Using MOM6

Alistair Adcroft

GFDL MOM6 team includes Raphael Dussin, Robert Hallberg, Stephen Griffies, Matthew Harrison, Hae-Cheol Kim, John Krasting , Marshall Ward, Niki Zadeh

Other contributors for presentation: Kate Hedstrom, Andrew Shao

Outline

- "Using MOM6"
  - "Where to start?"
- Every application is unique
  - Sometimes requires custom code specific to that one configuration
- Building new configurations
  - Best to start from working example

1. Getting code/compiling/running
2. Controlling
   - Parameters
   - Diagnostics
3. Examples
   - Places to start
4. Model/repository structure

“Getting started” (Cloning, compiling, running)

- “MOM6 wiki”
  - https://github.com/NOAA-GFDL/MOM6-examples/wiki
  - (first result on google)
- Instructions for
  - Cloning (obtaining code)
  - Compiling (fairly portable)
  - Running (fairly standard)
- User-contributed
- Instructions assume some familiarity with linux & models
- Best limited to stand-alone ocean-only configurations
  - Coupled models are so much more complicated
- Does not cover working within an environment like GFDL’s FRE or CESM’s CIME
  - just low-level basics

Model input: run-time parameters

- Parameter syntax is key, value pairs

```
KH = 25.
```

  - Self-documenting runs

```
KH = 25.0                       !   [m2 s-1] default = 0.0
                                ! The background Laplacian horizontal viscosity.
```

  - Simple API

```
    call get_param(param_file, mdl, "KH", Kh,                       &
                   "The background Laplacian horizontal viscosity.", &
                   units = "m2 s-1", default=0.0, scale=US%m_to_L**2*US%T_to_s)
```

  - MOM6 always writes out
    - `MOM_parameter_doc.all` (everything)
    - `MOM_parameter_doc.short` (non-defaults)

- Bootstrapped parameter parser
  - namelist in input.nml

```
&MOM_input_nml
        output_directory = '.',
        input_filename = 'n'
        restart_input_dir = 'INPUT',
        restart_output_dir = 'RESTART',
        parameter_filename = 'MOM_input',
                             'MOM_saltrestore',
                             'MOM_override'
/
```

- Typical setup
  - Baseline uses blank `MOM_override`
  - Perturbation runs concisely contained in `MOM_override`
- Lots of error checking

Parallel decomposition

- Reported in `MOM_parameter_doc.layout`

```
NIHALO = 4        ! default = 4
                  ! The number of halo points on each side in the x-direction.  With
                  ! STATIC_MEMORY_ this is set as NIHALO_ in MOM_memory.h at compile time; without
                  ! STATIC_MEMORY_ the default is NIHALO_ in MOM_memory.h (if defined) or 2.
NJHALO = 4        ! default = 4
                  ! The number of halo points on each side in the y-direction.  With
                  ! STATIC_MEMORY_ this is set as NJHALO_ in MOM_memory.h at compile time; without
                  ! STATIC_MEMORY_ the default is NJHALO_ in MOM_memory.h (if defined) or 2.
NIPROC = 2        !
                  ! The number of processors in the x-direction. With STATIC_MEMORY_ this is set
                  ! in MOM_memory.h at compile time.
NJPROC = 4        !
                  ! The number of processors in the y-direction. With STATIC_MEMORY_ this is set
                  ! in MOM_memory.h at compile time.
LAYOUT = 2, 4     !
                  ! The processor layout that was actually used.
IO_LAYOUT = 1, 1  ! default = 1
                  ! The processor layout to be used, or 0,0 to automatically set the io_layout to
                  ! be the same as the layout.
```

- Bitwise reproduces across layout
- Optimal tile size ~ 12x12-30x30
  - Tile size = `NIGLOBAL/NIPROC,NJGLOBAL/NJPROC`
- Halos ~3-4 most typically needed
  - Parameters `NIHALO,NJHALO`
  - Without high-order advection and certain choices of time-stepping
- Tile dimensions should be >= `NIHALO,NJHALO`
- Tiles may not be uniform!
  - Different sized tiles are allowed

Restarts

- Many runs take longer than a single job submission
- Bitwise reproducibility across a restart boundary
- Online time-averaged diagnostics are not included in restarts
  - handled by FMS framework
- Input data usually read from INPUT/
- Diagnostic output is in current directory
- Restart files generally written to RESTART/
- Restart files are read from INPUT/

Controlling diagnostics

- FMS diag_manager parses

  `diag_table`

- File definition

```
"ocean_daily",           1, "days",   1, "days", "time"
"ocean_month_snap",      1, "months", 1, "days", "time"
"ocean_month",           1, "months", 1, "days", "time"
"ocean_month_z",         1, "months", 1, "days", "time"
"ocean_annual",         12, "months", 1, "days", "time"
"ocean_annual_z",       12, "months", 1, "days", "time"
"ocean_scalar_month",    1, "months", 1, "days", "time"
"ocean_scalar_annual",  12, "months", 1, "days", "time"
"ocean_static",         -1, "months", 1, "days", "time"
```

- Variable lists per file/module

```
  "ocean_model",    "thetao",       "thetao",          "ocean_annual",       "all", "mean", "none",2  # if use pre-TEOS10
#"ocean_model",    "thetao",       "thetao",          "ocean_month",        "all", "mean", "none",2  # if use pre-TEOS10
  "ocean_model_z", "thetao",       "thetao",          "ocean_annual_z",     "all", "mean", "none",2  # if use pre-TEOS10
  "ocean_model_z", "thetao",       "thetao",          "ocean_month_z",      "all", "mean", "none",2  # if use pre-TEOS10
  "ocean_model_z", "thetao_xyave", "thetao_xyave",    "ocean_annual_z",     "all", "mean", "none",2  # if use pre-TEOS10
```

- MOM6 wraps diag_manager
  - Registers same diagnostic in multiple vertical coordinates, multiple names, xy-averages
- Available diagnostics written by MOM6 at run-time

  `available_diags.0000`

- `Regional diagnostics`

MOM6-examples

```
MOM6-examples
├── coupled_AM2_LM3_SIS
├── coupled_AM2_LM3_SIS2
├── ice_ocean_SIS
├── ice_ocean_SIS2
│   ├── Baltic_OM4_025
│   ├── Baltic_OM4_05
│   ├── OM4_025
│   ├── OM4_05
│   └── SIS2_icebergs
├── ocean_only
│   ├── CVmix_SCM_tests
│   ├── DOME
│   ├── ISOMIP
│   ├── Phillips_2layer
│   ├── SCM_idealized_hurricane
│   ├── adjustment2d
│   ├── benchmark
│   ├── buoy_forced_basin
│   ├── circle_obcs
│   ├── double_gyre
│   ├── external_gwave
│   ├── flow_downslope
│   ├── global
│   ├── global_ALE
│   ├── idealized_hurricane
│   ├── lock_exchange
│   ├── mixed_layer_restrat_2d
│   ├── nonBous_global
│   ├── resting
│   ├── rotating_gravity_current
│   ├── seamount
│   ├── single_column
│   ├── sloshing
│   ├── tides_025
│   ├── torus_advection_test
│   ├── tracer_mixing
│   └── unit_tests
├── src
│   ├── FMS
│   ├── MOM6
│   ├── SIS2
│   ├── atmos_null
│   ├── coupler
│   ├── ice_param
│   ├── icebergs
│   ├── land_null
│   └── mkmf
└── tools
├── analysis
├── matlab
├── python
└── tests
```

ice-ocean configurations

ocean only configurations

sub-modules to source

- **`double_gyre`**
  - Wind driven gyre using stacked shallow water equations
- **`Phillips_2layer`**
  - Idealized channel model
- **`flow_downslope`**
  - adjustment problem over topography using different coordinates
- **`OM_05`**
  - 1/2° ice-ocean global model

ocean_only/double_gyre

Visualizing and animating sea-surface height in the double-gyre example.ipynb

Raw

We will use `matplotlib.pyplot` for plotting and scipy's netcdf package for reading the model output. The `%pylab inline` causes figures to appear in the page and conveniently alias pyplot to `plt` (which is becoming a widely used alias).

This analysis assumes you changed DAYMAX to some multiple of 5 so that there are multiple time records in the model output.

To see this notebook with figures, see https://gist.github.com/adcroft/2a2b91d66625fd534372.

In [1]:
```
%pylab inline
import scipy.io.netcdf
```

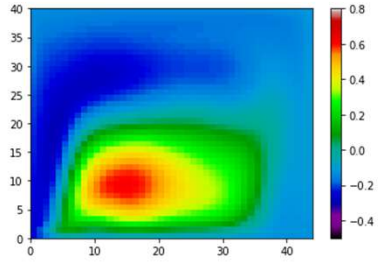
```
Populating the interactive namespace from numpy and matplotlib
```

We first create a netcdf object, or "handle", to the netcdf file. We'll also list all the objects in the netcdf object.

In [2]:
```
prog_file = scipy.io.netcdf_file('prog__0001_006.nc')
prog_file.variables
```

Out[2]:
```
{'Time': <scipy.io.netcdf.netcdf_variable at 0xb60364ac>,
 'e': <scipy.io.netcdf.netcdf_variable at 0xb603686c>,
 'h': <scipy.io.netcdf.netcdf_variable at 0xb603676c>,
 'u': <scipy.io.netcdf.netcdf_variable at 0xb603668c>,
 'v': <scipy.io.netcdf.netcdf_variable at 0xb603670c>,
 'xh': <scipy.io.netcdf.netcdf_variable at 0xb603648c>,
 'xq': <scipy.io.netcdf.netcdf_variable at 0xb603634c>,
 'yh': <scipy.io.netcdf.netcdf_variable at 0xb603632c>,
 'yq': <scipy.io.netcdf.netcdf_variable at 0xb603652c>,
 'zi': <scipy.io.netcdf.netcdf_variable at 0xb60365ec>,
 'zl': <scipy.io.netcdf.netcdf_variable at 0xb60363cc>}
```

In [10]:
```
for n in range( e_handle.shape[0]):
    display.display(plt.gcf())
    plt.clf()
    plot_ssh(n)
    display.clear_output(wait=True)
```



- jupyter notebook
  - (sorry about rainbow colormap)
  - This one uses matplotlib and scipy
- Much more needs to be added
  - Notebook for other exampels
  - Others use netCDF4 instead of scipy
  - Will add xarray, seaborn examples
- Very much NOT advocating for one analysis system/style

ocean_only/flow_downslope

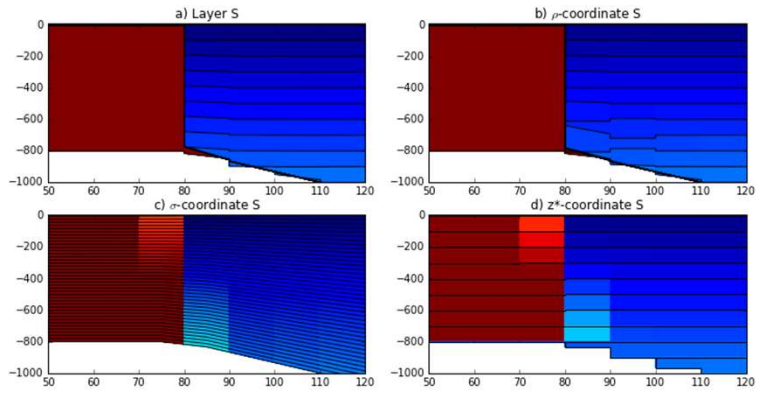
Understanding native output data from MOM6.ipynb

Raw

This "flow downslope" example involves four sub-directories, `layer`, `rho`, `sigma` and `z`, in which the model is running in one of four coordinate configurations. To use this notebook it is assumed you have run each of those experiments in place and have kept the output generated in default configuration.

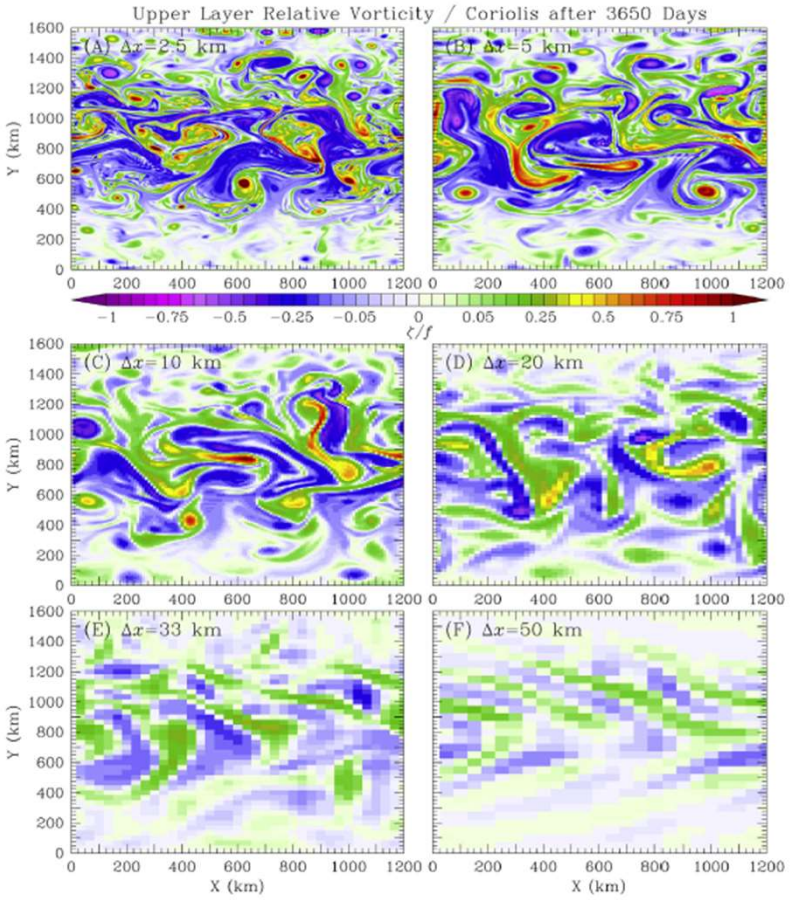
CAVEAT: This is a tutorial of how to make vertical section plots which also illustrates some poor ways to plot data for comparison. Read through to the end.

```
plt.figure(figsize=(12,6))
plt.subplot(2,2,1); plot_section(layer_file, 0, xq, xl=xxl, yl=yl, clim=(34,35), rep='plm'); plt.title('a) Layer S');
plt.subplot(2,2,2); plot_section(rho_file, 0, xq, xl=xxl, yl=yl, clim=(34,35), rep='plm'); plt.title(r'b) $\rho$-coordinate S');
plt.subplot(2,2,3); plot_section(sigma_file, 0, xq, xl=xxl, yl=yl, clim=(34,35), rep='linear'); plt.title(r'c) $\sigma$-coordinate S');
plt.subplot(2,2,4); plot_section(z_file, 0, xq, xl=xxl, yl=yl, clim=(34,35), rep='pcm'); plt.title('d) z*-coordinate S');
```

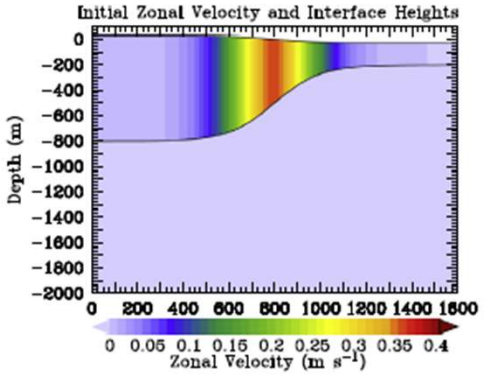


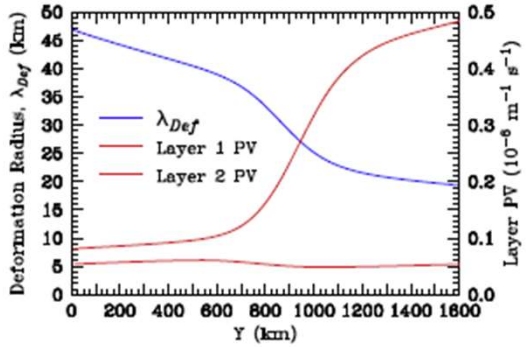
- 2d density current
- Notebook is a treatise on how plotting in the vertical can go wrong
  - Idea is to explain how the model stores data in the vertical
    - … and how to look at the vertical in native space

ocean_only/Phillips_2layer

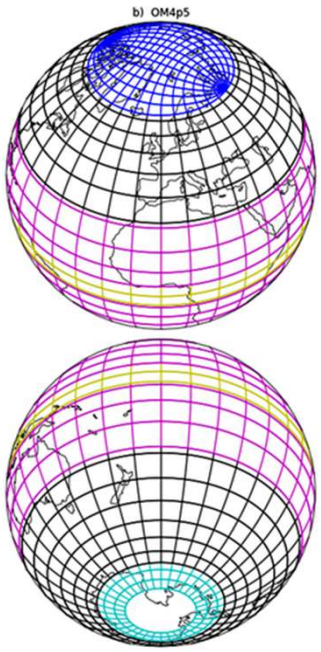


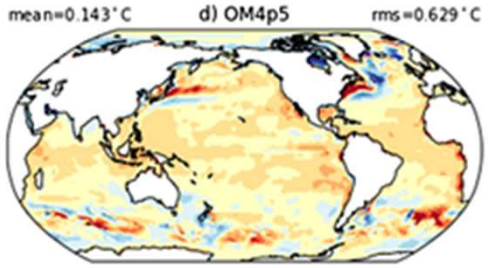
- Hallberg, 2013
- Idealized zonal channel
  - Customized forcing
- No jupyter notebooks yet
  - plots were done with Ferret

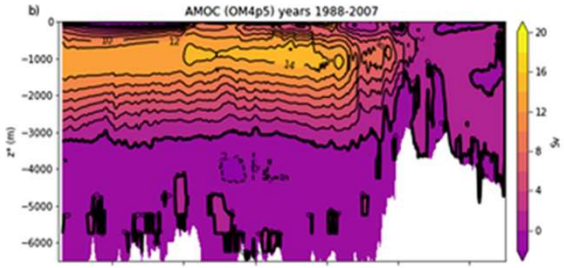


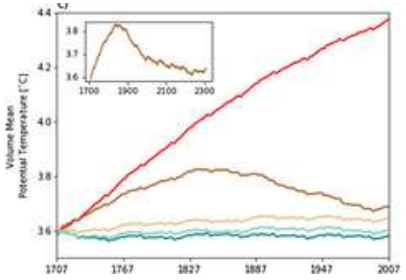


ice_ocean_SIS2/OM4_05







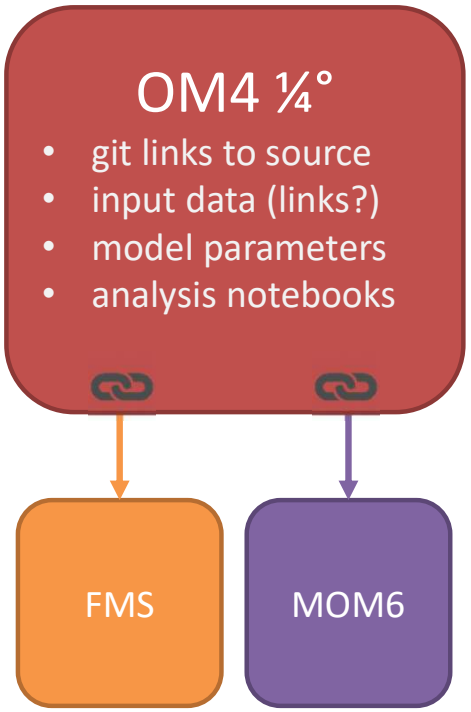


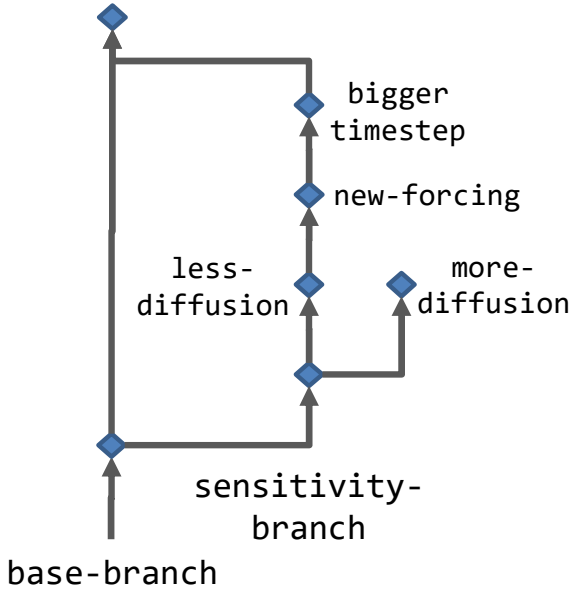
- ½° global ice-ocean model
  - Uses GFDL SIS2 sea-ice model and GFDL coupler
- Uses GFDL vertical physics
  - ePBL, JHL, …
- Non-eddying (coarse resolution)
- Uses GM and neutral-diffusion parameterizations

Adcroft et al., 2019

Experiment oriented repositories

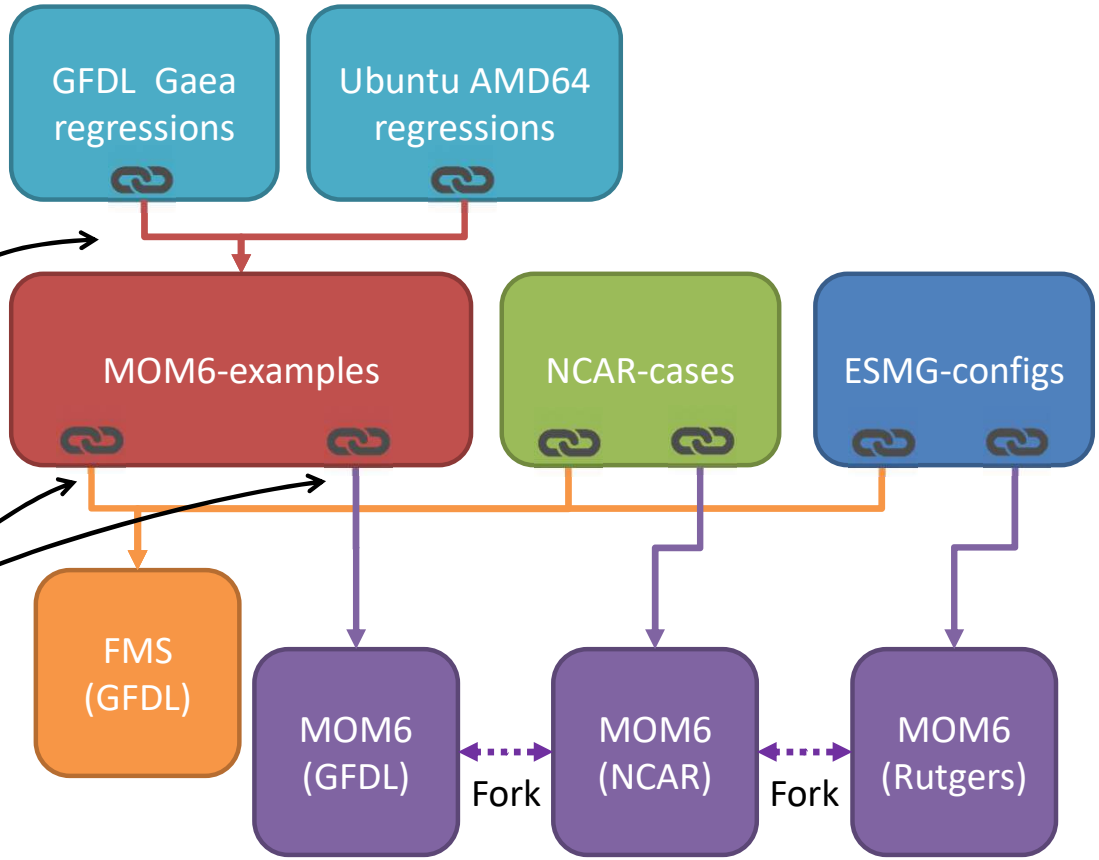
- Using a repository for experiment development
  - treating configurations like code
- No new tools (just `git`)
- Provides history of experiment design
  - Recoverable / reproducible
- Used in other workflows
  - e.g. Payu, ROMS





Repository organization (experiment suites)

- Layered repositories using sub-modules
  - Regression results
    - Output from regression tests
      - Platform dependent
    - Records specific version of configurations
  - Configurations
    - Input files (parameters)
    - Records specific versions of source
      - Including URLs (for forks)
  - Source for MOM6, FMS, SIS2, …
    - Pure source code (+ packages)



Code tree: driver layer

```
MOM6/
├── config_src/
│   ├── coupled_driver/
│   ├── dynamic/
│   ├── dynamic_symmetric/
│   ├── mct_driver/
│   ├── nuopc_driver/
│   └── solo_driver/
├── docs/
├── pkg/
│   ├── CVMix-src/
│   ├── geoKdTree/
│   ├── GSW-Fortran/
│   └── MOM6_DA_hooks/
└── src/
    ├── ALE/
    ├── core/
    ├── diagnostics/
    ├── equation_of_state/
    ├── framework/
    ├── ice_shelf/
    ├── initialization/
    ├── ocean_data_assim/
    ├── parameterizations/
    │   ├── lateral/
    │   └── vertical/
    ├── tracer/
    └── user/
```

Driver layer

External packages

Model

`config_src/`

- Selectively compiled
  - NCAR coupled mode:
    `nuopc_driver + dynamic`
  - Stand-alone ocean model
    `solo_driver + dynamic`
- Alternative version of same code

  e.g.

  `dynamic` or `dynamic_symmetric`

Code tree: external packages

```
MOM6/
├── config_src/
│   ├── coupled_driver/
│   ├── dynamic/
│   ├── dynamic_symmetric/
│   ├── mct_driver/
│   ├── nuopc_driver/
│   └── solo_driver/
├── docs/
├── pkg/
│   ├── CVMix-src/
│   ├── geoKdTree/
│   ├── GSW-Fortran/
│   └── MOM6_DA_hooks/
└── src/
    ├── ALE/
    ├── core/
    ├── diagnostics/
    ├── equation_of_state/
    ├── framework/
    ├── ice_shelf/
    ├── initialization/
    ├── ocean_data_assim/
    ├── parameterizations/
    │   ├── lateral/
    │   └── vertical/
    ├── tracer/
    └── user/
```

Driver layer

External packages

Model

pkg/

- Not compiled in place
- Symbolic links to required source live under `src/` and point to `pkg/`
  - External packages often contain more than source and not all source compiles!
- Each package is a `git` sub-module
  - using specific commit hash

Code tree: main model

```
MOM6/
├── config_src/
│   ├── coupled_driver/
│   ├── dynamic/
│   ├── dynamic_symmetric/
│   ├── mct_driver/
│   ├── nuopc_driver/
│   └── solo_driver/
├── docs/
├── pkg/
│   ├── CVMix-src/
│   ├── geoKdTree/
│   ├── GSW-Fortran/
│   └── MOM6_DA_hooks/
└── src/
    ├── ALE/
    ├── core/
    ├── diagnostics/
    ├── equation_of_state/
    ├── framework/
    ├── ice_shelf/
    ├── initialization/
    ├── ocean_data_assim/
    ├── parameterizations/
    │   ├── lateral/
    │   └── vertical/
    ├── tracer/
    └── user/
```

Driver layer

External packages

Model

`src/`

- Code for solving equations of motion, tracers, diagnostics, etc.
- Always compiled
- No CPP macros except for MEMORY and GRID macros
  - the few existing exceptions will be removed one day

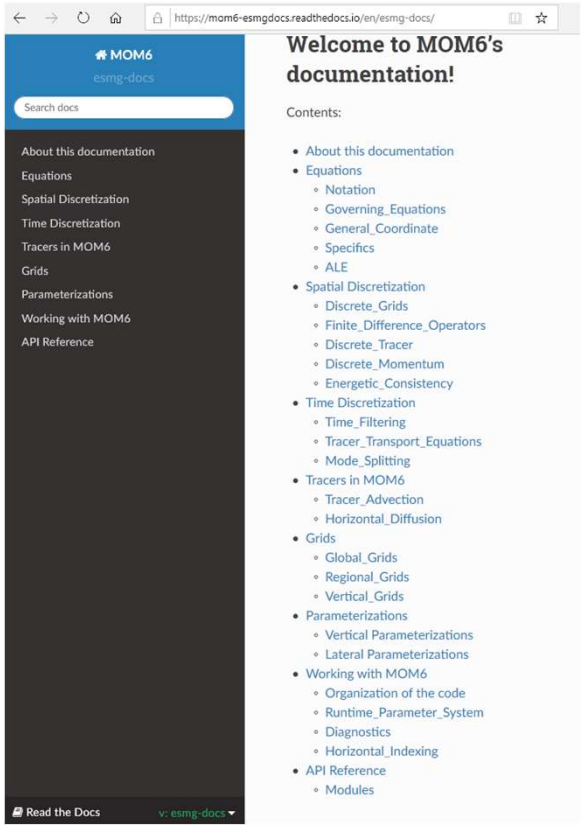
Code tree: main model

```
MOM6/
└── src/
    ├── ALE/
    ├── core/
    ├── diagnostics/
    ├── equation_of_state/
    ├── framework/
    ├── initialization/
    ├── parameterizations/
    │   ├── lateral/
    │   └── vertical/
    ├── tracer/
    └── user/
```

- Vertical remapping
- Dynamic core (momentum, continuity)
- Some collective diagnostic
- Five equations of state
- Interface to FMS (communications, I/O)
- Grid/state allocation/initialization
- Lateral parameterizations
- Vertical parameterizations
- Tracers including parameterizations (Redi)
- Configuration specific code (forcing/init)

Code tree: the most important bits

```
MOM6/
├── .testing/
│
│
│
│
└── docs/
```



`.testing/`

- Continuous integration
  - Runs tests on Travis-CI
    (soon also GitHub Actions)
  - Can be used for development

`docs/`

- Source for documentation hosted at https://mom6.readthedocs.io
  - Under dev. by K. Hedstrom

Future topics

- Verification and validation of MOM6 contributions
  - Marshall Ward
- Equations and algorithms
  - Bob Hallberg
- Lagrangian remap method
  - Stephen Griffies
- Analysis and tools
  - Raphael Dussin
- Ocean data assimilation interfaces
  - Matthew Harrison