SANGOMA

WP2: Sharing and Collaborative Development

Lars Nerger

Alfred Wegener Institute for Polar and Marine Research
Bremerhaven, Germany

and
Bremen Supercomputing Competence Center BremHLR
Bremen, Germany

lars.nerger@awi.de

BremHLR
Kompetenzzentrum für Höchstleistungsrechnen Bremen

Synopsis

Provide tools of interest to the Data Assimilation community
and avoid redundant developments

by

adapting existing and developing new tools according to data model and interface standard from WP1.

Possible Tools

- Identification by WP1
- Preliminary inventory of existing tools generated when proposal was formulated
- New tools identified by WP3 and WP4 during project

4+1 categories

- Diagnostic tools
- Perturbation tools
- Transformation tools
- Utilities
- [Analysis steps ➜ WP 3]

Existing tool boxes

Existing tools spread over range of tool boxes:

- Beluga/Sequoia (Toulouse)
- OpenDA (Delft)
- PDAF (AWI Bremerhaven)
- SESAM (Grenoble)
- NERSC EnKF repository (Bergen)
- OAK (Liege)
- [DART (NCAR, Boulder, CO, USA)]

- Tool boxes developed for their particular requirements

➜ Keep the tool boxes, but harmonize tools in them

Examples

- Diagnostic tools
  - statistical consistency checks (Histograms, Brier, CRPS)
  - relative entropy, mutual information (for PFs)
- Perturbation and stochastic modeling tools
  - generate perturbations for initial ensembles
  - stochastic perturbations during ensemble forecasts
- Transformation tools
  - Gaussian Anamorphosis
- Utilities
  - sophisticated observation operators
  - cost function and it’s gradient from PODs

Adapting and Developing DA tools

- WP1 identifies existing and required new tools
- WP2
  - adapts existing tools
  - develops new tools
- Follow data model and interface specified in WP1

SANGOMA tools

- Provide tools together with
  - documentation
  - simple application examples (test routines)
  - use ‘make’ to build tool library
  - use ‘make’ for application examples

Programming Languages

Matlab/Octave .m

- reduced development time
- if CPU performance is not essential
- Matlab or Octave frequently used for
  - testing
  - data manipulation
  - post-processing

Fortran

- for tools tightly coupled to numerical models
- if CPU performance is essential
- Fortran frequently used for large-scale numerical models (NEMO, TOPAZ, HYCOM, etc.)

Adaptation of existing tools

- Various tools already exist in DA software of consortium partners
- Implementations vary
  - limited re-use
  - harmonization required

- Adapt tools to the specifications of WP1
- Ideally performed by originator of tools (spread relatively uniform)

Development and implementation of new tools

- WP1 identifies necessary additional tools
  - required by WP4
- Discuss new tools in developer’s forum to meet requirements
- Implement new tools according to standards from WP1
- Dispatch work between all partners
  (WP leader in charge of balanced workload)

Work distribution

- Main contributors: AWI and TU Delft (both also strongly involved in WP1)
- Collection of tools from all partners
- All partners involved in adaption and development

Partner	Ulg	UREAD	AWI	TUD	CNRS	NERSC
man-months	2	2	6	4	4	2

Timing of Tasks

- Use SVN repository created at M! of SANGOMA
- Adaption and development of tools (M7 to M48)
- Codes in SVN repository updated continuously

Milestones & Deliverables:

- Milstones:
  - Public bundled versions
  - V0 (Month 12), V1 (M30), V2 (M48)
- Deliverables:
  - Software reports for V0, V1, V2

Green: completed; blue: due 30/4/2014

SVN Server (Task 2.1)

- SVN: version control system
- Central server for shared development
- Used internally
- Storage for
  - Documents (www, templates, reports, etc.)
  - Software codes
- Description for SVN server and structure (D2.1)
  - Standard organization for code (trunk/, tags/, branches/)
  - Directories for documents, templates, etc.

Software release V0 – included tools

Diagnostic tools

sangoma_ComputeHistogram

sangoma_ComputeEnsStats

mutual_information

relative_entropy

sensitivity

Perturbation tools

sangoma_MVNormalize

sangoma_EOFCovar

weakly constrained ensemble perturbations

Transformation tools

anam_setup

anam_transform

anam_invtransform

Utilities

hfradar_extractf

PodCalibrate

EnKF

Language: Fortran, Matlab/Octave, Java

Software release V0

- Intended to test the collaborative development
- Codes not yet adapted to data model

- Required work:
  - Adaptation to data model
  - Ensure independence from assimilation system
  - Uniform naming scheme
  - Implementation of application examples

Steps toward release V1 (month 30)

- Adaption and addition of tools
- How to extend the selection of tools?
  - Discussion:
    - Which tools are important?
    - If it exists, which partner can provide/adapt it?
    - If new, who can implement it?
    - Fortran or Matlab? Java?

Discussions after last project meeting

Collected “wish lists” from project partners by email

- Most interest in diagnostic tools, e.g.
  - whiteness of innovations/residuals
  - degree of nonlinearity of a model relative to a perturbation
  - measure of influence of SVN truncation on covariances
- Transformations
  - anamorphosis (Fortran)
- Ensemble generation (perturbations)

Software release V1 – included tools

Diagnostic tools

sangoma_computeBRIER
sangoma_computeCRPS
sangoma_ComputeEnsStats
sangoma_ComputeHistogram
sangoma_ComputeMutInf
sangoma_computeRCRV
sangoma_ComputeRE
sangoma_ComputeSensitivity
sangoma_ComputeSensitivity_op
sangoma_arm

mutual_information
relative_entropy
sensitivity

Perturbation tools

sangoma_MVNormalize
sangoma_EOFCovar
mod_sangoma_pseudornd

weakly constrained ensemble perturbations

Transformation tools

anam_setup
anam_transform
anam_invtransform

Utilities

sangoma_computepod
sangoma_costgrad

hfradar_extractf

Language: Fortran, Matlab/Octave

Software release V1

- Focus on tools to compute scores for Benchmarks

- All codes adapted to data model
- Example cases provided for all tools
  (not yet complete)
- Naming scheme: sangoma_*.F90

Example of tool interface (from Deliverable 2.4)

4.1.4 `sangoma_computeCRPS` — Compute the CRPS and its decomposition (Source File: sangoma_computeCRPS.F90)

INTERFACE:

```
SUBROUTINE sangoma_computecrps(ENS, ANA, MISSING, NCASES, NENS, &
   CRPS, RELI, RESOL, UNCERT, BB, AA, CB_SORT, CB_SORT2) &
   BIND(C, name="sangoma_computecrps_")
```

USES:

```
USE, INTRINSIC :: ISO_C_BINDING
USE sangoma_base, ONLY: REALPREC, INTPREC
IMPLICIT NONE
```

Specifications for C-binding

ARGUMENTS:

```
INTEGER(INTPREC), INTENT(in)   :: NCASES       ! Size of the verification set
INTEGER(INTPREC), INTENT(in)   :: NENS         ! Ensemble size
REAL(REALPREC), INTENT(in)     :: ENS(NCASES,NENS) ! Ensemble
REAL(REALPREC), INTENT(in)     :: ANA(NCASES) ! Verification (analysis or observation)
INTEGER(INTPREC), INTENT(in)   :: missing(m)   ! 0 = obs is OK ; 1 = obs is not used
REAL(REALPREC), INTENT(inout) :: CRPS         ! CRPS (global score)
REAL(REALPREC), INTENT(inout) :: RELI         ! reliability part
REAL(REALPREC), INTENT(inout) :: RESOL        ! resolution part
REAL(REALPREC), INTENT(inout) :: UNCERT       ! uncertainty
REAL(REALPREC), INTENT(inout) :: BB(0:NENS)   ! decomposition coefficients of the CRPS
REAL(REALPREC), INTENT(inout) :: AA(0:NENS)   ! decomposition coefficients of the CRPS
```

Example of tool interface (Deliverable 2.4)

Interface specifications necessary for C-binding

CALL-BACK ROUTINES:

```
! Sorting routine for one vector
INTERFACE
   SUBROUTINE CB_SORT(NENS, V) BIND(C)
     USE sangoma_base, ONLY: REALPREC, INTPREC
     INTEGER(INTPREC),INTENT(in)  :: NENS    ! Size of vector
     REAL(REALPREC),INTENT(inout) :: V(NENS) ! Input/output vector
   END SUBROUTINE CB_SORT
END INTERFACE

! Sorting routine for two vectors
INTERFACE
   SUBROUTINE CB_SORT2(NENS, V1, V2) BIND(C)
     USE sangoma_base, ONLY: REALPREC, INTPREC
     INTEGER(INTPREC),INTENT(in)  :: NENS     ! Size of vector
     REAL(REALPREC),INTENT(inout) :: V1(NENS) ! Input/output vector
     REAL(REALPREC),INTENT(inout) :: V2(NENS) ! 2nd Input/output vector
   END SUBROUTINE CB_SORT2
END INTERFACE
```

Discussion: Some open questions

1. Treatment of single and double precision
2. For 2D arrays:
   Is the difference in Fortran and C storage a problem?
3. Do we need C-prototypes?
4. How to continue?
5. Other points?

1. Double and single precision

Status

- Tools internally use ‘REAL’ (no KIND)
- KIND is used in c-bind interface
- LAPACK/BLAS calls use double precision
- Module sangoma_base uses C_DOUBLE

Possibility

- No changes to source code when changing precision
- Specification at compilation
  - -fdefault-real8
  - preprocessor flags?

➢ How to treat LAPACK/BLAS calls?

➢ How to handle C_DOUBLE in sangoma_base?

Conclusion

- WP2 results in
  - Collection of harmonized existing DA tools
  - Addition of new tools with standard data model and interface
  - Publicly available bundle of “Sangoma-Tools”
- Expected achievements
  - Improved re-use of DA tools
  - larger selection of available tools
  - simplified use of tools
    (documentation, test cases)
- Current status
  - Adaption and extension of tools for next release
    (V1, due 30/4/2014)