KEI - The Tutorial

Revision History:

2014-09-29 Created Benjamin Saenz

2019-05-09 Updated w forcing file instructions, etc. Benjamin Saenz

Table of Contents

[Introduction 2](#_Toc8287567)

[KEI run files 2](#_Toc8287568)

[Building the executable 2](#_Toc8287569)

[Run Options File 3](#_Toc8287570)

[Forcing File 4](#_Toc8287571)

[Forcing File Creation Tutorial 5](#_Toc8287572)

[Running the model 6](#_Toc8287573)

[Postprocessing and viewing model output 7](#_Toc8287574)

[kei\_hacks.f90 7](#_Toc8287575)

[KEI Palmer LTER mega commands 7](#_Toc8287576)

[Some thoughts on use of KEI 8](#_Toc8287577)

# Introduction

The KPP-Ecosystem-Ice, or KEI, model consists of vertical water column model, using the KPP boundary layer mixing scheme, fully coupled to a thermodynamic sea ice model. KEI is forced at the atmospheric boundary by time-varying atmospheric data through various standard exchange energy and mass and flux parameterizations. Various unphysical assimilation routines and ‘hacks’ are available for developing tests for particular problems.

KEI is written in FORTRAN 95. Many helper routines for creating, editing, and post-processing data are available for MATLAB, however MATLAB is not required for run the model.

This model derives from:

* Large et al. [1994], Doney et al. [1996](KPP mixing)
* Ukita and Martinson [2001](Mixed layer - ice interactions)
* Saenz and Arrigo [2012 and 2014] (SIESTA sea ice model)
* Hunke and Lipscomb [2008] (CICE v4 ice model – various components)
* Moore et al. [2002,2004] (Biogeochemical Ecosystem Model)

# KEI run files

Three things are required for KEI to run:

1) Executable (from FORTRAN compiler)

2) Run Options File

3) Forcing Data File (NetCDF)

## Building the executable

There are five necessary components to build KEI: A FORTRAN 95 compiler, LAPACK, NetCDF, Make, and the KEI source code.

FORTRAN:

For Macintosh: http://hpc.sf.net (use GCC 4.8; 4.9 was broken last I checked and 5.0 is untried)

For Linux: gfortran, or intel ifort (both free for non-commercial use)

CPP or an equivalent preprocessor is also required, but is installed with GCC and/or ifort.

\*\*\* Note, the CPP that is included in with GCC/gfortran v. 7 or higher is problematic, because it inserts some C-style comments into the top of every file it processes. This happens by default in Linux, but somehow in the GCC binaries available at hpc.sf.net for Mac OS do not exhibit this behavior. To avoid having to manually remove the comments from each file that CPP processes, there are a few options:

* Explore other preprocessor packages you can install (fpp is one I think, there are a few around, maybe they need to be compiled)
* Figure out a gfortran preprocessor flagging (haven’t tried)
* Figure out CPP switches to modify this behavior (tried for a while but failed).
* Modify the Makefile to not use CPP/avoid changes to the files that require CPP

LAPACK:

On Macintosh, LAPACK is available as a part of Developer Tools, but you must build the interface. Linking is available in the Makefile. On Linux, ifort ships with LAPACK; heinous linking is already done in the Makefile for ifort 11.1, but for different versions it can be tough to figure out. For gfortran, you must download LAPACK, build it and link it.

NetCDF:

On Macintosh, it must be downloaded and built with FORTRAN 90 interfaces. On linux, your distribution likely has a package you can install and link. This can be a pain.

KEI source code:

The FORTRAN 95 source code consists of ~35 files. The file that contains the Main driver routine is KEI.f90. The Makefile used by Make to build the executable should be distributed with the source code.

Make/Makefile:

If developer tools are installed, you should have the make command available. Edit the file "makefile" to suit your system. Setting the FC variable to either ifort for gfortran will give you independent control over those two compilers. You will have to edit the paths to the NetCDF include and lib directories, and also amend the linking to some version of LAPACK. (Note that ifort and MacOS X ship with LAPACK - you must build the LAPACK FORTRAN interfaces before using however). Windows is not supported, mostly because it is incredibly difficult to get the FORTRAN NetCDF interface installed on windows.

Type "make clean", then "make." The executable KEI.run should be produced. Various warnings are produced by different compilers; they seem to be benign on my test systems thus far.

## Run Options File

Currently much of this file is legacy garbage, but the model is expecting some of it until I clean it up. We are interested in only lines:

Line 13: start time (in days) - corresponds to the "time" forcing variable; the number of days after the start of forcing data

Line 14: time step start (always 0), end (# desired steps), and time step length (seconds - 1hr (3600s) time step is the only one tested!).

Line 21: The PATH where output data will be written.

Line 24: Toggles for switching on/off parts of the model, such as ice, ecosystem, KPP, etc. I have only tested toggling the ecosystem and ice.

Line 50: Path/filename to where the forcing NetCDF file is located

Example run options file: run.00.so

## Forcing File

The forcing file expected by KEI is a NetCDF file, with specific variables describing the grid, physical and ecosystem initializations, and atmospheric and ocean time series forcing data. The data file can be generated using MATLAB from a specific MATLAB data structure that is relatively easy to manipulate. Furthermore, the forcing and initialization data can be (mostly) generated from raw data, in the form of ECMWF ERA Interim Climatology and various NCEP BCG model outputs.

Example NetCDF forcing file: kf\_200\_100\_2000.nc

A corresponding MATLAB structure containing the same data: kf\_200\_100\_2000.mat

The MATLAB structure can be written directly to netcdf format needed by KEI with this command:

kei\_write\_forcing(kf\_200\_100\_2000,’kf\_200\_100\_2000.nc’)

Inside this MATLAB data structure are 4 fields ...

1. forcing
   1. Contains atmospheric, sea ice, and potentially ocean forcing time series data
   2. Data is hourly
   3. Date field is in decimal days starting with the first day at 0
   4. ic, ain, aout, and divu are sea ice variables; ic is required for ice-conforming runs, and the other fields are needed if dynamic ice growth/ridging through convergence is wanted (this is usually off); can be zeros otherwise.
   5. wct is an optional field with hourly temperature profiles. If this variable is present, it can be used to perform simple temperature/salinity assimilation.
2. grid
   1. Contains three variables use by the ocean model to describe vertical layer boundaries, midpoints, and thicknesses
3. init
   1. Contains initial profiles of temperature, salinity (and u and v, typically set to zero; lateral inputs are still in the KPP code but have not been tested).
4. eco\_init
   1. Contains initial profiles of all BGC variables

## Forcing File Creation Tutorial

There are several functions that can help create forcing data. In the following tutorial we will create a new forcing file for the Palmer LTER station 600.040, for the calendar years of 2002 through 2003.

First, get the latitude/longitude (here we find from LTER grid points)

[lat, lon] = grid2llLter(200,100); % grid2llLter comes from LTER toobox

Generate the vertical grid. 100 layers, over 100 m. Keeping the first 2 numbers the same in this command will generate a 1m-resolution grid, which is a dependency of many of the postprocessing tools, and is the only grid resolution that has been tested. The grids will have 1 extra later ay the bottom (in this case 101 total layers), with the bottom bin having a very tiny thickness, and in which tracer values are constant. \*\*\* Remember if the grid size is changed, it must also be changed in the FROTRAN code in kei\_parameters.f90. In this case you would change the NZ variable to 100.

[grid.dm grid.hm grid.zm] = kei\_generate\_grid(100,100,0,1);

Retrieve and interpolate ECMWF atmospheric forcing data. This function depends on having downloaded yearly ECMWF-interim netcdf files with appropriate fields (t2m,d2m,tau\_x,tau\_y,prain,psnow,…). Note: don’t use date ranges ending on day 1 (i.e. 2001,1,2004,1). The interpolation scheme fails with only one day to interpolate across. Would be good to fix that at some point.

forcing=kei\_fdat\_prepare(lat,lon,2001,1,2003,365,'/data/ECMWF\_int/',0)

Find some valid inital BGC values. Here we pull and interpolates profiles from CESM output given a lat, lon, and month(1-12). (The file below actually only has January)

eco\_init = kei\_prepare\_eco\_init('ecosys\_jan\_IC\_gx1v6\_20100514.nc4',lat,lon, grid.dm,1);

Get and interpolate ice concentration observations. There are several different daily sources from the NSIDC from passive microwave sensing. There is currently no exact tool for extracting this data. ASCII text files are available from Palmer LTER for the LTER grid locations for many years.

<copy/paste 2 years of daily ice concentration into MATLAB variable ‘ic\_600\_040’>

day = 1:730 - 1;

hour = day\*24;

houri = 0:hour(end);

ic\_interp = interp1(hour,ic\_600\_040,houri,'linear');

forcing.ic = ic\_interp(1:17496);

For now, zero ain, aout, and div forcing variables since they are not being used.

forcing.ain(1:17496)=0;

forcing.aout(1:17496)=0;

forcing.div(1:17496)=0;

Also zero the ‘dustf’ forcing variable. There is a function for extracting dustf from CESM GCM runs (see kei\_prepare\_dust\_flux.m), however we found the dust-generated Fe values much to high in the Antarctic Peninsula region, in general.

forcing.dustf(1:17496)=0;

Get initial T and S profiles. Here we are using Palmer LTER CTD casts that have been interpolated to 1dB pressure, assumed to correspond approximately to 1m depth resolution of the grid. We set initial u & v to 0 m/s for the whole water column.

load castStructLTERall.mat % loads castAll matlab structure

cast\_num = find\_LTER\_cast(castAll,2003,600,040)

init.t = castAll(cast\_num).te1db(1:101);

init.s = castAll(cast\_num).sa1db(1:101); % 1db assumed to correspond to 1m depth

init.u(1:101)=0;

init.v(1:101)=0;

Create the final matlab structure, using any name for the structure, but with these specifically-named four sub-structures.

kf\_600\_040\_2002.grid = grid

kf\_600\_040\_2002.forcing = forcing

kf\_600\_040\_2002.init = init

kf\_600\_040\_2002.eco\_init = eco\_init

Save the matlab structure and corresponding NetCDF file, then update the run options text file and run!

save kf\_600\_040\_2002 kf\_600\_040\_2002

kei\_write\_forcing(kf\_600\_040\_2002,’kf\_600\_040\_2002.nc’);

# Running the model

Running the KEI.run executable is simple:

KEI.run < options\_file

Where options\_file is the file created for the run, and which points to the data directory where the forcing file should be and the output NetCDF file will be written. Run times will vary based upon the processor speed, disk speed, compiler and optimization. KEI writes out a tremendous amount of output data, both to screen and to disk (~750 Mb/yr). Run times can be decreased by using an SSD drive for writing and reading, and by suppressing output printing by piping screen output to NULL like this:

KEI.run < options\_file > /dev/null

# Postprocessing and viewing model output

The majority of output NetCDF variables can be read in MATLAB for analysis using:

k=kei\_read(‘output.nc’)

This can be extremely slow, especially with all BCG tracers. For most current plot commands, kei\_read\_plotting() works faster by not reading BCG data. The are other reading functions as well that read different parts of the results.

Example data loading and plotting:

k=kei\_read\_plotting('john.nc')

load kf\_200\_100\_2000; kf = kf\_200\_100\_2000

kei\_plot\_fluxes(k,200,[1,length(k.time)],kf,19.625\*24,2000)

kei\_plot\_ecoline(k,200,[1,length(k.time)],kf\_200\_100\_2000,19.625\*24)

kei\_plot\_ecoiceline(k,200,[1,length(k.time)],kf,19.625\*24,2000)

kei\_compare\_ctd\_plus\_ice(k,castAll,2000,200,100,kf,200)

For quick and dirty views of physics and BGC:

k=kei\_read(‘output.nc’)

kei\_plot\_ice\_d(k,200)

kei\_plot\_eco\_d(k,200)

# kei\_hacks.f90

This file contains the toggles necessary to examine certain physical and BCG processes. These are hard-coded changes, meaning the code must be re-compiled for them to take effect.

These ‘hacks’ and switches remain to be fully documented, but here is what a few of them do:

ic\_conform = 1 Enable SSM/I sea ice concentration conforming

shortwave\_multiplier = 0.8 Fractional multiplier for shortwave irradiance (0.8 for ECMWF is standard)

fe\_multiplier = 1.0 Multiply initial Fe profile

fe\_offset = 200. Shift downward Fe profile(m) [negative to use forcing] bio\_offset = 300. Shift living things downward (m) [0.0 to use forcing]

ice\_diatChl = 25.6/0.6 Melting ice contains diatoms (mmol/m^3)

ice\_fe = 6. Melting ice releases Fe (nm)

# KEI Palmer LTER mega commands

These commands are examples of how to automate running multiple different forcing configurations and perform processing. This function takes a vector of different years, a name to label the runs with, and a Boolean that allows overwriting old results. It is useful to build something like this to automate work. Things it does, looping over each year:

* Looks up the options file, forcing file, and start day-of-year for a particular year in a lookup table. The start day-of-year is also in the options files but it seemed easier to put it in the script rather than have MATLAB parse the text file.
* On MacOS, changes some shell environment variables that are incompatible with gfortran.
* Performs the run using the system() command. Note you likely have to replace libgfortran.dylib in you MATLAB install – see the code for what to do.
* On MacOS, changes those shell environment variables back.
* Makes a copy of the code (kei\_moorings only) and reads and copies the NetCDF output file out of the DATA directory and renames it.
* Reads the forcing file, and then make plots and saves plots and calculates summary tables.

kei([1997:2010],'run\_name',1);

kei\_moorings(2007:2011,run\_name,1,[0,1,2]);

See README\_KEI\_Palmer\_LTER.txt for more information on the LTER specific stuff in these commands.

# Some thoughts on use of KEI

1. Winds – we performed an informal sensitivity analysis to high-frequency wind speed using sub-hourly data from an AWS station from and island outside of Marguerite Bay, an compared it to using climatological (3- or 6-hourly winds). In each case the winds were averaged (in the case of the AWS) or interpolated (ECMWF climatology) to a 1-hr timestep. KPP seems to over-estimate mixing with the high-frequency (more gusty) wind data. This sort of makes sense, since the KPP missing scheme has been refined to use GCM data? Something to think about when considering wind forcing.
2. Shortwave irradiance: Extensive testing found the using 80% of climatological shortwave energy produced proper ice melt and mixed layer warming in the west Antarctic Peninsula region. We didn’t dig into why, but it could due to problems with polar clouds in climatologies, but maybe due to simplified specular reflection in the ocean part of KEI, or something else too.