Data model ideas for SANGOMA

Web meeting May 8, 2012

Martin Verlaan, Nils van Velzen & Umer Altaf

EOF example

Motivation:

- EOF or POD is mentioned in various forms as a tool to be shared in Sangoma.
- Basics are simple enough for a discussion
- Many complications in extensions

Core computations as Matlab code:

```
function Z=eof(X,p);
E=X'*X;
[U,D,V]=svd(E); %E=U*D*V'
T=V/sqrt(D);
Z=X*T(:,1:p);
```

Example of EOF module

Input: x_1,..,x_N
Output: z_1,..,z_M with M<=N

Pseudo code:

- E(i,j)=dot(x_i, x_j)
- Symmetric eigenvalue decomposition of E
- z_i= linear combination of x_1,...,x_N

- Computational core is too simple to share? Just 2x blas + one lapack routine.
  → Useful tools need to contain significant building blocks to be useful.

Extension 1: Data storage

Input and output for EOF can be available as:

- NetCDF files
  - CF needs multiple variables (salinity,temperature)
  - CF needs meta-data, eg. Grid, time, units,...
  - Possibly too large to fit in memory
- In memory (fortran-like)
  - Single array
  - Scattered over multiple arrays
- Different programming language
  - Matlab, Java, ...
- Distributed over processes (parallel computing)

NetCDF storage

Program eof_netcdf
  Read from netcdf
    Per variable access for CF → store in X
    Keep meta-data separately for writing of eof's
      :
    Call eof(X,Z) subroutine
      :
  Write to netcdf
    Write per variable ← extract from Z
    Add meta-data

Issues:

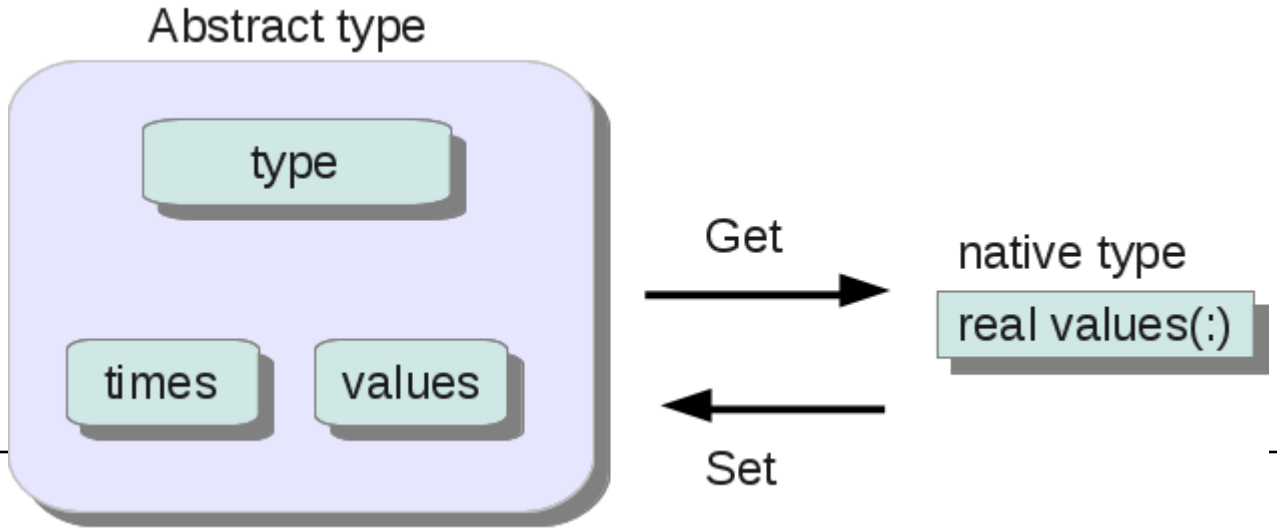
- Needs more work if it does not fit in memory
- Netcdf wrapper potentially more work than eof core

In memory storage

- Single array vs multiple blocks
  - No pointer arrays like float **X or array<float[]> in fortran → needs Type in fortran
  - real x_1(:),...x_n(:) to X(:,:) transform always requires a copy of the data
  - real X(:,:) to x_1(:),...x_n(:) transform can be handled with pointers or by copying
- Portability
  - Basic types like, real(:), map fine to c in fortran2003
  - Complex types and pointers are more difficult to map

Abstract data types

- No direct access to data → only through subroutines
  - get_values and set_values
- Does not require copying data if performance is important
- Allows different implementations with same interface; eg netcdf vs in-memory



Extension 2: transformations

- Equal summed weights per variable
- Energy norm:
  - U → 0.5*dx*dy*dz*rho*u^2
  - H → 0.5*dx*dy*rho*h^2
- Non-linear transforms
  - Log for positive variables
  - Anamorphosis

→ Requires meta-data

Extension 3: Parallel Computing

- Options:
  - Explicit MPI calls
  - Additional version without MPI or empty stub-routines
  - Hide MPI functionality from algorithm
- 2-directions: members vs. domains