Mesh Generation and Design in COMPASS

Mark Petersen and Xylar Asay-Davis
COSIM meeting, May 20, 2020

Major technical contributions by
Doug Jacobsen, Darren Engwirda,
Steven Brus, Phillip Wolfram, Matt Hoffman
MPAS instructions: Clone repo, compile

# Set USERNAME variable
USERNAME=`whoami` # bash
set USERNAME=`whoami` # tcsh (c-shell)
echo $USERNAME

# on grizzly. Clone repo, checkout branch, update submodule:
cd /lustre/scratch4/turquoise/$USERNAME
module load git
git clone git@github.com:MPAS-Dev/MPAS-Model.git
cd MPAS-Model/
git checkout -b ocean/develop origin/ocean/develop
git submodule update --init

# load gnu modules, python package for compass
source /usr/projects/climate/SHARED_CLIMATE/anaconda_envs/load latest compass.sh # for bash
source /usr/projects/climate/SHARED_CLIMATE/anaconda_envs/load latest compass.csh # for c-shell
module use /usr/projects/climate/SHARED_CLIMATE/modulefiles/all/
module load gcc/5.3.0 openmpi/1.10.5 netcdf/4.4.1 parallel-netcdf/1.5.0 pio/1.7.2

# compile MPAS-Ocean with gnu (3 minutes)
make gfortran CORE=ocean

# instructions for other platforms here on confluence, and downloaded version here.
Overview: Steps in Mesh Generation for MPAS in COMPASS: Configuration of MPAS Setups

1. define base mesh
cell width as function of latitude, longitude

2. base mesh
jigsaw creates global spherical mesh

3. culled mesh
remove land cells from spherical mesh

4. initial state
add temp., salinity, layers, bathymetry

These images from COMPASS CUSP12 (Coastal US Plus, 12 km) init case.

These steps create the MPAS-Ocean and MPAS-Seaice mesh. MPAS-Landice also uses COMPASS.
COMPASS Instructions: list test cases

# Set USERNAME variable
USERNAME=`whoami` # bash
set USERNAME=`whoami` # tcsh (c-shell)
echo $USERNAME

# Go to your MPAS repository and load modules
cd /lustre/scratch4/turquoise/$USERNAME/MPAS-Model
cd testing_and_setup/compass/
source /usr/projects/climate/SHARED_CLIMATE/anaconda_envs/load_latest_compass.sh # for bash
source /usr/projects/climate/SHARED_CLIMATE/anaconda_envs/load_latest_compass.csh # for c-shell

# list all test cases
./list_testcases.py

# list all test cases with resolution of QU240
./list_testcases.py -r QU240

# list all initialization cases
./list_testcases.py -t init
COMPASS Instructions: set up QU240 init case

# You need to set up a config.ocean file, the first time only. For today, just use mine:
cp /users/mpeterse/share/config.ocean.training .

# You can change the paths to your repos later if you would like.

# list all initialization cases
./list_testcases.py -t init

# we will use this one:
   [74]: -o ocean -c global_ocean -r QU240 -t init

# set up QU240 init case
./setup_testcase.py \
   --config_file config.ocean.training \ 
   --model_runtime runtime_definitions/mpirun.xml \ 
   --work_dir /lustre/scratch4/turquoise/$USERNAME/compass_training \ 
   --case_number 74
COMPASS Instructions: run test case

# Get compute nodes
salloc -N 1 -t 1:0:0 --qos=interactive

# load gnu modules, python package for compass
source /usr/projects/climate/SHARED_CLIMATE/anaconda_envs/load_latest_compass.sh # for bash
source /usr/projects/climate/SHARED_CLIMATE/anaconda_envs/load_latest_compass.csh # for c-shell
module use /usr/projects/climate/SHARED_CLIMATE/modulefiles/all/
module load gcc/5.3.0 openmpi/1.10.5 netcdf/4.4.1 parallel-netcdf/1.5.0 pio/1.7.2

# Go to the work directory
cd /lustre/scratch4/turquoise/$USERNAME/compass_training
cd ocean/global_ocean/QU240/init

# Run the case (all four steps)
./run.py

# This created the files:
base_mesh/cellWidthVsLatLon.nc  culled_mesh/culled_mesh.nc
base_mesh/base_mesh.nc          initial_state/initial_state.nc
1. define base mesh: cell width as function of latitude, longitude

# Run just the base mesh portion
cd base_mesh
./run.py

define_base_mesh.py creates cellWidth:

def cellWidthVsLatLon():
    ddeg = 10.0
    constantCellWidth = 240

    lat = np.arange(-90, 90.01, ddeg)
    lon = np.arange(-180, 180.01, ddeg)

    cellWidth = constantCellWidth * np.ones((lat.size, lon.size))
    return cellWidth, lon, lat

Files created:
cellWidthVsLatLon.nc
cellWidthGlobal.png
2. base mesh: Jigsaw creates a global spherical mesh

```python
# Run, same as step 1
cd base_mesh
./run.py
```

Jigsaw takes the cellWidth array as an input, and creates an unstructured mesh - a spherical centroidal Voronoi Tessellation. We call Jigsaw through a python interface.

**Intermediate files produced:**
- mesh-HFUN.msh
- mesh.log
- mesh.msh
- mesh.jig
- mesh-MESH.msh
- mesh_triangles.nc

**Final files produced:**
- base_mesh.nc
- base_mesh_vtk

![cell width, from base_mesh.nc](image)
3. culled mesh: remove land cells from global mesh

# Run this step only:
cd culled_mesh
./run.py

This step includes many smaller steps:
1. Define coast with coastline data set
2. Remove inland seas
3. Open critical passages
4. Close critical land blockages
5. Remove narrow (single cell) river outlets
6. Add land ice cavities if requested

These steps make use of two other repositories:
https://github.com/MPAS-Dev/MPAS-Tools
https://github.com/MPAS-Dev/geometric_features

Final files produced:
culled_mesh.nc
3. culled mesh: remove land cells from global mesh

The steps use line segments in geojson files in the geometric features repo:
1. Open critical passages
2. Close critical land blockages
4. initial state: add temperature, salinity, layers, bathymetry

# Run this step only:
cd initial_state
./run.py

Gridded initial condition data is automatically downloaded from https://web.lcrc.anl.gov/public/e3sm (full link) to path specified in your COMPASS config.ocean file:

```
initial_condition_database =
```

so it is only downloaded once.

Final files produced:
```
initial_state.nc
initial_state.png
vertical_grid.png
```
4. initial state: add temperature, salinity, layers, bathymetry

These plots are auto-generated, so the user can verify that the initial condition was created properly.
COMPASS Instructions: set up CUSP test case

# Find CUSP test case (Coastal US Plus)
```bash
cd /lustre/scratch4/turquoise/$USERNAME/MPAS-Model/testing_and_setup/compass/
./list_testcases.py | grep CUSP
  65: -o ocean -c global_ocean -r CUSP12 -t init
  66: -o ocean -c global_ocean -r CUSP12 -t spin_up
```

# Set up CUSP
```bash
./setup_testcase.py \
  --config_file config.ocean.training \
  --model_runtime runtime_definitions/mpirun.xml \
  --work_dir /lustre/scratch4/turquoise/$USERNAME/compass_training \
  --case_number 65
```
```
cd /lustre/scratch4/turquoise/$USERNAME/compass_training/ocean/global_ocean/CUSP12/init
```

# Directories include the steps we just discussed
```
base_mesh culled_mesh initial_state_state run.py
```
How do we customize refined resolution?

```
import mdt

# Process begins with a background mesh:
EC60to30 = mdt.EC_CellWidthVsLat(lat)

# Then a sequence of steps that adds higher resolution using these files:
coastline_CUSP.geojson
land_mask_Mexico.geojson
region_Bering_Sea.geojson
region_Gulf_of_Mexico.geojson
land_mask_Kamchatka.geojson
region_Arctic_Ocean.geojson
region_Central_America.geojson
region_Gulf_Stream_extension.geojson
```

Base mesh construction process:

```
cd base_mesh
# Once it gets to Jigsaw, you can hit control-c
vi define_base_mesh.py
# Change dlon to 1.0 degrees to make it run faster:
30     dlon = 1.0
./run.py
# Once it gets to Jigsaw, you can hit control-c
vi define_base_mesh.py
```
How do we customize refined resolution?

Single step to add higher resolution region with `coastline_CUSP.geojson` region file.

```python
68     fileName = 'coastline_CUSP'
69     distanceToTransition = 600.0*km
71     transitionWidth = 600.0*km
```

This uses a signed distance function, to control transition at some distance from the region.

For coastal masks from high-resolution coastlines, Steve Brus and Phillip Wolfran have made a tool set. See hurricane test cases (USDEQU120at30cr10rr2)
How do we customize refined resolution?

It is easy to alter and add these high resolution regions! Load .geojson on [http://geojson.io](http://geojson.io) for GUI.

You can easily move and add points to the region definition by hand.

Or load a data set into `define_base_mesh.py` to define regions based on data contours.
Mesh discussion and approval process

A new mesh is posted as a pull request on the MPAS-Model repo. Discussions and alterations are on this PR, until reviewers agree. That mesh, with revision number, is frozen, and continues with E3SM testing. Any revisions to mesh gets a new PR and new revision number.

Naming convention, short name:

SOwISC12to60E2r02
SO type of mesh (Southern Ocean)
wISC special tag (with Ice Shelf Cavities)
12to60 resolution span (12 to 60 km cells)
L64 number of vertical levels (64)
E2 E3SM version 2
r02 revision number 2

long name:

SOwISC12to60kmL64E3SMv2r02
Automated testing suite for MPAS-Ocean

Regression testing packages together a number of test cases, and can provide:

- pass/fail status
- bit-for-bit comparison on output of previous test
- performance comparison with previous test

```bash
cd /lustre/scratch4/turquoise/$USERNAME/MPAS-Model/testing_and_setup/compass

./manage_regression_suite.py --setup \
    --test_suite ocean/regression_suites/nightly.xml \ 
    --config_file config.ocean.training \ 
    --model_runtime runtime_definitions/mpirun.xml \ 
    --work_dir /lustre/scratch4/turquoise/$USERNAME/compass_training \ 

# on compute node, with modules loaded:
cd /lustre/scratch4/turquoise/$USERNAME/compass_training
./nightly_ocean_test_suite.py
```
Additional steps to run in E3SM

1. define base mesh cell width as function of latitude, longitude
2. base mesh jigsaw creates global spherical mesh
3. culled mesh remove land cells from spherical mesh
4. initial state add temp., salinity, layers, bathymetry
5. spin up ocean only in MPAS-Ocean, to get up to normal time step. COMPASS cases have spin_up sets for this.
6. E3SM coupling files Detailed list here. Many files are autogenerated within COMPASS in init/e3sm_coupling
7. Add case to E3SM scripts see example