SANGOMA: Stochastic Assimilation for the Next Generation Ocean Model Applications

EU FP7 SPACE-2011-1 project 283580

Deliverable 1.1: First identification of common tools

Due date: 01/2/2012 Delivery date: 01/2/2012 Delivery type: Report , public

Arnold Heemink Nils van Velzen Martin Verlaan Umer Altaf
Delft University of Technology, NETHERLANDS

Jean-Marie Beckers Alexander Barth
University of Liège, BELGIUM

Peter Jan Van Leeuwen
University of Reading, UK

Lars Nerger
Alfred-Wegener-Institut, GERMANY

Pierre Brasseur Jean-Michel Brankart
CNRS-LEGI, FRANCE

Pierre de Mey
CNRS-LEGOS, FRANCE

Laurent Bertino
NERSC, NORWAY

Table of Contents

1 Introduction

This document gives an initial list of software components that are eligible to be shared in SANGOMA. The modules that are mentioned in this document only form an initial set, a starting point for some of the initial activities in the SANGOMA project. Later, using the experience gathered in the collaboration between the partners and using new insights we will extend this list. The extended (most up to date) list of the modules will be part of the living document (DL1.2). The reader should therefore skip the present document and directly read DL1.2 if it is available.

There are significant differences in the design of the various data assimilation systems that have been developed and used by the various partners. However, this does not necessarily mean that we cannot reuse parts of the existing systems or build new components that can be used in combination with the various data assimilation systems. In order to be able to realize components that can be shared, we must keep the various designs of the systems in mind. This document therefore does not only present a list of modules but also discusses a number of other important issues like the impact of programming languages and differences between the various assimilation systems as well.

The data assimilation systems in use contain various kind of tools ranging from simple methods to more complicated ones suitable to various applications and ease of use. The list of tools for SANGOMA are selected mainly considering the following points.

- Reusability of the existing software components and share the available code.
- The tools should be of sufficient complexity, otherwise one can develop and use it easily.
- The tools should provide nonlinear aspects of data assimilation.

The structure of this document is as follows; The first section discusses the strategy already presented at the kick off meeting in Nov 2011 at Liège. The aspect of various programming languages and how they can be mixed is then discussed. This follows an explanation on components design so that they can be shared. After discussing these general design aspects of the modules we present the initial list of components as suggested by the partners. For later reference we have added the list of possible modules from the proposal as well. To conclude, we will summarize this document and list our recommendations.

2 Strategy

At the SANGOMA kick off meeting on 24 - 25 Nov 2011 at Liège we (TU-Delft) have presented our initial thought on how we can develop common components. As agreed during the said meeting we have compiled an on-line document that can be edited by all partners. This shared working document is used to gather the necessary information from all the partners. The main advantage of a shared document is that all partners can see and benefit from each others' contributions. The shared document contains templates. Using these templates, the partners could provide us the necessary information on

1. the software components they would like to have on the initial list and
2. the description of the model(s) they use.

The feedback on the components is used to compile this document. The model description acts as a starting point of a common data model. The design of a common data model is beyond the scope of this document but is necessary to make the components interchangeable. Prototypes of the data model can be found on the SANGOMA page on Sourceforge (http://sourceforge.net/projects/sangoma). The first official version will be described in another document, to be released in May 2012.

3 Overview of the current systems

In this section we give a brief overview of the data-assimilation systems that are developed and or used by the partners. We will only discuss important general properties of the systems that are relevant for designing components that can be used in the various systems.

3.1 PDAF

Description: The Parallel Data Assimilation Framework - PDAF - is a software environment for ensemble data assimilation. PDAF simplifies the implementation of the data assimilation system with existing numerical models. With this, users can obtain a data assimilation system with less work and can focus on applying data assimilation.
PDAF provides fully implemented and optimized data assimilation algorithms, in particular ensemble-based Kalman filters like LETKF and LSEIK. It allows users to easily test different assimilation algorithms and observations. The Framework is optimized for the application with large-scale models that usually run on big parallel computers and is applicable for operational applications. However, it is also well suited for smaller models and even toy models.
PDAF provides a standardized interface that separates the numerical model from the assimilation routines. This allows to perform the further development of the assimilation methods and the model independently. New developments on the algorithmic side can be readily made available through the interface such that they can be immediately applied with existing implementations. The test suite of PDAF provides small models for easy testing of algorithmic developments and for teaching data assimilation.

PDAF is an open-source project. Its functionality will be further extended by input from research projects. In addition, users are welcome to contribute to the further enhancement of PDAF, e.g. by contributing further assimilation methods or interface routines for different numerical models.

License: Open source (LGPL)
Parallelization: Parallelized using MPI for the ensemble integration, the model time-steps including domain decomposition, and the filter analysis.
Programming language: Fortran90
Coupling with model: Both black-box (using files) as in memory (for optimal performance). When the in memory coupling is used, the user must provide a number of support routines with a predefined interface and extend the model code with a few calls to PDAF routines.
Further information: http://pdaf.awi.de

3.2 OpenDA

Description: OpenDA is an open interface standard for (and free implementation of) a set of tools to quickly implement data-assimilation and calibration for arbitrary numerical models. OpenDA wants to stimulate the use of data-assimilation and calibration by lowering the implementation costs and enhancing the exchange of software among researchers and end-users.
A model that conforms to the OpenDA standard can use all the tools that are available in OpenDA. This allows experimentation with data-assimilation/calibration methods without the need for extensive programming. Reversely, developers of data-assimilation/calibration software that make their implementations compatible with the OpenDA interface will make their new methods usable for all OpenDA users (either for free or on a commercial basis).
OpenDA has been designed for high performance. Hence, even large-scale models can use it. Also, OpenDA allows users to optimize the interaction between their model and the data-assimilation/calibration methods. Hence, data-assimilation with OpenDA can be as efficient as with custom-made implementations of data-assimilation methods.
OpenDA is an Open Source project. Contributions are welcome from anyone wishing to participate in the further development of the OpenDA tool-set.

License: Open source (LGPL)
Parallelization: The model time-steps including domain decomposition.
Programming language: Java, C (for the HPC computational core), interfacing to Fortran90 or black-box. The algorithms are all implemented in java and use an object-oriented approach, without direct access to the underlying data.
Coupling with model: Both black box (using files) as in memory (for optimal performance). For the in memory coupling the programmer must implement a set of routines (in Java, C or Fortran) and compile these routines in a jar file (Java) or dynamic library (C, Fortran).
Further information: http://www.openda.org

3.3 Beluga/Sequoia

Description: SEQUOIA(Sequential Optimization, and Analysis Initialization) is a modular assimilation system builder developed by LEGOS and disseminated via the SIROCCO French national service. BELUGA is a 4D/local Ensemble Kalman Filter scheme built with SEQUOIA.
In addition an implementation by today's standards (Fortran95, modularity, etc..), SEQUOIA adapts equally well to structured grids (finite difference) as to finite elements / finite volume via a system of generalized grid. Its streamlined interface with the digital model allows it to control any model. Currently, SYMPHONY, and MOG2D/T-UGOm POLCOMS are interfaced, and other models are considered. The code also manages all of the simulations on cluster of PCs or remote machine.
The system of interchangeable core analysis allows to implement various assimilation schemes. These cores are available as part of SEQUOIA. Three analysis cores have been developed to date for SEQUOIA:

1. Mantaray, a core EOFs 3-D, close to SEEK (used in the thesis of G. and J. Jordà Lamouroux and several post-docs)
2. SOFA, a core of EOFs 1-D, which incorporates the ideas of the original code SOFA mentioned above (used in industrial partnerships and operational)
3. BELUGA, a core of full rank solved in the dual space.

Typical use cases for SEQUOIA are Ensemble Optimal Interpolation (EnOI) - cut to order with nuclei analysis Mantaray or SOFA, Ensemble Kalman Filter (EnKF) and Ensemble Kalman Smoother (EnKS) with the core analysis BELUGA.
SEQUOIA is a complete data-assimilation package, that is used both in research and for operational forecasting. It has been used extensively within European projects (MERSEA, MFS, ECOOP, MyOcean, GODAE OceanView, GOCEAN ESA) and operational industrial partnerships.

Licence: SEQUOIA :Open Source (GPL/LGPL?)
Parallelization: Scheduling of parallel runs in an ensemble
Programming language: Fortran95
Coupling with model: Both in-memory and black-box coupling is supported. Black-box coupling uses a dedicated standardized format.
Further information: http://sirocco.omp.obs-mip.fr/outils/Sequoia/Accueil/SequoiaAccueil.htm

3.4 SESAM

Description: The purpose of the SESAM code (developed in the MEOM group at LEGI) is to perform the various basic operations that are required in sequential data-assimilation systems.

These operations include square root and ensemble observational updates (with global or local parametrization of the forecast error statistics), adaptive statistical parametrizations, anamorphosis transformations, or the computation of truncated Gaussian estimators.
SESAM also provides diagnostic tools, to compute observation representers, EOF decompositions or regional RMS misfits, and various utilities for extracting observations, converting between file formats or performing simple algebraic operations.

License: Open source dedicated ‘CeCill licence’. The license claims compatibility with GPL. Since only block-box coupling is supported no linking with model-code occurs.
Parallelization: Model time-steps can be executed in parallel.
Programming language: Fortran90 (note: building blocks are coupled using NetCDF files)
Coupling with model: Black box coupling using NetCDF files.
Further information: http://www-meom.hmg.inpg.fr/Web/Outils/SESAM/

3.5 NERSC EnKF

Description: The NERSC repository contains two separate computer EnKF codes. (1) A Matlab toolbox, which is both a research tool (used for 4-5 publications) and an initiation tool for junior and senior scientists to get acquainted with the EnKF, its different flavours and technical settings. The package contains several stochastic and deterministic schemes of the EnKF, different approaches for localization and toy models of increasing non- linearity and dimensions. The source code is open, documented and kept up-to-date regularly. (2) A Fortran 90 implementation of the EnKF as used operationally in the TOPAZ system. This code is not fully generic as both the interface and parallelization are dependent on the HYCOM output file structure but can be adapted by a skilled scientist to a different model (This has been done at UCL, Belgium, for example). The code source is open and the upgrades are visible via a user-friendly Subversion (SVN) interface. Classical diagnostics such as the innovations, ensemble spread, Degrees of Freedom of Signal (DFS) and Spread Reduction Factor (SRF) are produced in NetCDF format for easy monitoring of real-time forecasting and long reanalysis runs. An Ensemble OI (EnOI) code is served in addition, with light modifications from the main EnKF code.

License: Open source (no licence specified).
Parallelization: Manual parallelization of the model time-steps. MPI parallelization for fortran analysis code. No parallel computation supported for Matlab code.
Programming language: Fortran90 / Matlab.
Coupling with model: Black box coupling, input and output files according to dedicated binary format.
Further information: http://enkf.nersc.no

3.6 Ocean Assimilation Kit (OAK)

Description: The Ocean Assimilation Kit (OAK) provides a modular toolbox for data-assimilation mainly aimed at oceanographic applications. The definition of state variables is very flexible in OAK. By means of easy configuration files the user can select arbitrary variables from NetCDF files. Curvilinear grids are supported as well. In addition OAK provides global and local versions of the analysis.

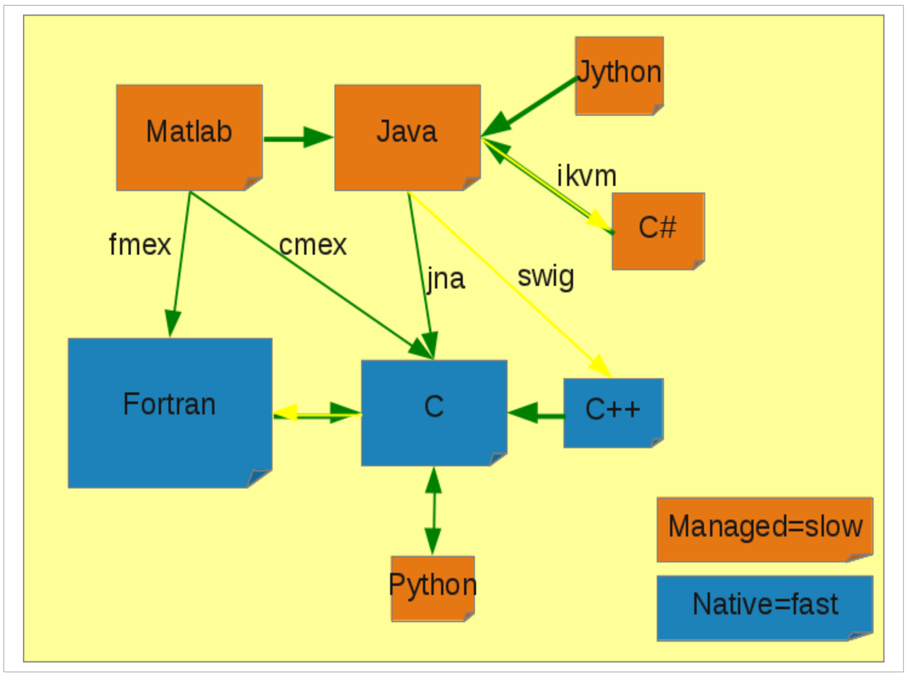
License: GPL
Parallelization: Parallel analysis with OpenMP or MPI. Time-stepping can easily be implemented in parallel by the user.
Programming language: Fortran90
Coupling with model: Black box, input and output files in NetCDF format.
Further information: http://modb.oce.ulg.ac.be/mediawiki/index.php/Ocean_Assimilation_Kit

4 Programming languages

The data assimilation systems of the partners are programmed using various programming languages. In order to use common components of various systems it is necessary to sort out the hurdles related to the programming languages. Many modern compilers and languages support coupling to other languages, but there are many constraints. On the other hand, many methods used for coupling languages are not much portable, thus requires additional efforts to use the code on another platform. Although there are many details that limit interoperability of languages, there are a few generic guidelines:

- Compiled languages are in most cases significantly faster. The execution time of these programs is therefore in general lower. Thus, the use of interpreted code within compiled code often results in deterioration of the performance.
- It is often more difficult to call interpreted code from compiled code than the other way around.
- Mixed interpreted and compiled code is often more difficult to debug.
- C-code can be included in most other languages. Since C is the language used by most operating systems, calling to system libraries (in C) is often one of the earliest connections to be supported.
- The compiled code from Fortran is different between compilers, even on the same platform. Although, one can use some tricks to make the program work for one specific case, portable solutions require a toolbox or use of the Fortran2003 standard.
- Fortran code compiled with different compilers often does not mix well because of the use of different run-time libraries, data representation and calling conventions.
- One should make certain that required libraries of different components do not create conflicts.
- Higher level interpreted languages allow for significantly faster prototyping than compiled lower-level languages.

The following diagram gives an overview of languages that are easy to couple. Green arrow show the best supported coupling. Note that the coupling is directional, from the main program to the library.



Based on interoperability, the C language would be the most likely candidate for implementation of the main computational routines. However, most of the existing code within the SANGOMA project is written in Fortran and only one of the partners uses C extensively. Also most modules proposed within SANGOMA focus on numerical computation and Fortran is a convenient language for this. In addition, most of the numerical models used by the SANGOMA partners are implemented in Fortran. Therefore, we propose the use of Fortran for main computational routines in SANGOMA project.

For prototyping Matlab/Octave is the most common language used by SANGOMA participants. With some efforts, most of the code can be written in such a way that it works on both Matlab and Octave without modifications. Use of another language with similar capabilities, would require much efforts and only provides few advantages. We therefore recommend to select Matlab/Octave for prototyping within the SANGOMA project.

5 Coupling of components

Sharing of software components is not trivial. For example, procedures need to use similar data structures for input and output. If no common design exists, one typically needs to write intermediate conversion routines, often called wrappers. Use of toolboxes can be convenient if they are shared between components, but may equally require additional work if different toolboxes are used in different parts of the software. In the worst case reuse of existing software may require a redesign of part of the software, which typically needs a major effort.

There are two main approaches for sharing/reusing software components in multiple environments:

1. Object oriented. The representation of the data is hidden and the data cannot be directly accessed. The software components have a well defined interface, a list of methods/functions to perform tasks on the state, content of the hidden data, of the component.
2. Data oriented. The data representation is strictly defined. The component provides a number of functions/subroutines to operate on the data. The interface of these functions is not strictly defined, but the data representation is.

5.1 Sharing data

Software components can be reused in the form of executable that perform a particular task. Such an executable will in general read a set of (well defined) input files, perform the operations on the input data, and finally write (well defined) output files. Some of the software systems of the partners are set up in this way. The main advantages of this approach are that limited programming skills are needed to alter the sequence of operations and implementing parallelization of model time steps is relatively easy. A disadvantage is that reading and writing large amounts of data is relatively slow and can form a sequential bottleneck for parallel computations.
Exchange of data between the computational nodes can be easily performed with the use of a shared storage, e.g. NFS or Samba. In addition high-performance alternatives are available, when required. NetCDF is by far the most common file-format for exchange of larger data-sets in oceanography. It is well supported by MyOcean and the SANGOMA partners have enough experience in using it. We therefore recommend the use of NetCDF, with CF standards within SANGOMA.

An alternative method to exchange data is to compile the separate components into one program. This allows the components to share data using the same memory, which is much faster than exchanging data through files. On the other hand, rewriting modules so that they can be compiled into one program often requires more efforts to create and more knowledge to fix.

A third alternative to share data between modules, is the use of network communication. Although network communication is a common solution for bridging the language barrier in software engineering, it is less common in high-performance computing. For high-performance computing MPI is the most popular tool for network communication, but has little support for language bridging. Also coupling modules through MPI is often harder than compiling them into into one program. Thus MPI should, in our opinion be viewed as a tool for parallel computing, not for coupling components.

Considering these arguments, the available software and expertise by SANGOMA partners, we recommend a dual approach. In this approach the modules are made available both as subroutines that share a common data-model as well as stand-alone programs that use NetCDF files for input and output. When sharing a common data-model, the IO to and from files can implemented easily in a generic manner, so the additional effort per module becomes small.

5.2 Software licensing

The SANGOMA partners have decided from the start to make the common modules available as an open-source. Contrary to common belief, an open-source software does need a software license, for example to limit the liability or to grant users free access to derived works. Unfortunately, open-source licenses do need some attention when using or creating open-source software:

1. Software licenses can be incompatible. For example, it may be illegal to use or distribute multiple components in one program or download.
2. Some open-source licenses do not allow commercial use.
3. Some open-source licenses require publication of a notice, either giving credit to the original developers or passing on the license to new users.
4. Some open-source licenses were written with little legal advice and may not be very useful in court.

For SANGOMA, we recommend the use of the popular LGPL (http://www.gnu.org/licenses/lgpl.html) This license has been tested in court, and is compatible with many other licenses. LGPL is in some respects similar to GPL, but differs as it allows the compilation of the code with other licenses. Participants of SANGOMA with code presently published as GPL and own the copyright of the code can contribute this code under LGPL to SANGOMA. GPL code that is not owned by the contributor cannot be used.

6 Initial list of common components

In this section we present the initial list of common components that are suggested by the partners. This list only represents a starting selection of modules that will be realized on SANGOMA. From a conceptual point of view pre-processing and post-processing modules are the most system independent and therefore natural candidates to be shared.

In short, the SANGOMA partners have suggested the following modules:

- TUD: POD calibration tool.
- CNRS-LEGOS: ArM, ArM-CA.
- AWI: compute_histogram, compute_stats, eofcovar, mvnormalize, adaptive_forget.
- ULg:Observation operator for High-Frequency Radar measurements.
- CNRS-LEGI: geof, lroa, adap, tgop, anam.

These modules will be described in more detail in the following sections:

6.1 POD calibration tool

- <u>Description of functionality/purpose</u>: Variational method for the calibration of dynamical models. This method does not require an adjoint of the dynamical model. The method uses an approximated adjoint, that is computed only using the foreward dynamical model. Proper Orthogonal Decomposition POD or Balanced Proper Orthogonal Decomposition (BPOD) is used for the approximation. references:
    - M. U. Altaf, A. W. Heemink, and M. Verlaan, *Inverse shallow-water flow modelling using model reduction*, International Journal for Multiscale Computational Engineering, 7 (2009), pp. 577–596.
    - M. U. Altaf, *Model Reduced Variational Data Assimilation for Shallow Water Flow Models,* PhD Thesis. Delft University of Technology (2011).
- <u>Inputs:</u>
    - Set of snapshots of the model states. Each snapshot is a collection of model states at various time instances.
    - Information of the model state that enables us to distinct various quantities in the state vector.
    - Object function formulation

- <u>Reqired work</u>: The module is programmed as a result of a PhD project. A partial rewrite is probably needed.
- <u>Language</u>: F90
- <u>Needs</u>: OpenDA
- <u>Outputs:</u> Optimal solution of the reduces system (parameters)
- <u>Host:</u> TU-Delft

6.2 Observation operator for High-Frequency Radar measurements

- Name of the Module: not decided
- Description of functionality/purpose: this module extracts the radial surface currents from a model state with horizontal scales that are comparable to measurements from a high-frequency radar site.
- Inputs: surface currents
- Outputs: radial currents
- Reqired work: The module needs to be programmed
- Language: ?
- Needs: ?
- Outputs: Optimal solution of the reduces system (parameters)
- Host: ULg/GHER

6.3 Weakly Constrained Ensembles (WCE)

- Purpose: This module creates ensemble perturbations that have to satisfy an a priori linear constraint. It can also be used to create perturbations that are aware of the land-sea mask or that use space- (or time-) dependent correlation length.
- Input: geometry of the domain, correlation length and other fields depending of the nature of the constraints (e.g. velocity field for advection constraint)
- Output: ensemble of constrained perturbations and eigenvalue decomposition of underlying covariance matrix.
- Required work: adaptation of programming interfaces
- Language: Matlab/GNU Octave
- Needs: No additional libraries are required (but NetCDF is recommended)
- Comments: May require data structures larger than 2 GB and may thus require the non-default configure options --enable-64 for GNU Octave.
- Host: ULg

6.4 Gaussian Anamorphosis

- Purpose: Applies a non-linear empirical transformation to a random variables such that the pdf of the transformed variables is closer to a Gaussian distribution
- Inputs: sample of the original variable
- Outputs: transformation function
- Required work: To be seen...
- Language: Matlab/GNU Octave
- Needs: No additional libraries are required
- Comments: No parallelization
- Host: ULg

6.5 ArM

- Description of functionality/purpose :
  - Purpose: Assess the performance of space-time observational arrays at detecting forecast error (as in Le Hénaff, De Mey & Marsaleix, O.Dyn. 59(1), 2009), assess the performance of DA scheme at extracting information from observations
  - Functionality: Representer Matrix spectral analysis, calculation of Array modes, their associated EV spectrum and Modal representers (representers of Array modes in state space)
  - Method: svd(1/sqrt(m-1) * R**(-1/2) * H * Af), where m is the Ensemble size, R is the (not necessarily diagonal) observational error covariance matrix, H is the observation operator for all observations occurring during the current forecast (4D), and Af contains the augmented forecast Ensemble anomalies (4D). The H * Af product is presently implemented here as Bf, the forecast Ensemble anomalies in data space (which has the advantage of being 4D for a modest storage cost, the elements of Bf being collected along the way) – Let us know whether people would favor one or the other formulation.
  - Calling policy: forecast time (end of forecast cycle)
  - Module dependencies: TBD
  - Language: F95
  - Needs: BLAS/LAPACK
- Inputs : 4D forecast Ensemble anomalies in state or data space (TBD), 4D obs. operator (TBD), 4D obs. error cov. matrix or its inverse, square root, or sqrt(inverse)), vector of parameters (number of desired modes, etc.)
- Outputs : eigenspectrum, array modes (data space), modal representers (state space), error code
- Host CNRS-LEGOS

6.6 ArM-CA

- Description of functionality/purpose :
  - Purpose: Array-space consistency analysis
  - Functionality: Project problem of statistical consistency between innovation and Ensemble statistics onto Array-space along Array-mode basis; implement systematic consistency criterion for each rank; provide an overall consistency criterion
  - Method: TBD
  - Module dependencies: TBD
  - Calling policy: forecast time (end of forecast cycle)
  - Language: F95
  - Needs: BLAS/LAPACK

- Module input : TBD, but most likely: 4D forecast Ensemble anomalies in state or data space, 4D obs. operator, 4D obs. error cov. matrix, 4D innovation, array modes and eigenspectrum, vector of parameters (number of desired ranks, etc.)
- Module output : TBD
- Host CNRS-LEGOS

6.7 compute_histogram

- Description of functionality/purpose : Compute rank histogram of an ensemble about some state (e.g. ensemble mean or true state). Computation is done for a single location. It increments the information stored in a histogram array.
- Inputs: Ensemble values for a single grid point. Single state entry. Size of ensemble. Histogram array (size ensemble size+1. It has to be initialized to zero before the first call).
- Outputs: Histogram array
- Required work: The module needs to be adapted to be generally usable. The input/output is currently different.
- Language: F95
- Needs: no libraries
- Host: AWI

6.8 compute_stats

- Description of functionality/purpose : Compute higher order ensemble statistics (skewness, kurtosis) for a single grid point.
- Inputs: ensemble values for a single grid point, ensemble mean value at the grid point. Size of ensemble.
- Outputs: Values of skewness and kurtosis
- Required work: The module needs to be adapted, because the implementation is not yet generic.
- Language: F95
- Needs: no libraries
- Host: AWI

6.9 eofcovar

- Description of functionality/purpose : Compute EOF decomposition (SVD) of perturbation matrix of some state trajectory. Used to prepare ensemble generation based on singular vectors/values of state trajectory. It also includes multivariate normalization. The current implementation also includes the reading of the model fields from model-specific files. Also, it writes model-specific outputs. (With PDAF, we always use the same output format for all models. However, the included fields are model-specific.)

- Inputs: Currently, it reads a state trajectory from a Netcdf file.
- Outputs: Currently, it writes the singular vectors, singular values, ensemble mean state as state vectors into a Netcdf file
- Required work: The code can be generalized. SVD computation part can be put into a separate subroutine. Also the file reading and writing can be separated.
- Language: F95
- Needs: NetCDF, LAPACK (DGESVD)
- Host: AWI

6.10 mvnormalize

- Description of functionality/purpose : Normalize an array for unit standard deviation. This is used to perform multivariate normalization in the module 'eofcovar'.
- Inputs: state perturbation array; size of array, offset of a field in the array; size of field
- Outputs: normalized field, value of standard deviation of field before normalization
- Required work: The module is currently part of eofcovar. It could be separated.
- Language: F95
- Needs: no libraries
- Host: AWI

6.11 adaptive_forget

- Description of functionality/purpose : Compute forgetting factor adaptively according to statistical consistency (variance of innovation = 1/forget variance ensemble + variance observations)
- Inputs: ensemble array, ensemble mean, observation vector, size of ensemble array; size of observation vector; ensemble size; default value of forgetting factor
- Outputs: computed forgetting factor
- Required work: The routine is experimental and a core routine of PDAF. It could be generalized for use outside of PDAF.
- Language: F95
- Needs: no libraries
- Comments: The routine is MPI-parallel for domain decomposition. There are two versions for global and local filters.
- Host: AWI

6.12 geof

- Description of functionality/purpose: Compute global EOF decomposition.
- Inputs: Directory with a set of control vectors.
- Outputs: Directory with the EOFs.
- Language: F95
- Needed: in WP4 for CNRS-LEGI.
- Remarks: The algorithm does not require to load the whole ensemble in memory(which is impossible for large size application).
- Host: CNRS-LEGI.

6.13 lroa

- Description of functionality/purpose: Perform square-root observational update, with localization algorithm (as described in Brankart et al., 2011).
- Inputs: Directory with the square root of the background error covariance matrix (as a set of control vectors), the background control vector, the observation vector (following SESAM NetCDF convention), the observation error covariance matrix (assumed diagonal).
- Outputs: Directory with the square root of the updated error covariance matrix, the updated control vector.
- Language: F95
- Needs: NetCDF
- Remarks: The algorithm does not require to load the whole ensemble in memory (which is impossible for large size application), and can be used to update an ensemble forecast.
- Host: CNRS-LEGI.

6.14 adap

- Description of functionality/purpose: Compute optimal inflation factors for the forecast error covariance matrix, and for the observation error covariance matrix (as described in Brankart et al., 2010).
- Inputs: A list of files with statistics describing the previous observational updates (as optionally output by the <lroa> module).
- Outputs: Optimized inflation factors (as optionally input by the <lroa> module).
- Language: F95
- Needs: NetCDF
- Host: CNRS-LEGI.

6.15tgop

- Description of functionality/purpose: Sample truncated Gaussian probability distribution.
- Inputs: Directory with the square root of the (reduced rank) covariance matrix of the reference Gaussian distribution (as a set of control vectors), the mean of the reference Gaussian distribution, directory with the set of linear inequality constraints (as a set of control vectors).
- Outputs: Directory with the required sample (as a set of control vectors).
- Language: F95
- Needs: NetCDF.
- Host: CNRS-LEGI.

6.16Anam

- Description of functionality/purpose: Estimate and apply anamorphosis transformation (as described in Béal et al., 2011).
- Input : Directory with the input ensemble (as a set of control or observation vectors).
- Outputs: (estimation of the transformation): Directory with a set of quantiles of the input ensemble (as a set of control or observation vectors).
- Outputs: (application of the transformation): Directory with the transformed ensemble (as a set of control vectors).
- Language: F95
- Needs: NetCDF.
- Remarks: The algorithm does not require to load the whole ensemble in memory (which is impossible for large size application).
- Host: CNRS-LEGI.

7 List of components from the proposal

The SANGOMA proposal document states a long list of components that might be shared. These modules are not mentioned by the partners in the initial inquiry. For completeness and future reference we give an overview of these modules in this section because it is likely that a number of these components will be realized in the future.

We have categorized the software components in five categories.

- Diagnostic tools,
- Perturbation and stochastic modelling tools,
- Transformation tools,
- Analysis step; These components change/improve the model predictions by use of the observed values.
- Utilities; These components are of general nature and possibly used to partially implement modules from the other categories.

7.1 Diagnostic tools

- Consitency checks on
  - the innovation d=y-h(xb)
  - residual after analysis r=y-h(xa)
  - model observations b=h(xa)-h(xb) = d-r
  - compare computed covariances with prescribed covariance matrices e.g. E[d dT]=HBHT+R, E[r dT]=R, E[b dT]=HBHT, E[r rT]=R(HBHT+R)-1R
- Tools to validate analysis error covariances (for qualification of new products) *(not currently available)*
- Tools to re-adapt signal/noise ratio based on those statistics.
- Tools for checking null hypothesis of diagonal **R** or adapting it.
- If **R** is not diagonal, tools to estimate the correlation length of the observation errors *(not currently available)*
- A posteriori data outlier detection based on diagonal part of E[r rT] *(not currently available)*
- Tests for checking null-hypothesis of unbiased innovations/increments *(not currently available)*
- Bias detection (via E[d], E[r] and E[b] *(This is partially already implemented but further development is needed)*
- Representers
  - Fitting of distributions *(not currently available)*

    - Rank histograms and probabilistic scores to check consistency of the pdf described by the ensemble with the observations (Talagrand et al, 1997; Hamill, 2001)
    - Taylor diagrams
    - Measures of observation impact such as sensitivity matrix, degrees of freedom, and, for linear systems, relative entropy and mutual information.
    - Checking of null hypothesis of a Gaussian distribution (e.g. Anderson–Darling test,...) *(not currently available)* Generalization of statistics for non-linear **h** (read HBHT as mean of (h(x+x')-h(x))(h(x+x')-h(x))T)
    - Increment statistics. They have the advantage of repeated realizations. This is especially interesting for local bias analysis and systematic modelling errors. *(not currently available)*
    - Analysis step diagnostics in terms of the performance of the observational array at detecting prior errors (through singular value analysis of Sakov's "**S**" matrix , Sakov et al., 2010) *(not currently available)*
    - Analysis step diagnostics of the performance of the observational array with the ensemble observation sensitivity method of Liu and Kalnay (2008)**.**
    - For all tests, including versions dealing with covariance matrix analyses, simple tests on trace values can be provided when repeated realizations do not allow a correct estimation of statistical averages. In these the trace evaluation uses spatial averaging, which is used as a proxy of the mathematical expectation (E[.]) via the ergodic theorem and can lead to very efficient tests.

7.2 Perturbation and stochastic modelling tools

In stochastic data assimilation schemes, uncertain aspects of the model need to be perturbed within the range of uncertainty. This can be done either by perturbing directly the uncertain forcing fields and parameters or by adding a perturbation to the model state, which should
represent the accumulated effect of the model uncertainties. By choosing the method to generate the perturbations, one defines implicitly their pdf. It is generally easier to create a perturbation scheme that produces realistic perturbations (for example smooth perturbations), than to define first the pdf of the perturbations and sample from this pdf.

- Spectral methods (Evensen, 2003) *(tools for 1D and 2D are already available but should be adapted. A version for higher dimensions needs to be developed)*
- Dynamically balanced and boundary-aware perturbations *(some tools exist already for multivariate perturbations consistent with the harmonic shallow water equations and tracer perturbations constrained by the advection-diffusion equations (Barth, 2009). Other types of dynamical balances need to be implemented.)*
- Perturbations based on multivariate EOFs *(such tools are already available by most partners, but must be adapted. Since using a relatively small number of EOFs introduces spurious long-range correlations, a localized perturbation scheme based on EOFs might be needed.)*

- Perturbations with prescribed spectrum: such techniques are used in stochastic modelling of the turbulence (e.g. Delsole, 2004) *(their use is currently not explored in the context of data assimilation.)*
- Observation perturbations. Some assimilation schemes require also perturbing the observations (Burgers et al. 1998). *(Such tools already exist and will be adapted and shared among partners.)*
- Easily configurable pdfs for uncertainty of simple parameters (even by GUI)
- Modular framework for extending a model with stochastic sources
- Wavelet-based stochastic fields, e.g. for uncertainty in wind forcing with good parallel performance.
- Coupling to existing uncertainty modelling tools like DUE and UncertML

7.3 Transformation tools

- The analysed model state is sometimes modified a posteriori in order to ensure that the model state is balanced and satisfies some constraints (which often cannot be expressed by covariances). In particular, methods to adjust the model state into statically stable situations are often used. These tools will be generalized and shared.
- EOF calculations (possibly multivariate and multigrid) need appropriate normalization depending on variables and grid size.
- Gaussian Anamorphosis (Wackernagel, 2003; Bertino et al., 2003; Simon and Bertino, 2009; Béal et al., 2010). The analysis is performed on transformed variables, which have a distribution closer to a Gaussian distribution. Tools to derive this transformation will be made available.

7.4 Analysis step

The analysis step is the central piece of all DA models and therefore available for a large number of DA schemes not repeated here. Here, the ambitions for collaborative developments are more difficult since these tools are sometimes more strongly coupled to models. However, knowledge and experience with different methods and algorithms from partners will be shared among the partners. WP4 deals specifically with the innovative analysis steps.

7.5 Utilities

- Some observation types require sophisticated observation operators and need a “model” describing the platform (e.g. virtual glider in a model, virtual Argos in a model instead of pseudo profiles). Those observation operators should be shared to avoid the duplication of efforts.
- General purpose data manipulation tools not found in software libraries will also be shared.

8 Summary and recommendations

Software components will be developed within the SANGOMA project that can be used by all partners. The necessary information was gathered from the partners using a shared on-line document.

Before these modules can be developed we need to have insights in the data assimilation frameworks of the various partners and an initial idea of modules we need. Therefore we have to look at the various aspects of the data assimilation systems. Important aspects we need to consider before we can start developing any common tool are the licence of the software, forms of parallelization, coupling to the model and programming language.

We have investigated how various languages can be combined and what is difficult and what not. With regard to interoperability, C would be the best choice to program common tools. However, since Fortran and Matlab/Octave is extensively used by the partners we advice to use Fortran for the computational shared routines and Matlab/Octave for prototyping purposes.

Software components can be shared if we define interfaces (object oriented) or if we define a data model. We have looked at the setup of the components in the various data assimilation frameworks of the partners and a common data model seems to fit best. The common data model, that needs to be developed, will have both an in memory as a NetCDF specification. This approach allows components to be reused in memory or as stand alone program using NetCDF files.

Software licensing is a very important issue. When software components are shared we are confronted with the incompatibility of various licenses. From investigating the available licenses of the data assimilation frameworks of the partners we advice to use the LGPL license for all shared components in SANGOMA. Based of the input from the partners we have specified an initial list of modules we will develop in SANGOMA. This list will grow in time.

Based on the analysis presented in this report, we recommend:

1. To use Fortran for the main computational routines in SANGOMA.
2. To use Matlab/Octave for prototyping within the SANGOMA project and write this code such that it runs under Matlab and Octave without modification.
3. A dual approach where modules are made available both as subroutines that share a common data-model as well as stand-alone programs that use NetCDF files (with CF standard meta-data) for input and output.
4. The use of the popular LGPL (http://www.gnu.org/licenses/lgpl.html). Participants of SANGOMA that publish there code as GPL and own the copyright, can contribute this code under LGPL to SANGOMA. GPL code that is not owned by a SANGOMA partner cannot be used.