number of required Markov chains), and N, the number of iteration in the chains (i.e. the number of multiple products that are combined to build a new member).

References

Meeker, W.Q. and Escobar, L.A., 1994: An algorithm to compute the CDF of the product of two normal random variables. *Commun. in Statist.-Simula.*, **23(1)**, 271-280.

Nadarajah, S. and Pogány, T., 2015: On the distribution of the product of correlated normal random variables. *Comptes Rendus Mathematique*, **354**, 10.1016/j.crma.2015.10.019.