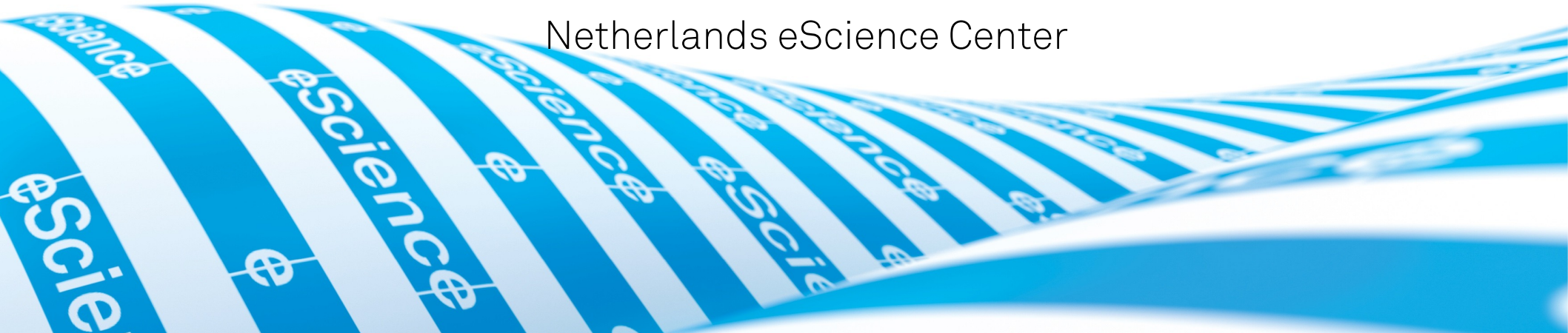
A model interface for ERA5

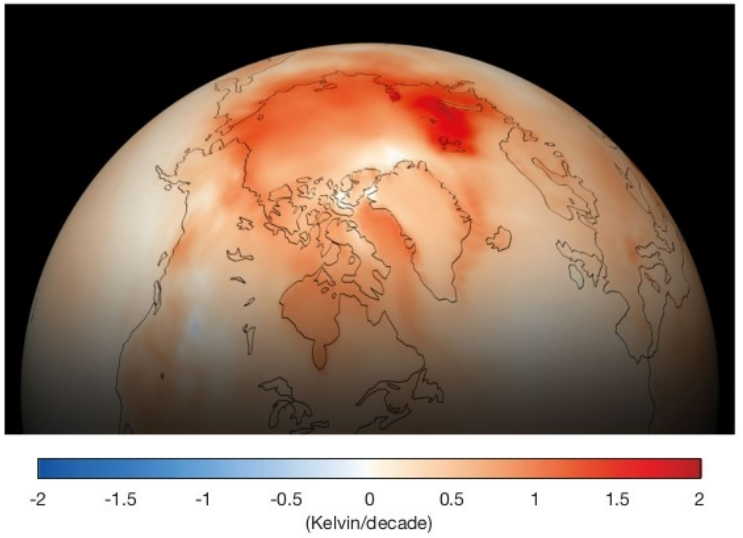
Inti Pelupessy (i.pelupessy@esciencecenter.nl)
Maria Chertova – Gijs van den Oord - Ben van Werkhoven

Netherlands eScience Center



Introduction: the ERA5 global reanalysis dataset

– Data product of the ECMWF
– global, homogeneous climate dataset
– 31 km resolution, 37 vertical levels, hourly
– 1979-present (soon: 1950-present)
– combination of:
IFS model +
observational data +
data assimilation
– available through the CDS climate store:



https://cds.climate.copernicus.eu/

the ERA5 dataset from a modellers perspective

ERA5 data is used as modelling resource to (a.o.):

- provide initial conditions to climate experiments
- set boundary conditions for regional simulations
- provide forcings to ocean or land surface models

problem: workflow to get ERA5 data into a model:
download data → convert → save into model format

This workflow is time consuming, error prone and generates a lot of intermediate data of low reuse value

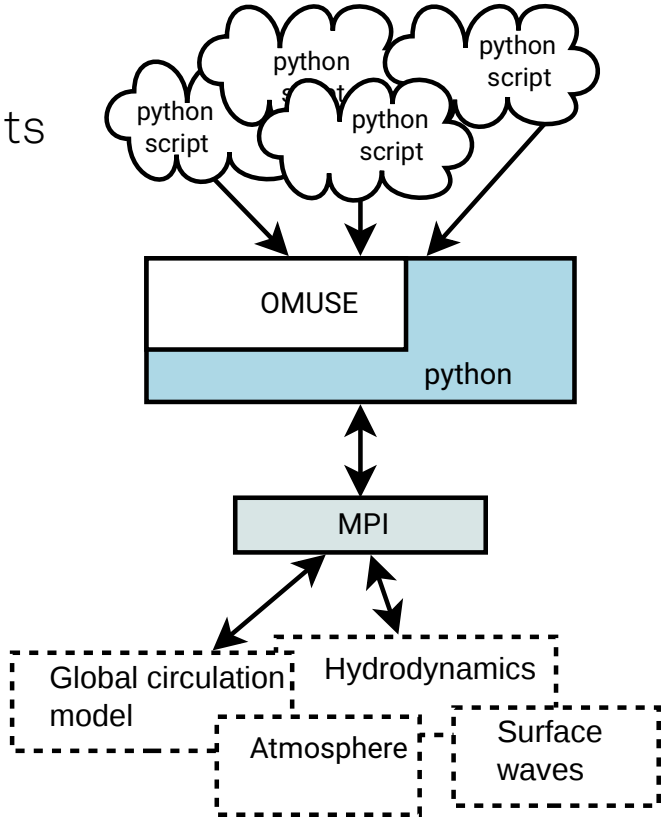
our solution:
provide an easier way to work with ERA5 by wrapping access to the data into a model-centric interface within the OMUSE framework for Earth System modelling.

What is OMUSE?

→ Oceanographic Multi-PUrpose Software Environment
→ OMUSE is a Python environment for numerical experiments in oceanography and other climate sciences
→ based on *AMUSE*

Goals:

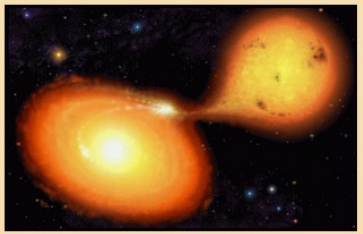
- provide a homogeneous enviroment to run *community* codes
- enable new code couplings and interactions between components
- facilitate multi-physics and multi-scale simulations



a short history of AMUSE & OMUSE

- AMUSE has its origins in the astrophysical MODEST community

- developed, with funding from NOVA, NWO and the NLeSC, at Leiden Observatory



MODEST

Modeling and Observing DEnse STellar systems

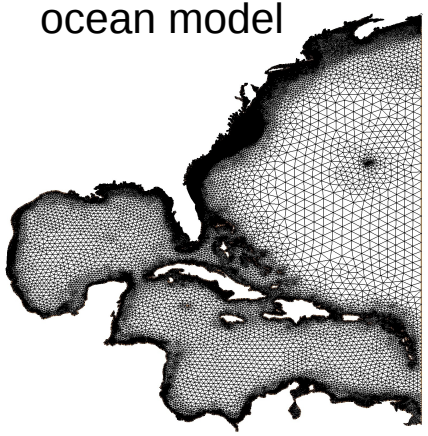
- actively being used by 15+ groups worldwide, 60+ publications, 8+ theses

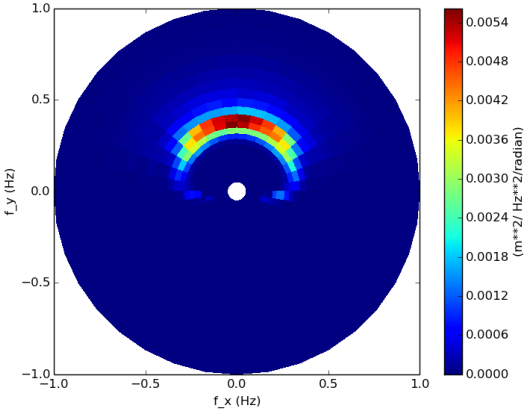
- 2014 – 2016: NLeSC project @IMAU (Pelupessy, van Werkhoven) to generalize this approach → OMUSE

- OMUSE currently being used in research: Southern Ocean, LES-GCM coupling in meteorology, exo-ocean (Titan) & the stochastic multiscale climate modelling NLeSC project

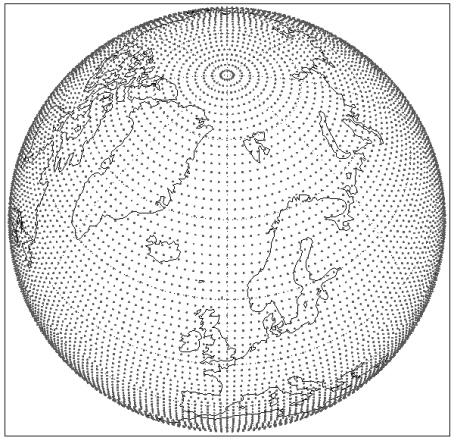
current codes in OMUSE

ADCIRC regional ocean model

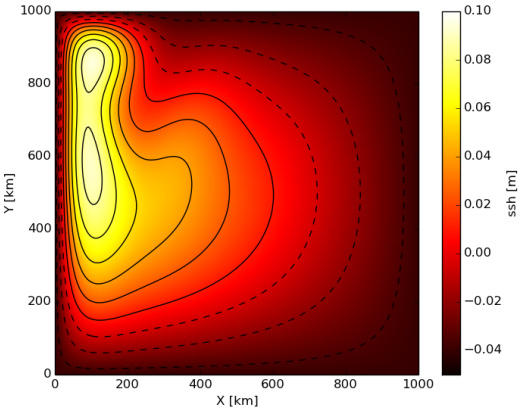




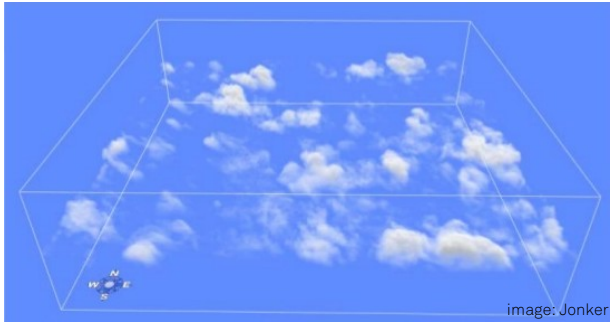
SWAN wave transport



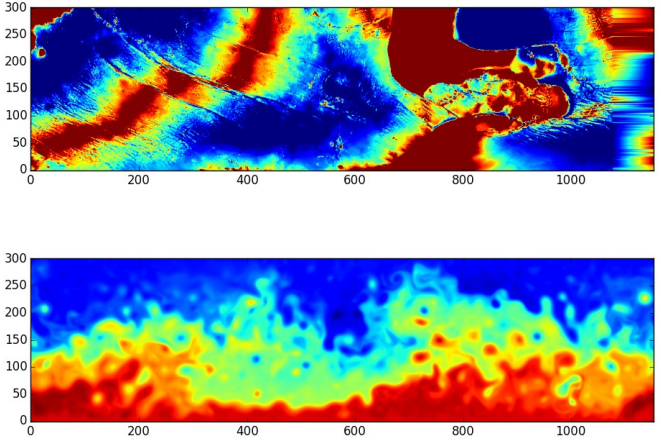
openIFS global atmosphere



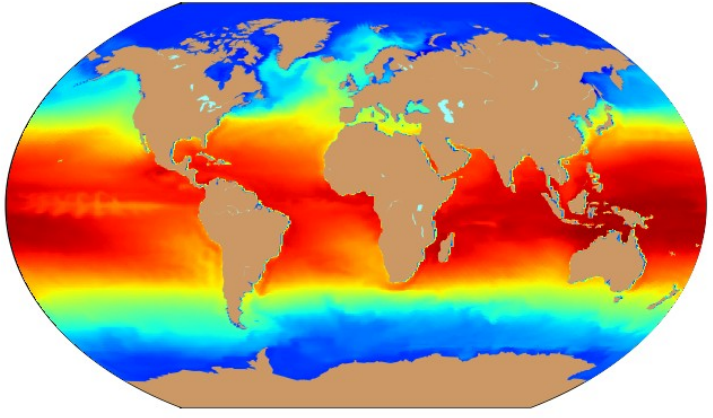
QGmodel



DALES cloud resolving model



QGCM



POP global ocean model

'Hello Ocean'

“imports”

```
from omuse.units import units
from omuse.community.qgmodel.interface import QGmodel
from amuse.io import read_set_from_file
```

“initial conditions”

```
input=read_set_from_file('initial_condition')
```

“instantiate code”

```
code=QGmodel()
```

“initialize model”

```
code.parameters.dt=0.5 | units.hour
code.grid.psi=input.psi
```

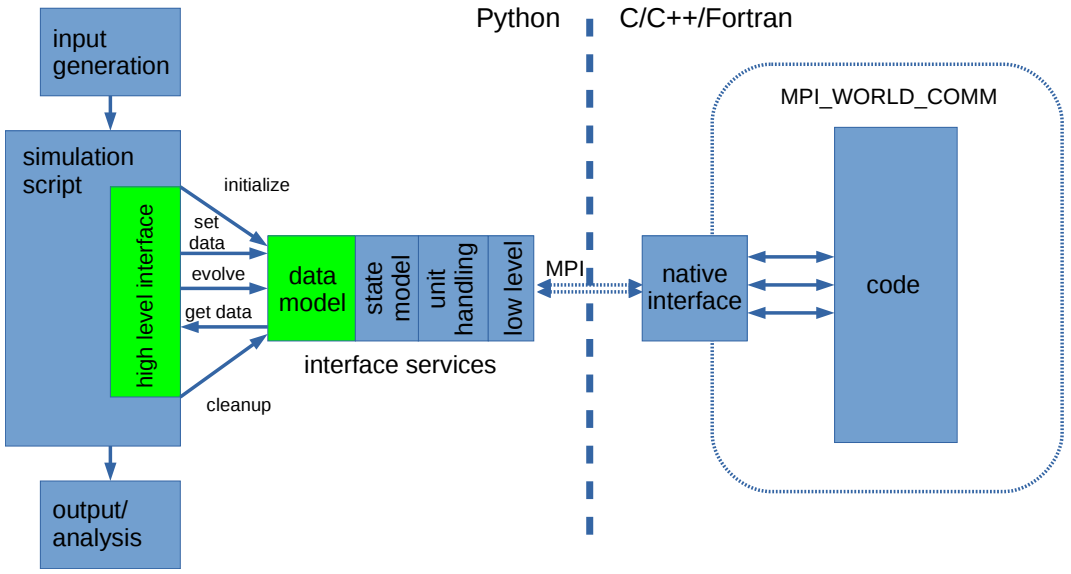
“evolve”

```
code.evolve_model(1.| units.day)
```

“analysis”

```
print code.grid.psi.max().in_(units.Sv/units.km)
```

The OMUSE interface: a bird's eye view



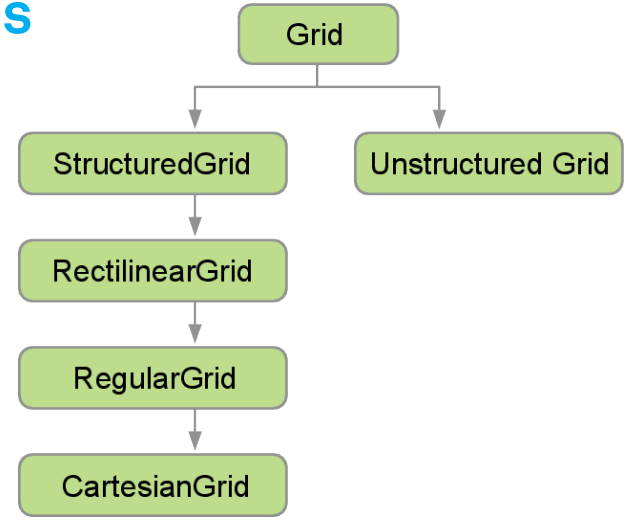
– OMUSE is based on remote code interfaces

– codes run in parallel & calls can be asynchronous

– OMUSE provides a number of *interface services:* unit handling, data structures, code state handling etc..

– the user is presented with uniform high level interfaces, which integrates into a scientific python workflow

OMUSE data structures

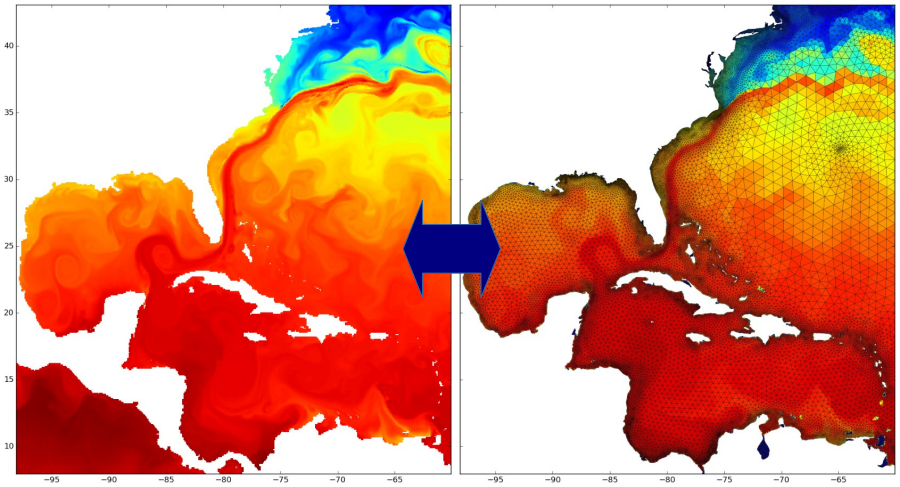


– high level interface uses grid data structures

– grids have domain specific attributes

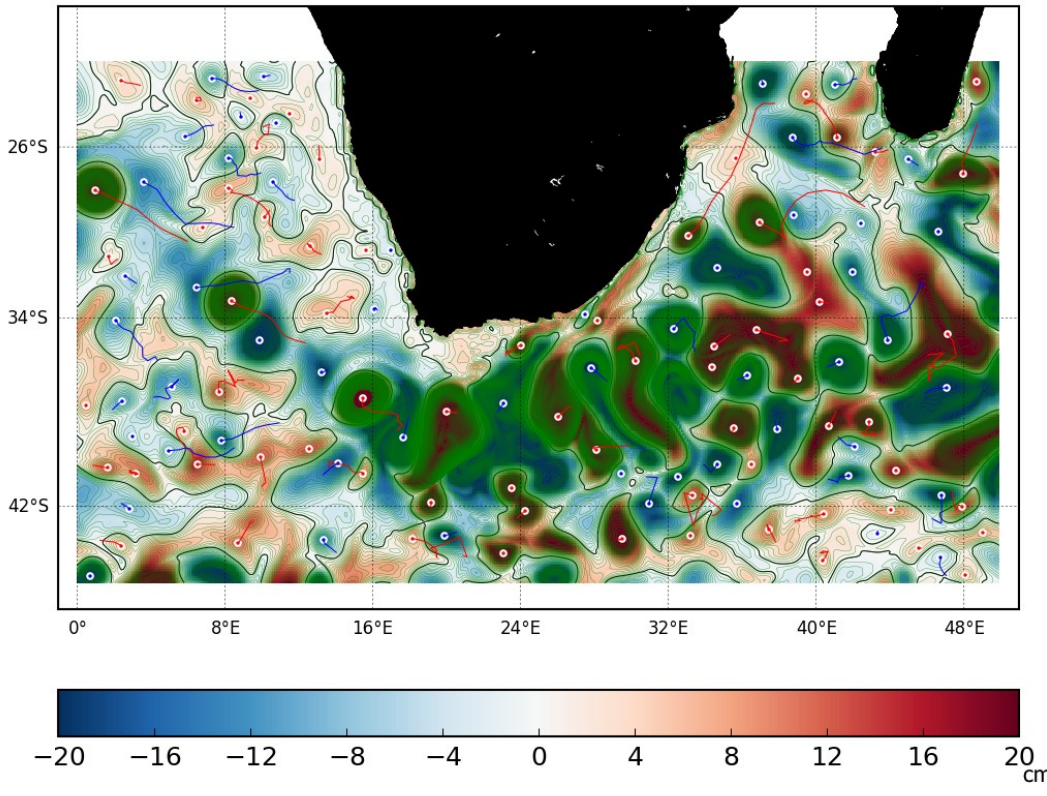
– OMUSE improved grid support:

  – different grid types
  – add grid remapping channels
  – add grid (functional) transform
  – (some) grid generation routines
  – improved support for large parameter sets and files

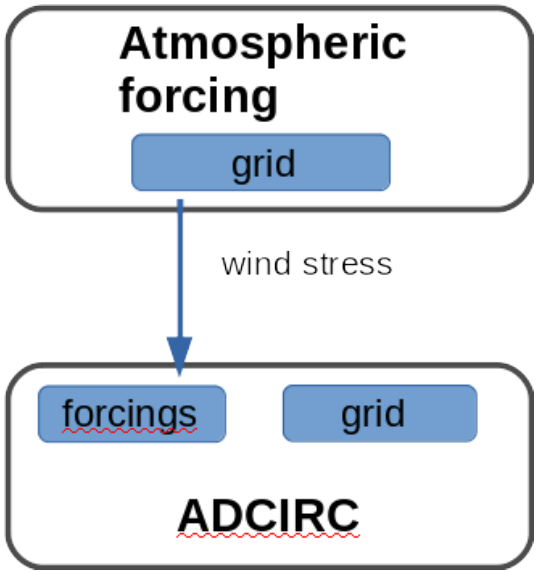


What can you do with OMUSE?

- simplify model setup and runs,
- scripting:
  - event detection,
  - stopping conditions
- 'online' data analysis
- ensemble simulations:
  - parameters searches
  - optimizations (e.g. MCMC)
  - data assimilation
- model comparison: running problems with different codes and methods
- coupling different codes to construct new solvers

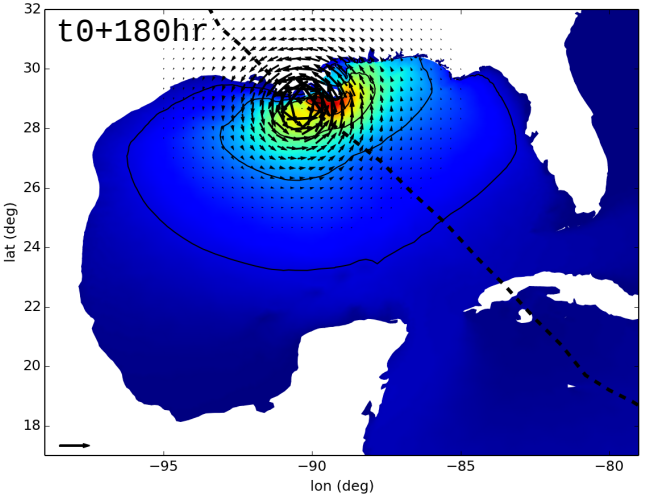


Code couplings with AMUSE

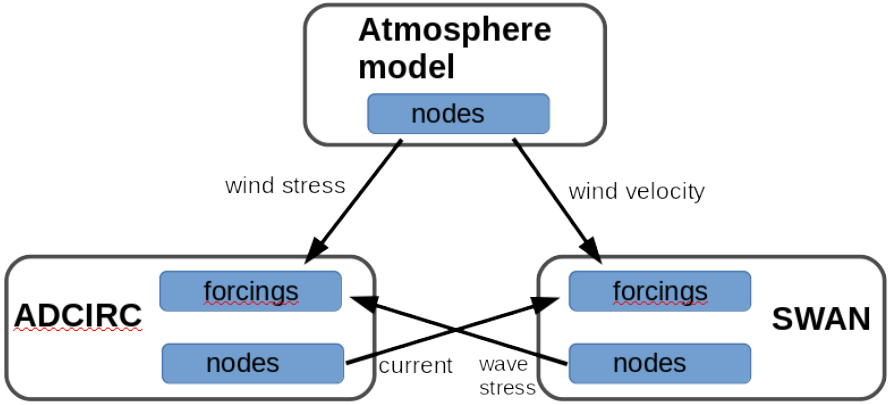



```
adcirc = Adcirc()

meteo = netcdf_meteo( "windstress_2001to2010.nc")

channel = meteo.grid.new_channel_to( adcirc.forcings )

while adcirc.model_time<tend:
    meteo.evolve_model( adcirc.model_time + dt )
    channel.copy_attributes( ["tau_x", "tau_y"] )
    adcirc.evolve_model( adcirc.model_time + dt )
```

Hurricane Gustav with a coupled hydrodynamic/ wave model



(Pelupessy+ 2017)




```
channel1=hurricane.grid.new_channel_to( swan.forcings )
channel2=hurricane.grid.new_channel_to( adcirc.forcings )
channel3=adcirc.nodes.new_channel_to( swan.forcings )
channel4=swan.nodes.new_channel_to( adcirc.forcings )
while time<tend:
    hurricane.evolve_model(time+dt/2)
    channel1.copy_attributes(["vx","vy"])
    channel2.copy_attributes(["tau_x","tau_y"])
    adcirc.evolve_model(time+dt/2)
    swan.evolve_model(time+dt/2)
    channel3.copy_attributes(["current_vx","current_vy"])
    channel4.copy_attributes(["wave_tau_x","wave_tau_y"])
    time+=dt
```

The OMUSE ERA5 component

Python module

```
from omuse.community.era5.interface import ERA5
```

Features:

– provides a model-centric view on ERA5 data
– downloads data as needed
– caches the data
– interface follows standards of an OMUSE community code
– data is presented in an OMUSE grid high level data structure, with units

→ minimal difference between coupling a model to static ERA5 data and a life model.

User guide

Installation

```
pip install omuse-era5
```

(assuming **cdsapi** has been configured,
more advance use requires a development install of omuse)

example use:

```
from omuse.community.era5.interface import ERA5

era5=ERA5(variables=["2m_temperature", "total_precipitation"],
          invariate_variables=["land_sea_mask", "orography"],
          nwse_boundingbox=[70, -15, 40, 15] | units.deg,
          grid_resolution=1.0 | units.deg,
          start_datetime=datetime.datetime(1979,1,2) )

era5.evolve_model(1. | units.day)        ← evolve the “model”
t2=e.grid._2m_temperature                ← note the leading ‘_’
```

Caveats & Future work

– ERA5 is not a model...
  → no guarantees on local or global constraints
– view of the dataset is time-slice based (evolve_model)
– currently limited to 2D fields
– accumulations not handled

apart from these shortcomings, need work on:

– more intelligent/efficient data handling
– cache maintenance
– prefetching
– other datasets

netherlands eScience center

Resources

OMUSE-ERA5 PyPI page
pypi.org/project/omuse-era5/ omuse-era5

OMUSE repository:
github.com/omuse-geoscience/omuse

OMUSE example scripts:
github.com/omuse-geoscience/omuse-examples

OMUSE Code paper:
Pelupessy et al. 2017, GMD 10, 3167