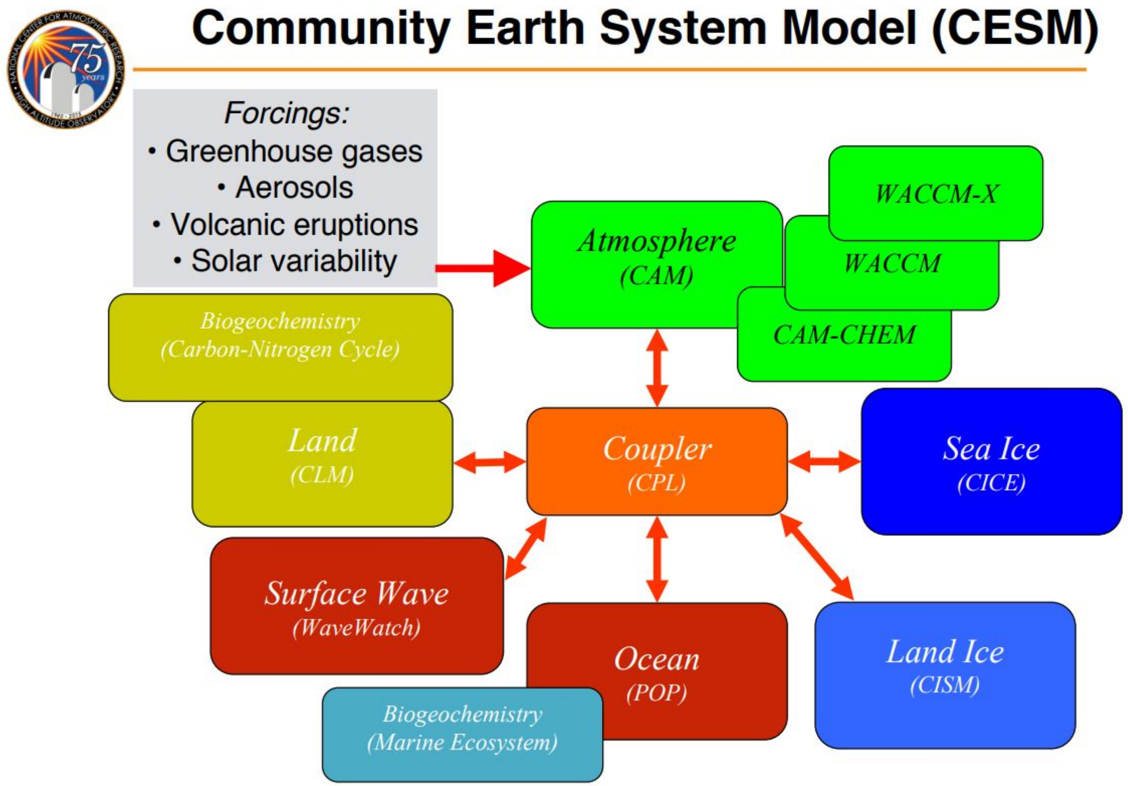
WACCM-X and JEDI Model Interface

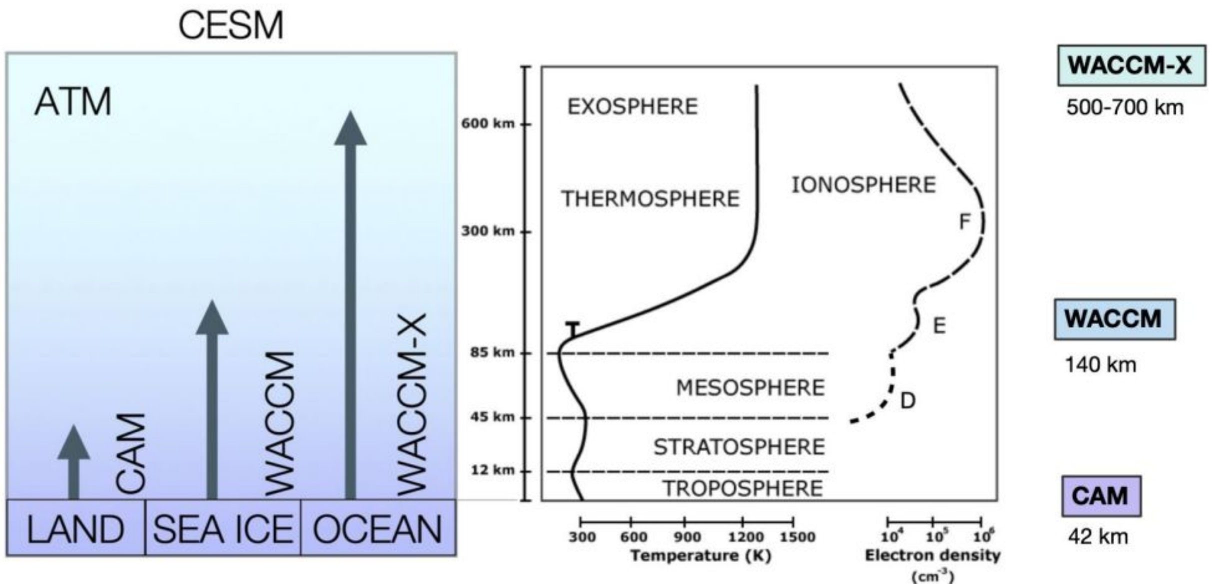
Simin Zhang (CU), Mark Miesch(CU/CIRES),
Nicholas Pedatella(NCAR/HAO), Hui Shao(JCSDA),
Thomas Berger (CU), Eric Sutton(CU)

2025 Community Space Weather Modeling and Data Assimilation Workshop
September 10-11, Boulder

Introduction of WACCMX and JEDI

The Whole Atmosphere Community Climate Model with thermosphere and ionosphere extension (WACCM-X)

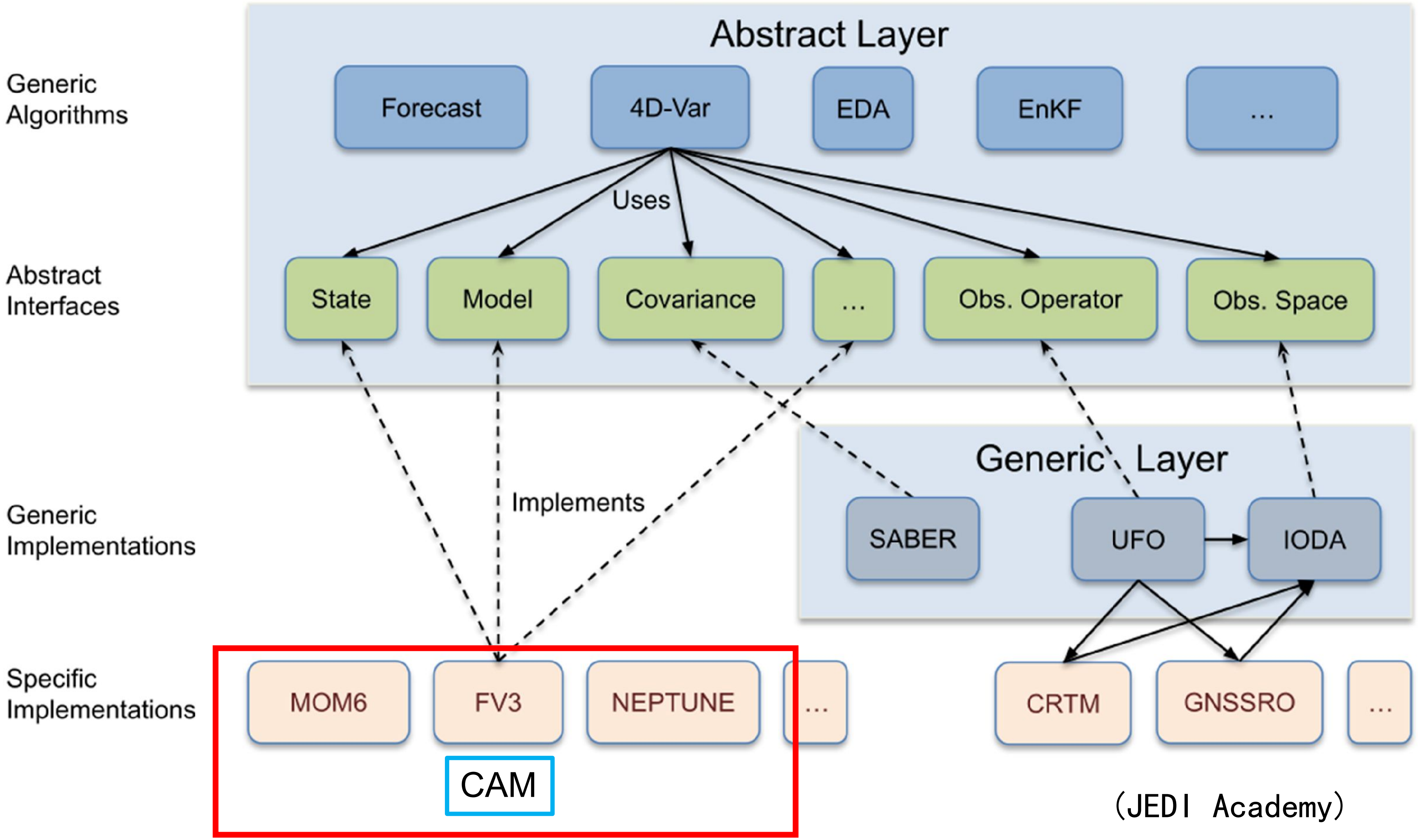




WACCM-X is built on WACCM
WACCM is built on CAM
CAM is the NCAR Community Atmosphere Model

WACCM-X is a model of the entire atmosphere that extends into the thermosphere and includes the ionosphere.

JEDI Generic Design



Generic, portable, model-agnostic DA system

Use **object-oriented** and **generic** programing

Each model implements pre-defined abstract interfaces

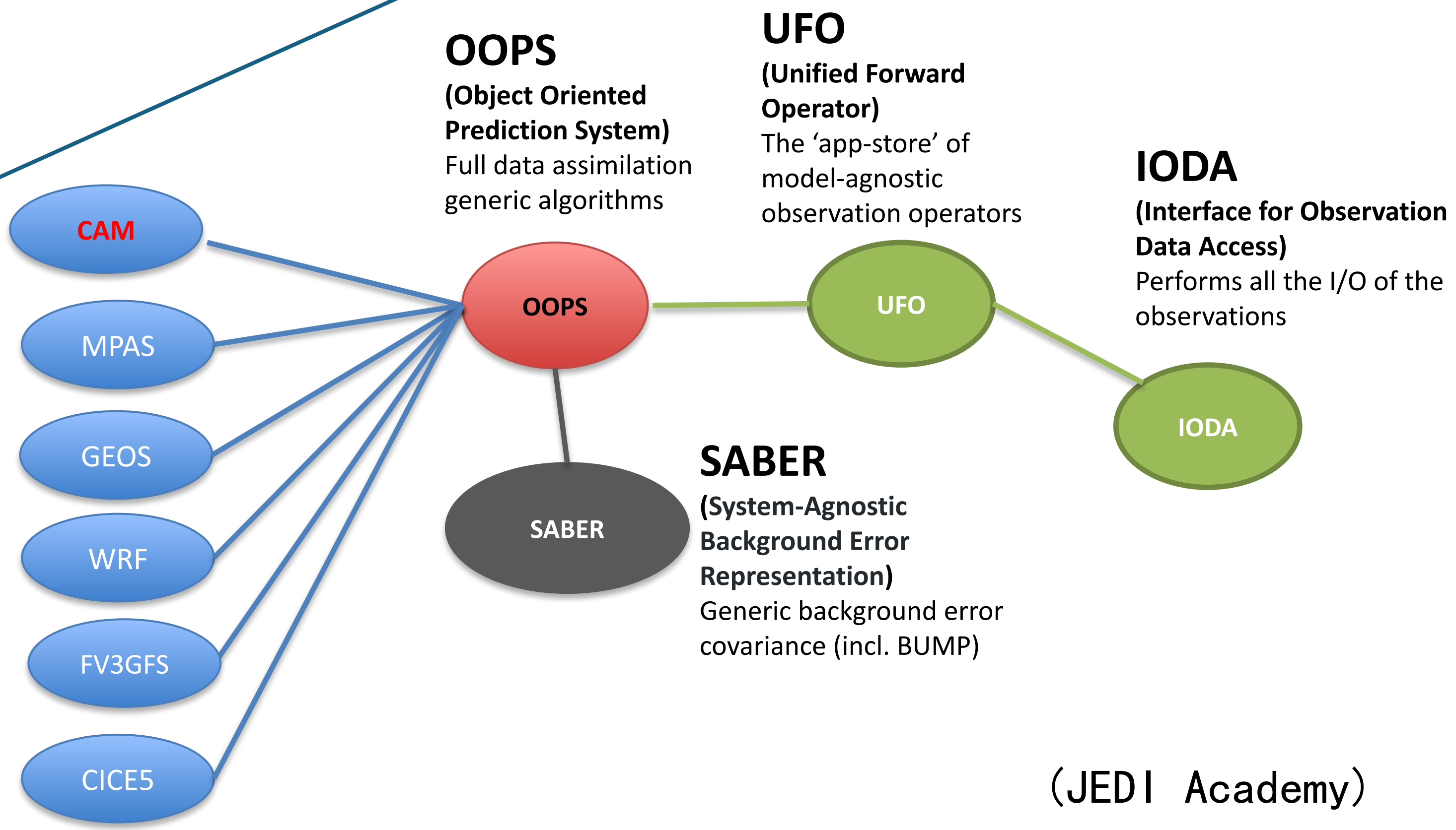
JEDI Components

OOPS SABER IODA

$$J(\mathbf{x}) = (\mathbf{x}-\mathbf{x}_b)^T \mathbf{B}^{-1} (\mathbf{x}-\mathbf{x}_b) + \sum_{i=0}^{n} (\mathbf{y}_i - \mathbf{H}_i[\mathbf{x}_i])^T \mathbf{R}^{-1} (\mathbf{y}_i - \mathbf{H}_i[\mathbf{x}_i])$$

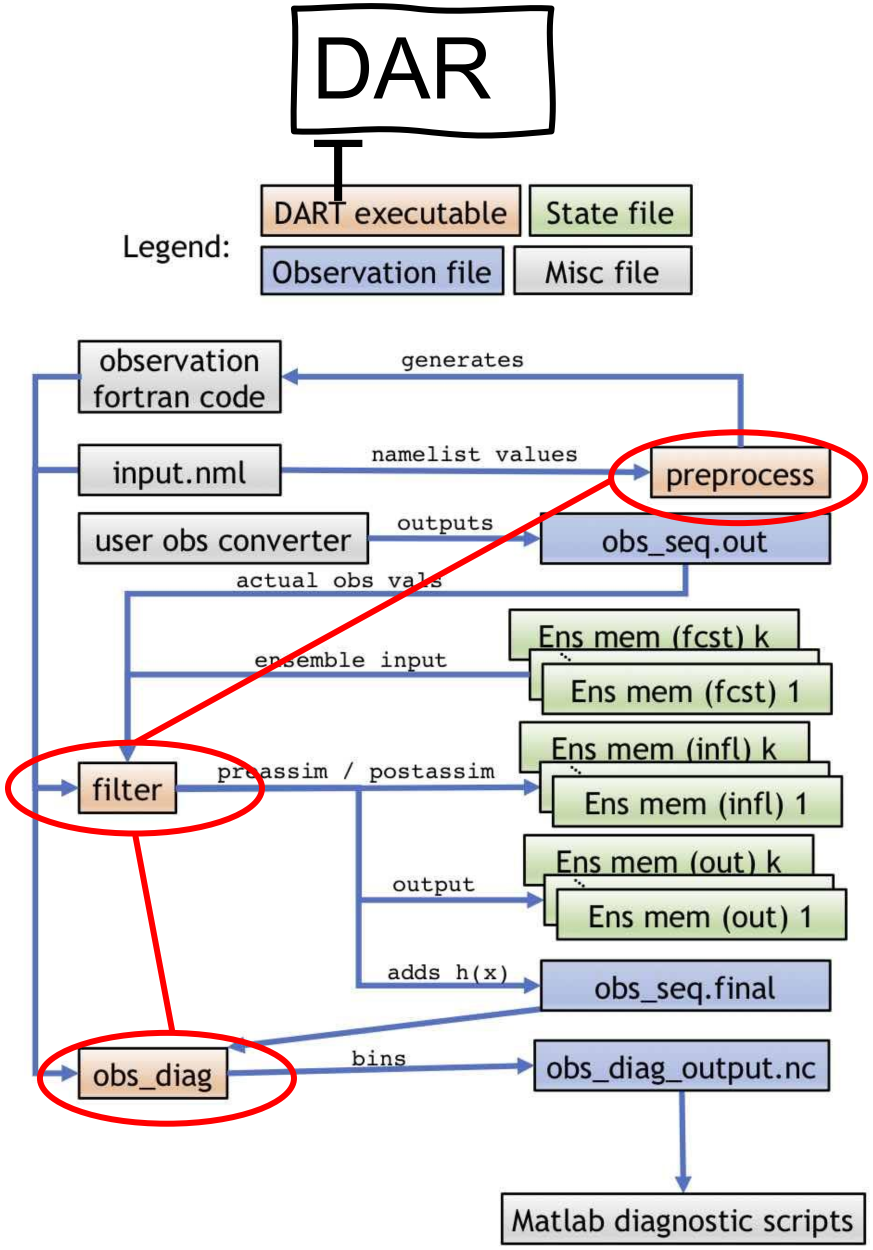
Model (e.g. CAM-JEDI) UFO

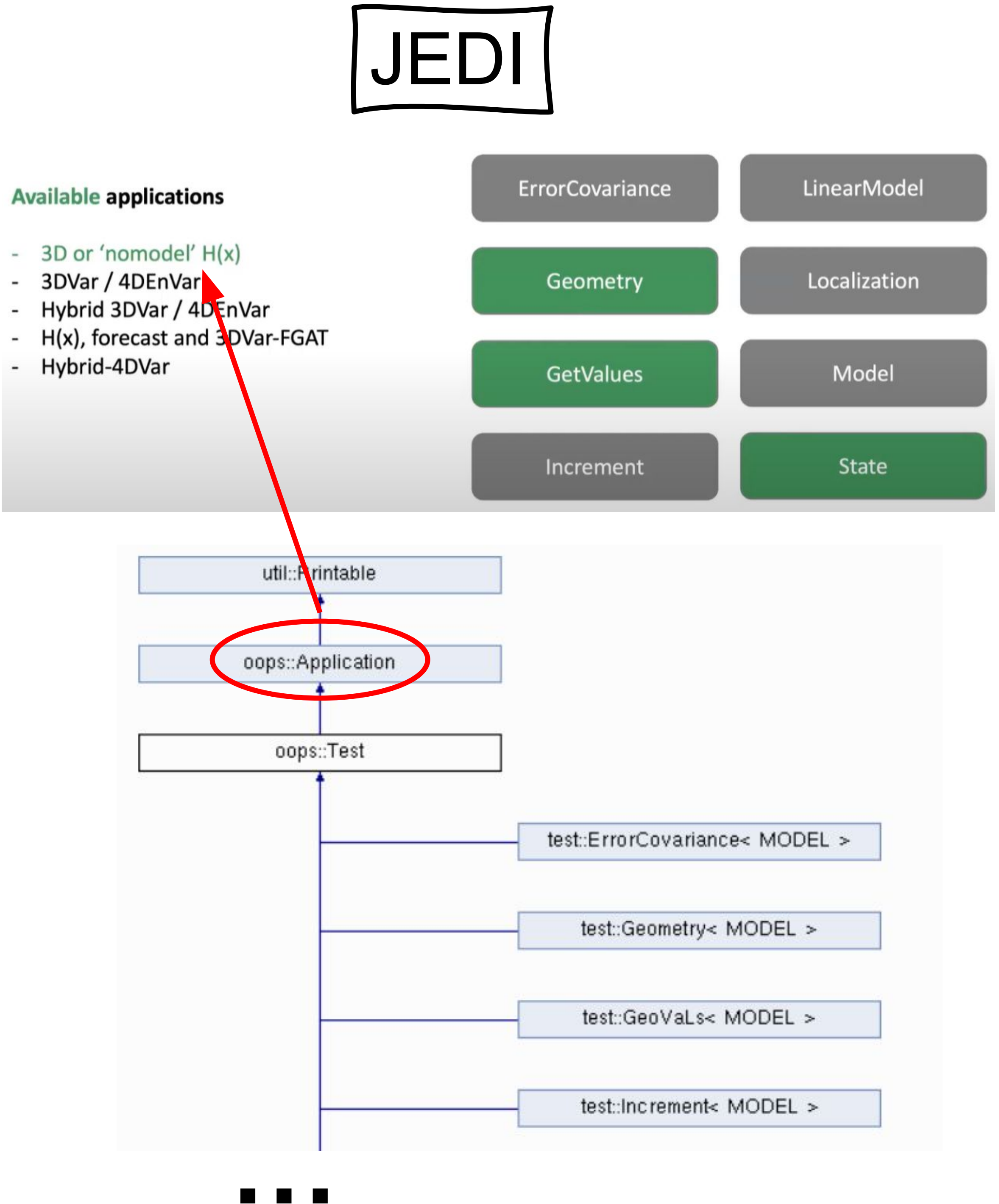
$$\mathbf{x}_i = \mathbf{x}(t_i) = \mathbf{M}_{0\rightarrow i}(\mathbf{x})$$



(JEDI Academy)

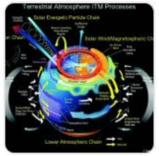
JEDI workflow





CAM-JEDI Progress

CAM repo in Github



SWORD Space Weather Center of Excellence

cam-bundle : run the JEDI

cam-bundle Public

JEDI bundle for NCAR's Community Atmosphere Model (CAM)

Shell ☆ 0 Apache-2.0 0 0 0 Updated on Oct 9, 2024

```
#ecbuild_bundle( PROJECT eckit    GIT "https://github.com/ecmwf/eckit.git" TAG 1.24.4 )
ecbuild_bundle( PROJECT fckit    GIT "https://github.com/ecmwf/fckit.git" TAG 0.11.0 )
ecbuild_bundle( PROJECT atlas    GIT "https://github.com/ecmwf/atlas.git" TAG 0.35.0 )

ecbuild_bundle( PROJECT oops     GIT "https://github.com/jcsda/oops.git"        BRANCH develop UPDATE )
ecbuild_bundle( PROJECT vader    GIT "https://github.com/jcsda/vader.git"       BRANCH develop UPDATE )
ecbuild_bundle( PROJECT saber    GIT "https://github.com/jcsda/saber.git"       BRANCH develop UPDATE )
```

```
ecbuild_bundle(PROJECT cam-jedi SOURCE "../../cam-jedi" UPDATE)
```

Bundle containing all the repositories needed to build the JEDI interface for NCAR's Community Atmosphere Model (CAM).

cam-jedi : the interface between WACCM-X and JEDI

cam-jedi Private

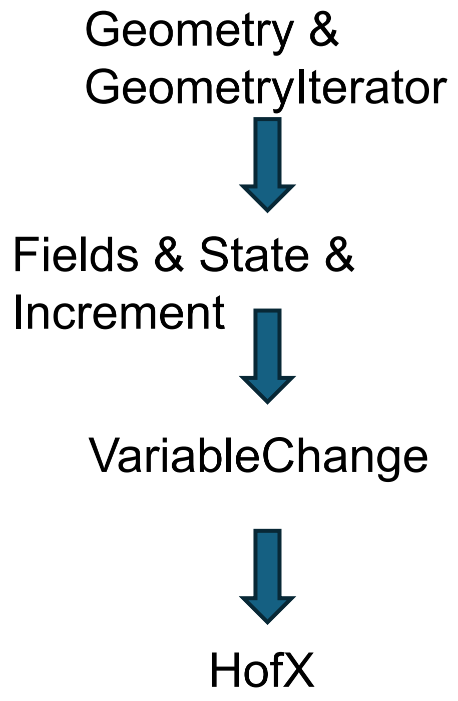
JEDI Model Interface for NCAR's Community Atmosphere Model (CAM)

Fortran ☆ 0 Apache-2.0 0 0 1 Updated 3 days ago

Instead of linking WACCM-X with JEDI directly (as with fv3, MPAS), we will run them separately, communicating through files like in DART

The model interface classes

Class	Description
ErrorCovariance	Background error covariance model (also implemented in SABER).
Geometry	The geometry of the forecast model/background grid.
GeometryIterator	Iterator over the grid points, needed only for LETKF applications.
GetValues	Interpolation from the model space state to observation locations.
Increment	Everything associated with an increment with model variables on the model grid.
LinearModel	The tangent linear and adjoint version of the forecast model.
LinearVariableChange	Transform between an increment with one set of fields and another.
Localization	Model ensemble localization (also implemented in SABER).
Model	The actual forecast model.
ModelAuxControl	
ModelAuxCovariance	Classes for dealing with model error.
ModelAuxIncrement	
State	Everything associated with the model state.
VariableChange	Transform between a state with one set of fields and another with different fields.



Geometry and GeometryIterator class

Geometry type

```
type :: camjedi_geom
   integer :: nlat, nlon, nlev,nilev, ngrid
   real :: P0              ! reference presure ;units "Pa"
   real (kind=kind_real),dimension(:),allocatable :: lat,lon
   real (kind=kind_real),dimension(:),allocatable :: lev       ! hybrid level at midpoints (10
nate
                                                            ! a:hyam b:hybm p0:P0 ps:PS
   real (kind=kind_real),dimension(:),allocatable :: hyam,hybm  ! hybrid A & B coefficient at l
   real (kind=kind_real),dimension(:),allocatable :: ilev       ! hybrid level at interfaces (1
nate
                                                            ! a:hyai b:hybi p0:P0 ps:PS
   real (kind=kind_real),dimension(:),allocatable :: hyai,hybi  ! hybrid A & B coefficient at l
   real (kind=kind_real),dimension(:),allocatable :: gw         ! latitude weights

   real (kind=kind_real),dimension(:,:),allocatable :: latgrid,longrid

   type(fckit_mpi_comm) :: f_comm
   type(atlas_functionspace) :: afunctionspace

   integer :: lon_size, lat_size, nodes, cells, lon_min_in, lon_max_in, lat_min_in, lat_max_in
   real (kind=kind_real) :: lon_min, lon_max, lat_min, lat_max
end type camjedi_geom
```

```
std::stringstream gridname;
gridname << "L" << nlon << "x" <<nlat;
atlas::RegularGrid grid(gridname.str());
atlas::MeshGenerator generator("structured");
atlas::Mesh mesh = generator.generate(grid);
```

Atlas: a ECMWF library for parallel data-structures supporting unstructured grids and function spaces

geometry.yaml: read variables from nc file

```
geometry:
  nml_file: "./modelinput/input_geo.nml"
  cam_file: "./testdata/CASE230503.cam.h0.2000-02.nc"
```

input_geo.nml: to generate the atlas grid

```
&geometry
   nlat = 96
   nlon = 144
   nlev = 126
   nilev = 127
/
```

Fields-State-Increment class

field and fields type: used in State and Increment class

```
!Field type (individual field)
type :: camjedi_field
 character(len=256) :: variable                         ! Field  name
 character(len=256) :: long_name                         ! Field long name
 character(len=256) :: units                            ! Field units
 character(len=256) :: cell_method                             ! usually time:mean
 character(len=256) :: horizontal_stagger_location      ! Stagger location in horizontal
 real(kind=kind_real), allocatable :: array(:,:,:)
 type(fckit_mpi_comm) :: comm                          ! Communicator
endtype camjedi_field
```

```
!Fields
type :: camjedi_fields
  type(camjedi_geom),pointer :: geom                 !< Geometry
!  type(oops_variables) :: vars                    !< List of variables
  integer :: nf                                   ! number of variables in subFields
  character, allocatable :: fldnames(:)                ! variable identifiers
  type(camjedi_field), allocatable :: fields(:)      ! array of variables
  type(datetime) :: time

  contains

     procedure :: axpy          => axpy_
     procedure :: dot_prod      => dot_prod_
     procedure :: gpnorm        => gpnorm_
     procedure :: rms           => rms_
     procedure :: self_add      => self_add_
     procedure :: self_schur    => self_schur_
     procedure :: self_mult     => self_mult_
     procedure :: self_sub      => self_sub_
     procedure :: zeros         => zeros_
     procedure :: ones          => ones_

     procedure :: copy          => camjedi_fields_copy
     procedure :: create        => camjedi_fields_create
     procedure :: delete        => camjedi_fields_delete
     procedure :: read_file     => fields_read
     procedure :: write_file    => fields_write
     procedure :: serial_size   => serial_size
     procedure :: serialize     => fields_serialize
     procedure :: deserialize   => fields_deserialize
     procedure :: to_fieldset
     procedure :: from_fieldset
     procedure :: check
     procedure :: check_resolution
     procedure :: sizes         => sizes_
end type camjedi_fields
```

Derived

Derived

Model Space: State

$$J(\Delta x) = \frac{1}{2}\Delta x^T \mathbf{B}^{-1}\Delta x + \frac{1}{2}\left(y_o - H(x_b + \Delta x)\right)^T \mathbf{R}^{-1}\left(y_o - H(x_b + \Delta x)\right)$$

$$\Delta x_a = \mathbf{B}\mathbf{H}^{\mathbf{T}}\left(\mathbf{H}\mathbf{B}\mathbf{H}^{\mathbf{T}} + \mathbf{R}\right)^{-1}\left(y_o - H(x_b)\right)$$

```
!> Fortran derived type to hold CAMJEDI state
type, extends(camjedi_fields) :: camjedi_state
contains
  procedure, public :: add_increment
end type camjedi_state
```

Model Space: Increment

$$J(\Delta x) = \frac{1}{2}\Delta x^T \mathbf{B}^{-1}\Delta x + \frac{1}{2}\left(y_o - H(x_b) - \mathbf{H}\Delta x\right)^T \mathbf{R}^{-1}\left(y_o - H(x_b) - \mathbf{H}\Delta x\right)$$

$$\Delta x_a = \mathbf{B}\mathbf{H}^{\mathbf{T}}\left(\mathbf{H}\mathbf{B}\mathbf{H}^{\mathbf{T}} + \mathbf{R}\right)^{-1}\left(y_o - H(x_b)\right)$$

```
type, extends(camjedi_fields) :: camjedi_increment
contains
  procedure, public :: diff_incr
  procedure, public :: dirac
  procedure, public :: random
  procedure, public :: getpoint
  procedure, public :: setpoint

end type camjedi_increment
```

Fields-State-Increment class

Fields

Inheritance

Inheritance

State

Increment

type camjedi_field							
variable	character(len=256)						
long_name	character(len=256)						
units	character(len=256)						
cell_method	character(len=256)	usually time:mean					
horizontal_stagger_location	character(len=256)	Stagger location in horizontal					
array(: , : , :)	real(kind=kind_real), allocatable						
comm	type(fckit_mpi_comm)						
type camjedi_fields			type, extends(camjedi_fields) :: camjedi_state		type, extends(camjedi_fields) :: camjedi_increment		
geom	type(camjedi_geom),pointer		procedure, public :: add_increment		procedure, public :: diff_states		
nf	integer	Number of variables			procedure, public :: dirac		
fldnames(:)	character(len=256), allocatable	Variable identifiers=field::variable			procedure, public :: random		
fields(:)	type(camjedi_field), allocatable				procedure, public :: getpoint		
time	type(datetime)				procedure, public :: setpoint		
procedure:			camjedi_state_interface_mod.F90	State.h/cc	camjedi_increment_interface_mod.F90	Increment.h/cc	
axpy	self,zz,rhs = self+zz*rhs	axpy_	camjedi_state_axpy_f90	accumul(const real_type & zz, const State & xx)	camjedi_increment_axpy_inc/state_f90	axpy(const real_type & zz, const Increment/State & dx, const bool check)///accumul(const real_type & zz	
dot_prod	fld1,fld2,zprod = sum(fld1*fld2)	dot_prod_	camjedi_state_dot_prod_f90		camjedi_increment_dot_prod_f90	dot_product_with(const Increment & other)	
gpnorm	fld,vpresent,vmin,vmax,vrms	gpnorm_	camjedi_state_gpnorm_f90	print(std::ostream & os)	camjedi_increment_gpnorm_f90	print(std::ostream & os)	
					camjedi_increment_random_f90	random()	
rms	fld,prms	rms_	camjedi_state_rms_f90	norm()	camjedi_increment_rms_f90	norm()	
self_add	self,rhs =self + rhs	self_add_	camjedi_state_self_add_f90		camjedi_increment_self_add_f90	operator+=(const Increment & dx)	
self_schur	self,rhs =self * rhs	self_schur_	camjedi_state_self_schur_f90		camjedi_increment_self_schur_f90	schur_product_with(const Increment & dx)	
self_mult	self,zz = zz*self	self_mult_	camjedi_state_self_mul_f90		camjedi_increment_self_mul_f90	operator*=(const real_type & zz)	
self_sub	self,rhs =self - rhs	self_sub_	camjedi_state_self_sub_f90		camjedi_increment_self_sub_f90	operator-=(const Increment & dx)	
zeros	self	zeros_	camjedi_state_zero_f90	zero()	camjedi_increment_zero_f90	zero()	
ones	self	ones_	camjedi_state_ones_f90		camjedi_increment_ones_f90	ones()	
copy	self,other	camjedi_fields_copy	camjedi_state_copy_f90	operator=(const State & rhs)	camjedi_increment_copy_f90	operator=(const Increment & rhs)	
create	self,geom, vars	camjedi_fields_create	camjedi_state_create_f90	State	camjedi_increment_create_f90	Increment	
delete	self	camjedi_fields_delete	camjedi_state_delete_f90	~State	camjedi_increment_delete_f90	~Increment	
read_file	self,f_conf,vdate	fields_read	camjedi_state_read_file_f90	read(const eckit::Configuration & config)	camjedi_increment_read_file_f90	read(const eckit::Configuration & config)	
write_file	self,f_conf,vdate	fields_write	camjedi_state_write_file_f90	write(const eckit::Configuration & config)	camjedi_increment_write_file_f90	write(const eckit::Configuration & config)	
serial_size	self,vsize	serial_size	camjedi_state_serial_size_f90	serialSize()	camjedi_increment_serial_size_f90		
serialize	self,vsize,vect_fld	fields_serialize	camjedi_state_serialize_f90	serialize(std::vector<real_type> & vect)	camjedi_increment_serialize_f90	serialSize()	
deserialize	self,vsize,vect_fld,index	fields_deserialize	camjedi_state_deserialize_f90	deserialize(const std::vector<real_type> & vect, size_t &	camjedi_increment_deserialize_f90	deserialize(const std::vector<real_type> & vect,size_t & index)	
to_fieldset	self,afieldset		camjedi_state_to_fieldset_f90	toFieldSet(atlas::FieldSet & fset)	camjedi_increment_to_fieldset_f90	toFieldSet(atlas::FieldSet & fset)	
from_fieldset	self,afieldset		camjedi_state_from_fieldset_f90	fromFieldSet(const atlas::FieldSet & fset)	camjedi_increment_from_fieldset_f90	fromFieldSet(const atlas::FieldSet & fset)	
check	self		camjedi_state_add_increment_f90	operator+=(const Increment & dx)			
check_resolution	fld1,fld2						
sizes	self,nlon,nlat,nlev	sizes_			camjedi_increment_print_f90		
					camjedi_increment_sizes_f90		
					camjedi_increment_getpoint_f90	getLocal(const GeometryIterator & iter)	getpoint
					camjedi_increment_setpoint_f90	setLocal(const oops::LocalIncrement & values, const GeometryIterator & iter)	setpoint
					camjedi_increment_random_f90	random()	random
					camjedi_increment_diff_incr_f90	diff(const State & x1, const State & x2)	diff
					camjedi_increment_dirac_f90	dirac(const eckit::Configuration & config)	dirac

VariableChange class

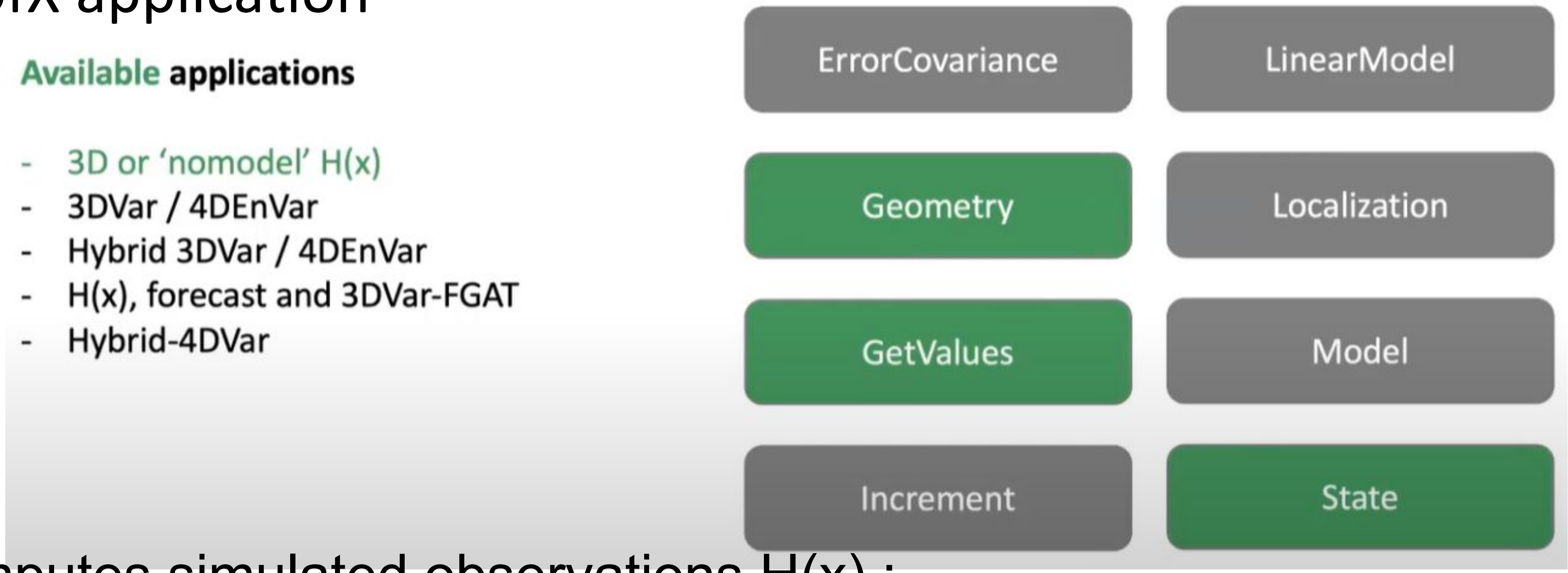
/cam-jedi/src/camjedi/VariableChange/

```
.
├── Base
│   ├── VariableChangeBase.cc
│   └── VariableChangeBase.h
├── Model2GeoVaLs
│   ├── camjedi_vc_model2geovals_interface_mod.F90
│   ├── camjedi_vc_model2geovals_mod.F90
│   ├── VarChaModel2GeoVaLs.cc
│   ├── VarChaModel2GeoVaLs.h
│   └── VarChaModel2GeoVaLs.interface.h
├── VariableChange.cc
└── VariableChange.h
```

```
input variables:
- air_temperature
- O
- O2
- air_pressure_at_pressure
- surface_geopotential
- specific_humidity
output variables:
- air_temperature
- O
- O2
- air_pressure_at_surface
- surface_geopotential
- specific_humidity
- geometric_height
- air_pressure
```

```
case ("air_temperature")
  have_T = .true.
  write(*,*) 'have_T=', have_T
  T = xg%fields(ivar)%array
  write(*,*) "case sum T:",sum(T)
case ("specific_humidity")
  have_Q = .true.
  write(*,*) 'have_Q=', have_Q
  Q = xg%fields(ivar)%array
   write(*,*) "case sum Q:",sum(Q)
case ("O")
  have_O = .true.
  write(*,*) 'have_O=', have_O
  O = xg%fields(ivar)%array
   write(*,*) "case sum O:",sum(O)
case ("O2")
  have_O2 = .true.
  write(*,*) 'have_O2=', have_O2
  O2 = xg%fields(ivar)%array
   write(*,*) "case sum O2:",sum(O2)
case ("air_pressure_at_surface")
  have_PS = .true.
  write(*,*) 'have_PS=', have_PS
  PS = xg%fields(ivar)%array
   write(*,*) "case sum PS:",sum(PS)
case ("surface_geopotential")
  have_PHIS = .true.
  write(*,*) 'have_PHIS=', have_PHIS
  PHIS = xg%fields(ivar)%array
   write(*,*) "case sum PHIS:",sum(PHIS)
case ("eastward_wind")
  have_U = .true.
  write(*,*) 'have_U=', have_U
  U = xg%fields(ivar)%array
  write(*,*) "case sum U:",sum(U)
case ("northward_wind")
  have_V = .true.
  V = xg%fields(ivar)%array
   write(*,*) "case sum V:",sum(V)
```

HofX application



computes simulated observations H(x) :
by interpolating model state variables to observation locations and applying the corresponding observation operators.

Current status:
implement HofX for radiosonde and ionosonde observations

(JEDI Academy)

HofX application

main/hofx.cc

```cpp
#include "camjedi/Traits.h"
#include "oops/runs/HofX3D.h"
#include "oops/runs/Run.h"
#include "ufo/instantiateObsErrorFactory.h"
#include "ufo/instantiateObsFilterFactory.h"
#include "ufo/ObsTraits.h"

int main(int argc, char ** argv) {
  oops::Run run(argc, argv);
  ufo::instantiateObsErrorFactory();
  ufo::instantiateObsFilterFactory();
  oops::HofX3D<camjedi::Traits, ufo::ObsTraits> hofx;
  return run.execute(hofx);
}
```

HofX ctest

```
# application
ecbuild_add_test(TARGET    camjedi_hofx_nomodel
                 MPI       6
                 ARGS      "testinput/hofx_nomodel.yaml"
                 COMMAND   camjedi_hofx.x
```

yaml input

nomodel: not propagated in time by model itself

```yaml
time window:
  begin: 2000-02-01T00:00:00Z
  length: PT1H
geometry:
  nml_file: "./modelinput/input_geo.nml"
  cam_file: "./testdata/FXSD_geovals_radiosonde.cam.h1.2018-04-15-00000.nc"
state:
  datetime: 2000-02-01T00:30:00Z
  state variables:
    #- T
    - air_temperature
    - O
    - O2
    - air_pressure_at_surface
    - surface_geopotential
    - specific_humidity
    - eastward_wind
    - northward_wind
    #- U
    - geometric_height
    - air_pressure
  filename: "./testdata/FXSD_geovals_radiosonde.cam.h1.2018-04-15-00000.nc"
observations:
  observers:
  - obs space:
      name: Radiosonde
      obsdatain:
        engine:
          type: H5File
          obsfile: "./testdata/sondes_obs_2018041500_m.nc4"
      obsdataout:
        engine:
          type: H5File
          obsfile: ./testdata/obsout_hofx3d_sondes.nc4
      simulated variables: [airTemperature,specificHumidity, windEastward, windNorthward]
    obs operator:
      name: VertInterp
      observation alias file: testinput/obsname.yaml
```

-ObsSpace(OBS): Metadata
-GeoVaLs(OBS): variables/location info
-ObsOperator(OBS): H simulate obs

-Observers(Model, OBS)
-Observations(OBS): H(x), QC, R, O-B

obsout.nc

Summary:

- Have started implementing a jedi interface to WACCMX
- And are now working on an H(x) application for radiosondes and ionosondes.

Next Step:

- H(x) application for radiosondes and ionosondes.

- integrating JEDI observation operators for meteor radar and GOLD observations into a 3DVar application with cam-jedi (SWORD Program task2.2 by Fazlul Laskar and Noah Peterson)

Next year

Complete initial WACCM-X/JEDI model interface and test single time-step applications using 3D-Var

the following year

Perform initial cycling DA experiments using ensemble methods (LETKF)