Installation and Compilation of ROMS on Mac OS 10.15.7 Catalina (Darwin operating system), based on user osean on 28 January 2021:

(1) Install Xcode Command Line Tools:

Launch the Terminal, type the following

>> xcode-select --install

Click ‘Install’ when a software window pops up, asking whether to install the command line tools. This way circumvents downloading the entire Xcode which is cumbersome and somewhat unnecessary.

(2) Install HomeBrew: https://brew.sh

>> mkdir homebrew && curl -L https://github.com/Homebrew/brew/tarball/master | tar xz --strip 1 -C homebrew

Through HomeBrew, download the necessary unix commands, including and not limited to the following by searching up

>> gcc (including gfortran)

>> subversion (svn)

>> netcdf

>> gmake (gnu make)

>> perl

>> hdf5

>> mpich (mpi)

Check carefully with the following command in Terminal whether all necessary commands are installed:

>> which xxx (where xxx is the command of interest)

(3) Download ROMS

Create a directory ~/ROMS and obtain the source code through svn:

>> cd ~(enter master directory, e.g. ~/Documents)

>> mkdir ROMS

>> cd ROMS

>> svn checkout --username xxxxx (enter) https://www.myroms.org/svn/src/trunk

The folder will then have a ‘trunk’ directory with all the source codes.

(4) Customising the Build Script (build\_roms.sh)

The ROMS source code comes with a build script in the ROMS/Bin directory. Examples written with bash (build\_roms.sh) and csh (build\_roms.csh) are provided. For convenience, we will work with build\_roms.sh since bash is the default shell on a Mac. A full description of the build script can be found here: https://www.myroms.org/wiki/build\_Script

At the same level as your new trunk directory, create a new folder named Projects and change into it.

>> mkdir Projects

>> cd Projects

Create a folder named Upwelling and change into it. 'Upwelling' is the name of the ROMS test case we are going to compile and run.

>> mkdir Upwelling

>> cd Upwelling

Copy the roms\_upwelling.in file into the Upwelling directory you just created.

>> cp ../../trunk/ROMS/External/roms\_upwelling.in .

Copy the upwelling.h file into the Upwelling directory in the same way.

>> cp ../../trunk/ROMS/Include/upwelling.h .

Copy the build.bash file into the Upwelling directory.

>> cp ../../trunk/ROMS/Bin/build.bash .

Open the build\_roms.sh script you just copied into your Upwelling directory using text editor.

Scroll down until you find MY\_ROOT\_DIR and MY\_ROOT\_DIR:

# Set a local environmental variable to define the path to the directories

# where all this project's files are kept.

export MY\_ROOT\_DIR=/Users/xxxxx/Documents/ROMS (🡨 enter the ROMS directory)

export MY\_PROJECT\_DIR=${MY\_ROOT\_DIR}/Projects/Upwelling

Scroll down and make sure MY\_CPP\_FLAGS is disabled, i.e. commented out:

#export MY\_CPP\_FLAGS="${MY\_CPP\_FLAGS} -D"

Scroll down to compiler section, and disable all the MPI options, switch on gfortran, and as well as USE\_LARGE, USE\_NETCDF4, and USE\_HDF5, like below:

export USE\_MPI= # distributed-memory parallelism

export USE\_MPIF90= # compile with mpif90 script

#export which\_MPI=mpich # compile with MPICH library

#export which\_MPI=mpich2 # compile with MPICH2 library

#export which\_MPI=mvapich2 # compile with MVAPICH2 library

export which\_MPI=openmpi # compile with OpenMPI library

#export USE\_OpenMP=on # shared-memory parallelism

#export FORT=ifort

export FORT=gfortran

#export FORT=pgi

#export USE\_DEBUG=on # use Fortran debugging flags

export USE\_LARGE=on # activate 64-bit compilation

export USE\_NETCDF4=on # compile with NetCDF-4 library

export USE\_HDF5=on # compile with HDF5 library

#export USE\_PARALLEL\_IO=on # Parallel I/O with NetCDF-4/HDF5

Scroll down to enable customised library paths:

#export USE\_MY\_LIBS=no # use system default library paths

export USE\_MY\_LIBS=yes # use my customized library paths

MY\_PATHS=${COMPILERS}/my\_build\_paths.sh

if [ "${USE\_MY\_LIBS}" = "yes" ]; then

source ${MY\_PATHS} ${MY\_PATHS}

fi

(5) Edit my\_build\_paths.sh file:

A few things need to be edited in this file to direct the build file to the correct paths.

MPI root set to (if use MPI) the following, but since MPI is not used here it can be ignored:

export MPI\_ROOT=/usr/local

Down to the compiler section, for MPI\_SOFT, since the directory is referred to many times in the following lines, change it to

export MPI\_SOFT=/usr/local

Go down to the section starting gfortran, edit to get the following

gfortran )

export ESMF\_COMPILER=gfortran

if [ -n "${USE\_DEBUG:+1}" ]; then

export ESMF\_BOPT=g

else

export ESMF\_BOPT=O

fi

export ESMF\_ABI=64

export ESMF\_COMM=${which\_MPI}

export ESMF\_SITE=default

~~export ARPACK\_LIBDIR=/opt/gfortransoft/serial/ARPACK~~

export ARPACK\_LIBDIR=/Users/xxxxx(enter Mac user name) /Documents/ROMS/trunk/Lib/ARPACK (basically, change to the trunk file directory)

if [ -n "${USE\_MPI:+1}" ]; then

if [ "${which\_MPI}" = "mpich2" ]; then

#export MPI\_SOFT=/opt/gfortransoft/mpich2

export MPI\_SOFT=/usr/local

elif [ "${which\_MPI}" = "openmpi" ]; then

#export MPI\_SOFT=/opt/gfortransoft/openmpi

export MPI\_SOFT=/usr/local

elif [ "${which\_MPI}" = "mvapich2" ]; then

#export MPI\_SOFT=/opt/gfortransoft/mvapich2

export MPI\_SOFT=/usr/local

fi

~~export MCT\_INCDIR=${MPI\_SOFT}/mct/include~~

~~export MCT\_LIBDIR=${MPI\_SOFT}/mct/lib~~

~~export PARPACK\_LIBDIR=${MPI\_SOFT}/PARPACK~~

export MCT\_INCDIR=${MPI\_SOFT}/include

export MCT\_LIBDIR=${MPI\_SOFT}/lib

export PARPACK\_LIBDIR=/Users/xxxxx/Documents/ROMS/trunk/Lib/ARPACK/PARPACK (change to PARPACK folder under trunk directory)

fi

if [ ! -n "${SINGULARITY\_COMMAND:+1}" ]; then

if [ -n "${USE\_NETCDF4:+1}" ]; then

if [ -n "${USE\_PARALLEL\_IO:+1}" ] && [ -n "${USE\_MPI:+1}" ]; then

export ESMF\_DIR=${MPI\_SOFT}/esmf\_nc4

~~export NETCDF=${MPI\_SOFT}/netcdf4~~

export NETCDF=/usr/local/Cellar/netcdf/4.7.4\_2 (NETCDF directory, check by entering in Terminal >> nf-config --flibs)

export NF\_CONFIG=/usr/local/bin/nf-config (find out path by >> which nf-config)

export NETCDF\_INCDIR=${NETCDF}/include

export NETCDF4=1

else

export ESMF\_DIR=${MPI\_SOFT}/esmf\_nc4

~~export NETCDF=/opt/gfortransoft/serial/netcdf4~~

export NETCDF=/usr/local/Cellar/netcdf/4.7.4\_2

export NF\_CONFIG=/usr/local/bin/nf-config

export NETCDF\_INCDIR=${NETCDF}/include

export NETCDF4=1

fi

else

export ESMF\_DIR=${MPI\_SOFT}/esmf\_nc3

~~export NETCDF=/opt/gfortransoft/serial/netcdf3~~

export NETCDF=/usr/local/Cellar/netcdf/4.7.4\_2

export NETCDF\_INCDIR=${NETCDF}/include

export NETCDF\_LIBDIR=${NETCDF}/lib

export NETCDF\_classic=1

fi

fi

if [ -n "${USE\_PNETCDF:+1}" ]; then

~~export PNETCDF=${MPI\_SOFT}/pnetcdf~~

export PNETCDF=${MPI\_SOFT}

export PNETCDF\_LIBDIR=${PNETCDF}/lib

export PNETCDF\_INCDIR=${PNETCDF}/include

fi

if [ -n "${USE\_PIO:+1}" ]; then

~~export PIO=${MPI\_SOFT}/scorpio~~

export PIO=${MPI\_SOFT}

export PIO\_LIBDIR=${PIO}/lib

export PIO\_INCDIR=${PIO}/include

fi

if [ -n "${USE\_HDF5:+1}" ]; then

if [ -n "${USE\_PARALLEL\_IO:+1}" ] && [ -n "${USE\_MPI:+1}" ]; then

~~export HDF5=${MPI\_SOFT}/hdf5~~

~~export HDF5=${MPI\_SOFT}~~

export HDF5=/usr/local/Cellar/hdf5/1.12.0\_1 (this should be the directory with version specified, if installed through HomeBrew)

export HDF5\_LIBDIR=${HDF5}/lib

export HDF5\_INCDIR=${HDF5}/include

else

~~export HDF5=/opt/gfortransoft/serial/hdf5~~

~~export HDF5=${MPI\_SOFT}~~

export HDF5=/usr/local/Cellar/hdf5/1.12.0\_1

export HDF5\_LIBDIR=${HDF5}/lib

export HDF5\_INCDIR=${HDF5}/include

fi

fi

;;

(6) Create the Darwin-gfortran.mk file by simply copying Linux-gfortran.mk from the Compiler folder into the project folder:

cd ~/ROMS/trunk/Compilers

cp Linux-gfortran.mk Darwin-gfortran.mk

cd ~/ROMS/Projects/Upwelling

(7) Compile ROMS

Go into the ROMS directory:

>> cd ~/.../ROMS/Projects/Upwelling

and then

>> ./build\_roms.sh

If lots of stuff comes on the screen then you are good. This will take ~5 min.

If no error, when it finishes type >>ls

If you can see romsS (your executable file), then the compilation was successful.

However, it is likely, if netcdf was installed from HomeBrew, there would be an error like this:

“Undefined symbols for architecture x86\_64:

……

ld: symbol(s) not found for architecture x86\_64

collect2: error: ld returned 1 exit status

make: \*\*\* [/Users/XXXXX/Documents/ROMS/Projects/Upwelling/romsS] Error 1”

In this case, it is due to the HomeBrew netcdf not properly linked to source library. To fix this, go to /usr/local/bin, edit nf-config file (through text editor), and find the following line:

flibs="-L${libdir} "

Comment the line and add the following, so that the lines look like this:

#flibs="-L${libdir} "

flibs="-L/usr/local/lib -lnetcdff -lnetcdf"

Now, save the nf-config file. Go back to Terminal and re-compile,

>> ./build\_roms.sh

If no error and you can see romsS (your executable file), then the compilation was successful. Good luck!