U. S. Department of Commerce

National Oceanic and Atmospheric Administration

National Weather Service

National Centers for Environmental Prediction

5830 University Research Court

College Park, MD 20740

Technical Note

Keeping Ice'S Simplicity – A Modeling Start[†].

Robert W. Grumbine [‡]

Environmental Modeling Center

Marine Modeling and Analysis Branch

November 19, 2013

[†] MMAB Contribution No. 314.

[‡] e-mail: Robert.Grumbine@NOAA.gov

1. Abstract

Sea ice is, in some respects, a simple component of the weather and climate system. In this note, I outline a starting point for sea ice modeling that keeps that simplicity in the sea ice model proper. As future usages will demand less simple sea ice representations, the framework for the model is not at all simple. For additional future uses, it is important to develop Earth System Modeling Framework software connections in this model. Further, the version 0 model establishes a baseline for sea ice model skill. No new model can be accepted if it is not demonstrably better than version 0 by skill measures.

2. Introduction

Sea ice is quite difficult to model in some respects, involving complex dynamics [e.g. *Hibler* (1979)]. Yet, it shows a number of simplicities. For instance, one can accurately forecast sea ice cover between 40 S and 30 N for decades in advance. Granted that it is zero cover. One can also [*Thorndike and Colony* (1982), *Grumbine* (1998), *Grumbine* (2013)] predict sea ice's drift fairly well with even extremely simple models, as compared to results of quite complex models [e.g. *Preller and Posey* (1989)].

On the other hand, providing the forecast environment for a sea ice model or module is not as simple a matter of computer science. One issue is to obtain all needed inputs, perform whatever quality control and mutual reconciliation is needed, execute the model, and produce some usable outputs and displays.

Part of the module requirement is to be able to run the model both in a free-standing mode, and as a NEMS (NOAA Environmental Modeling System [*NCEP/EMC* (2013)]) module/component. One portion of that being ESMF (Earth System Modeling Framework [*Modeling* (2012)] compatibility.

The free-standing mode is required because a) the sea ice model/module/component is far less computationally demanding than any ocean model or atmospheric model it may be coupled with and b) flux biases which may be considered very small for an atmospheric model can be overwhelmingly important for the sea ice pack. 10 W/m^2 bias for a year represents about 1 m of ice melt. Since Arctic ice pack average thickness had dropped to only 1.8 m by 1999 [*Rothrock et al.* (1999)], representing a loss of over 2 meters in 40 years, 2 meters in 2 years would be an enormous error. The observed melt over that span represents a bias of only about 0.5 W/m^2. We will, therefore, spend some considerable time examining how to correct flux biases from the atmosphere and ocean – requiring many runs of the model for years.

It is also an old observation from sea ice modeling, perhaps one in need of re-examination with more modern models, that most of the skill in sea ice prediction comes from the thermodynamics, rather than dynamics [*Hibler and Walsh* (1982)]. Thermodynamics-only having a skill of 0.48, the full dynamic-thermodynamic coupled improving this to only 0.58 on a scale of 0-1 for correlation in ice extent within 30 degree sectors across the northern hemisphere [*Hibler and Walsh* (1982)].

3. Model

For now, in the rough and ready execution, I'm using the GDAS + GFS for meteorological inputs, hindcast and forecast, respectively. PDY is the current day, PDYm1 is the day before.

Also for now, and for simplicity, the model runs on the full resolution GFS native grid, 1760x880 gaussian grid points (T574).

a. Stand-alone Run Structure

The structure of a run is:

i. Get the GDAS run from PDYm1 to perform the hindcast needed to bring the model to the present.

ii. Get the GFS run from PDY to make forecast.

iii. Get the sea ice analysis for PDY to conduct skill assessments.

iv. Execute the model system through the Environmental Equivalence (EE) *Tolman and Caruso-McGee* (2011) structure, constructing all model-specific inputs and producing and distributing (as for operations) output. The full output at this point is approximately 1.8 Gb for a 1 day hindcast 8 day forecast.

v. Back outside the EE, score forecasts from the previous 8 days against the current day's analysis, and copy files to the web and ftp server for review.

vi. Finally, remove the running directory. The running space is approximately 12 Gb for 1 day hindcast with hourly fields and 8 day forecast with 6-hourly fields.

The files needed for the 1 day hindcast (hourly from the GDAS) occupy about 4.5 Gb (grib1, full model resolution). The (potentially) 8 day forecast's files take approximately 6 Gb (6 hourly through 192 hours). This is not needed if running on the NWS operational system as the files and directories will already be present.

At this point, no real ocean inputs exist. They are created by an auxiliary program.

The execution, on my desk, on a single 3.3 GHz intel processor, requires approximately 25 minutes – almost all of it in retrieving and reading the 10 Gb of meteorological inputs. Running set size is approximately 430 Mb.

b. Inputs

With an eye towards being a NEMS component, no assumptions are made about the grid that the sea ice model is on. To keep that option open for when the model advances to a point of advecting fields, the model reads in the latitudes and longitudes of the grid points, and it reads in a land mask.

Oceanic inputs are currently held fixed through a run, and are constant in space. They are mixed layer temperature, salinity, depth, and velocity. Also the significant wave height and period. The current version of the model does not use these fields. But a future version will. Each time step for this file requires approximately 50 Mb.

Meteorological inputs are: u momentum flux to the surface, v momentum flux to the surface, ice concentration, ice thickness, precipitation rate, convective precipitation rate, T(2 meters), u(10 meters), v(10 meters), specific humidity (2 meters), surface pressure, ground heat flux, sensible heat flux, latent heat flux, downwelling longwave radiation flux, and downwelling shortwave radiation. The GFS computes its own fluxes, assuming the ice concentration and thickness that it evolves on its own *NCEP/EMC/GCWMB* (2005). As that model may be incompatible with this one and its descendants, it is necessary to be able to compute the fluxes of heat, moisture, and momentum inside this model. The external models' values for those fluxes, as well as ice concentration and thickness, is retained for quality purposes of comparing the external and internal computations of those values. This file is approximately 100 Mb per step.

As a NEMS component in RTOFS-Global *Mehra and Rivin* (2011) all these fields will be available, obviating the need for reading files.

c. Sea Ice Model/Module Restart fields

The model itself is doing nothing at this point, so could be given no restart file. But any real ice model will need some fields, so to keep the emulation of a real model going fields saved are: ice concentration, ice thickness, ice temperature, snow temperature, snow thickness, ice velocity (U, V), temperature and salinity of the mixed layer. For now these are all single fields – only one temperature (a skin temperature) for the sea ice, for instance. At a later date this will probably become 3-4. Mean ice thickness will continue to exist, but will become a diagnostic field as ice thickness distribution (fraction of cell area that is covered by ice of thickness between h_i and h_{i+1}, for some number of intervals) is introduced.

For diagnostic purposes and the later development of bias correction, the output during a run includes the restart fields plus the temperature and salinity of the mixed layer. These fields are output in the forecast run at the same frequency as the input weather fields are updated, 6-hourly.

4. Model Structure

The main execution loop

i. compute fluxes of heat, water, salt, and momentum between air-ice, air-ocean, ice-ocean. (12 fields)

ii. evalute thermodynamics of the ocean mixed layer (i.e. heat or cool it from above, as indicated by the fluxes)

iii. evaluate thermodynamics of the sea ice (i.e. perform freezing/melting and, when multiple categories, move ice appropriately between thickness categories). This also includes adding precipitation to the ice pack (snow thickness evolution), and the flooding of sea ice if/when the snow becomes thick enough to depress the ice surface below the ocean surface.

iv. evaluate dynamics of the sea ice (determine velocity field, including that, for example, convergent areas can resist the motion by ridging).

v. evolve the concentration field

This structure is applicable at this point, not because it actually does anything, but because it provides the names, locations, and structures that will be needed to support a real model.

5. Skill

Verification of sea ice drift is relatively well-known [e.g. *Grumbine* (1998), *Grumbine* (2013)]. But sea ice concentration poses some special issues because it is not global (as opposed to temperature) but is largely continuous in space where it does exist (as opposed to rainfall) and is highly consistent from year to year (making climatology a extremely good 'forecast'). Some work was done earlier [*Waldrop* (1997)] looking at sea ice concentration model forecast evaluation. But here, we will begin much more simply.

The scores at this level are simply comparison of the forecast sea ice field to the observed field, over the entire globe. This is done on the observation's grid, the high resolution sea ice analysis [*Grumbine* (2013, in preparation)], 5 arc minutes. The measures, shown in table 1 for KISS V0 forecasts valid on 2012/11/21, are mean concentration error, rms concentration error (concentrations in both cases being fractions [0-1]), a11, a12, a21, a22. The last four are the components of the verification contingency table [*Murphy* (1995)], a11 being that the model forecast there to be ice, and ice was observed. a12 is the model forecast ice, but it was not observed. a21 the model forecast there to be no ice, but some was observed. And a22 is that the model and observation agreed that there was no ice. All the a_{ij} are measured as fractions of the verification area.

Initially, the entire globe is used as the verification area. This leads to the extraordinarily high scores. % correct for 5 day lead was 98.8% (a11+a22). The probability of detection, PoD, a11/(a11+a21), was 85.3%. While the false alarm rate, FaR, a12/(a12+a11), was only 6.2%.

The first correction to this is to score only those areas that are water which at some point in the 30 year climatology period (1981-2010) was observed to be colder than 275.3 K (the filter temperature in *Grumbine* (2013, in preparation) ice analysis, and in the *Grumbine* (2009) analysis). This reduces the scoring area from 510 million km^2 to about 69 million km^2. Scores, nevertheless, remain extraordinarily high, see figures 1-5 for the bias, rms error, probability of detection, false alarm rate, and % correct scores for the models verifying on 26

March 2013. One can elaborate this somewhat farther, as the area of ocean which has seen an ice cover at any time in the satellite era is about half the 69 million km^2. But, as eliminating about 85% of the area scarcely budged the scores, intuition suggests an additional 50% will not bring about drastic changes. This points to the merit of examining sea ice concentration skill measures in their own right.

6. Ancillary/Additional Models

Implicit in weather models that cover sea ice covered regions, and likewise for ocean models that do so, is a sea ice model. It may be well-hidden, as for the NMMB *NCEP/EMC/MMB* (2011), but it is present. NCEP currently has several such sea ice models. They are also displayed in the ice model web page, `http://polar.ncep.noaa.gov/develop/icemodel/`, and their skill is assessed as for the KISS v0 model.

a. NMMB

The North American Mesoscale Model - B grid version includes ice cover through the NOAH [*Ek et al.* (2003)] land surface model. In this, all ice is infinitely thick (as land is), 100% compact, and does not move. Heat diffuses in to the ice and back out, and snow can land on it and accumulate/melt. Initial conditions (coverage) are taken from the IMS snow/ice analysis [*Chen et al.* (2012)].

The grid is limited domain, at about 4 km resolution.

b. GFS

The Global Forecast System ice cover is initialized with sea ice concentration from the half degree version of the NCEP sea ice concentration analysis [*Grumbine* (2013, in preparation)]. Sea ice has concentration initialized from that analysis, and then kept constant through the length of the run. Sea ice thickness is allowed to evolve, but not to the point of eliminating initial ice, nor to create ice in new areas. There is no operational initialization of sea ice thickness. Thicknesses range from 10 cm (the minimum [Wu, personal comm.]) to about 5 m. Snow can land on the sea ice, and may induce flooding. Energy is diffused through the sea ice and snow layers. Again, sea ice does not move.

The gridding is global, at T574 (approximately 30 km grid spacing).

c. RTOFS-Global

Energy loan thermodynamics [*Halliwell and Bleck* (2001)], no velocity. Initial conditions from Naval Oceanographic Office NCODA analysis [*Lunde and Coelho* (2009)].

The grid is tripolar, with variable resolution of about 3.5 km in the Arctic, and ca. 9 km in the Antarctic.

d. CFS

The Climate Forecast System, version 2 [*Saha et al.* (2011), *Wu and Grumbine* (2013)] sea ice is represented by a full dynamic-thermodynamic sea ice model. Dynamics are the Elastic-Viscous-Plastic [*Hunke and Dukowicz* (1997)] used in the CICE [*Hunke and Lipscomb* (2008)] model that is also a component of the NCAR CESM and Navy ACNFS [*NCAR* (2012), *NRL* (2013)]. Thermodynamics follow Winton, GFDL [*Winton* (2000)]. The model has a substantial bias towards overly extensive ice cover, particularly in forecasts for summertime [e.g. ARCUS].

The ocean grid is half-degree, tricolor north of 65 N, but current operational 6-hourly output is on a 1 degree grid.

7. ESMF-ification

Transition from a straightforward, and isolated, sea ice model to being an ESMF 'component' is required as a step towards being a NEMS-compatible module. For KISS v0, I proceed far down this path, but not all the way to an ideal NEMS module. That will await KISS v1.

The first step from free-standing model towards ESMF module is to segregate the sea ice aspects to their own subroutine(s), away from the meteorology and oceanography. This, so that in the future the meteorology and oceanography can replace the current file reading with models of their own. There is nothing ESMF-specific in this.

The second step, and lowest level of transition to ESMF structure is to begin to use ESMF structures for the model's control and execution. On the side of the ice model (module), and the meteorology and oceanography, this means that a module must be created that contains subroutines to perform initialization of the model, execution (run a step) of the model, and finalization (final outputs, closing files, etc.) for the sea ice component. Repeat for all components. There must also be in each component a routine which is public, and which owns the initialize/run/finalize routines. At this level, a minimalist ESMF overhead for the sea ice component looks something like:

```
MODULE icemodel
    USE ESMF

    IMPLICIT none !not mandatory but a good idea

!Declare model's variables

!---------------------------- START ESMF routines -------------------------------------
!   ! Public subroutine which the main program will call to register the
!   ! various user-supplied subroutines which make up this Component.

    SUBROUTINE Icemodel_SetServices(gcomp, rc)

      type(ESMF_GridComp) :: gcomp

      integer, intent(out) :: rc


       call ESMF_GridCompSetEntryPoint(gcomp, ESMF_METHOD_INITIALIZE, my_init, rc=rc)

       call ESMF_GridCompSetEntryPoint(gcomp, ESMF_METHOD_RUN, my_run, rc=rc)

       call ESMF_GridCompSetEntryPoint(gcomp, ESMF_METHOD_FINALIZE, my_final, rc=rc)


    END SUBROUTINE Icemodel_SetServices

!   ! User-written Initialization routine

    SUBROUTINE my_init(gcomp, importState, exportState, externalclock, rc)

      type(ESMF_GridComp) :: gcomp

      type(ESMF_State) :: importState

      type(ESMF_State) :: exportState

      type(ESMF_Clock) :: externalclock

      integer, intent(out) :: rc


      CALL invariants !user routine opening files and initializing data


      print *, "Icemodel initialize routine called"

      rc = ESMF_SUCCESS


    END SUBROUTINE my_init

!   ! User-written Run routine

    SUBROUTINE my_run(gcomp, importState, exportState, externalclock, rc)

      type(ESMF_GridComp) :: gcomp
      type(ESMF_State) :: importState
      type(ESMF_State) :: exportState
      type(ESMF_Clock) :: externalclock
      integer, intent(out) :: rc

      print *, "Icemodel run routine called"
      CALL icemodel_core                   !this is the actual ice model executor
      rc = ESMF_SUCCESS

    END SUBROUTINE my_run
!   ! User-written Finalization routine
    SUBROUTINE my_final(gcomp, importState, exportState, externalclock, rc)
      type(ESMF_GridComp) :: gcomp
      type(ESMF_State) :: importState
      type(ESMF_State) :: exportState
      type(ESMF_Clock) :: externalclock
      integer, intent(out) :: rc

      CALL restart_output !this and 'finish' are the only user parts shown
      CALL finish

      print *, "Icemodel finalize routine called"
      rc = ESMF_SUCCESS

    END SUBROUTINE my_final
!--------------------------- END ESMF routines ------------------------------------
```

```
!follow up with the sea ice model routines


END MODULE icemodel
```

This structure is repeated for the meteorology, oceanography, and any other things which will be made models/modules/components. Notice that at this point, the model contains no variables. Those are declared near the top, at the comment in this example. The variables

```
gcomp, importState, exportState, externalclock
```

are declared and passed at this level of development, but are not used. The first three, especially, are crucial to full NEMS usage, but their usage adds significantly to the obscurity of the code for those of us unfamiliar with ESMF in the first place.

The main program, in addition, must establish the ESMF control structures, enable and initialize the ESMF time management routines and variables, and pass the proper ESMF variables through each of the init/run/finalize steps of each component, and then perform a shutdown of the ESMF itself.

This is a sample skeleton, derived from the sample in the ESMF distribution:

```
      PROGRAM kiss
! Keep Ice'S Simplicity


! 24 January 2013 -- start adding in ESMF material:
      USE ESMF


      USE icemodel ! module for sea ice modelling
      USE meteo    ! module for meteorological modelling/file reading


      IMPLICIT none ! not required, but a good idea
```

```
! ESMF generic variables -- time management, return codes, memory:
      integer :: rc
      integer :: finalrc
      type(ESMF_Clock) :: tclock
      type(ESMF_Calendar) :: gregorianCalendar
      type(ESMF_TimeInterval) :: timeStep
      type(ESMF_Time) :: startTime, stopTime
      type(ESMF_VM) :: vm

! ESMF variables for managing components and initialize/run/finalize
      logical :: finished
      character(ESMF_MAXSTR) :: cname, cname1, cname2
! Local ESMF vars for managing the modules:
      type(ESMF_State) :: states(2)
      type(ESMF_GridComp) :: top
      type(ESMF_GridComp) :: gcomp1, gcomp2
      type(ESMF_CplComp) :: cpl

! --------------------------- Start ESMF elements ------------------------------
      PRINT *,'started the program '

      finalrc = ESMF_SUCCESS
!    ! Initialize the Framework, and get the default VM
      CALL ESMF_Initialize(vm=vm, defaultlogfilename="IceModelEx.Log", &
                logkindflag=ESMF_LOGKIND_MULTI, rc=rc)
      IF (rc .ne. ESMF_SUCCESS) THEN
```

```
        print *, "failed to initialize ESMF Framework"
        print *, "FAIL: ESMF_FieldCreateEx.F90"
        stop
      END IF
      PRINT *,'past the first bit of ESMF stuff, finalrc =',finalrc

! Start up time management pieces -- this is currently all dummy code wrt
!  running ice model.  See the TimeMgr document for the details on the
!  actual code needed to set up a clock.  Initialize calendar to be Gregorian type:
      gregorianCalendar = ESMF_CalendarCreate(ESMF_CALKIND_GREGORIAN, &
                name="Gregorian", rc=rc)
      IF (rc.NE.ESMF_SUCCESS) finalrc = ESMF_FAILURE

      ! initialize time interval to 6 hours
      CALL ESMF_TimeIntervalSet(timeStep, h=6, rc=rc)
      IF (rc.NE.ESMF_SUCCESS) finalrc = ESMF_FAILURE

      ! initialize start time to 5/1/2003
      CALL ESMF_TimeSet(startTime, yy=2003, mm=5, dd=1, &
                calendar=gregorianCalendar, rc=rc)
      IF (rc.NE.ESMF_SUCCESS) finalrc = ESMF_FAILURE

      ! initialize stop time to 5/2/2003
      CALL ESMF_TimeSet(stopTime, yy=2003, mm=5, dd=2, &
                calendar=gregorianCalendar, rc=rc)
      IF (rc.NE.ESMF_SUCCESS) finalrc = ESMF_FAILURE
      ! initialize the clock with the above values
      tclock = ESMF_ClockCreate(timeStep, startTime, stopTime=stopTime, &
                name="top clock", rc=rc)
      IF (rc.NE.ESMF_SUCCESS) finalrc = ESMF_FAILURE
      PRINT *,'past the time and clock bit of ESMF stuff finalrc=', finalrc

!Call the gridcompcreates
      cname1 = "Icemodel"
      gcomp1 = ESMF_GridCompCreate(name=cname1, rc=rc)
      IF (rc.NE.ESMF_SUCCESS) finalrc = ESMF_FAILURE
      PRINT *,'creating sea ice component, finalrc = ',finalrc
      ! This single user-supplied subroutine must be a public entry point
      !  and can renamed with the 'use localname => modulename' syntax if
      !  the name is not unique.
      ! (see below for what the SetServices routine will need to do.)
      CALL ESMF_GridCompSetServices(gcomp1, Icemodel_SetServices, rc=rc)
      IF (rc.NE.ESMF_SUCCESS) finalrc = ESMF_FAILURE
      PRINT *,'creating phys set services, finalrc = ',finalrc

      cname2 = "Meteorology"
      gcomp2 = ESMF_GridCompCreate(name=cname2, rc=rc)
      IF (rc.NE.ESMF_SUCCESS) finalrc = ESMF_FAILURE
      ! This single user-supplied subroutine must be a public entry point.
      CALL ESMF_GridCompSetServices(gcomp2, Meteo_Set_Services, rc=rc)
      print *, "Comp Create returned, name = ", trim(cname2)

! ESMF Create the necessary import and export states used to pass data
!  between components.
      states(1) = ESMF_StateCreate(name=cname1, &
                stateintent=ESMF_STATEINTENT_EXPORT, rc=rc)
      states(2) = ESMF_StateCreate(name=cname2, &
                stateintent=ESMF_STATEINTENT_EXPORT, rc=rc)

! --------------------------- END ESMF overhead elements ------------------------------
! Now done with initializing ESMF itself, do the initializations of the
!    module components

! Call each Init routine in turn.

      CALL ESMF_GridCompInitialize(gcomp1, exportState=states(1), &
                clock=tclock, rc=rc)
      CALL ESMF_GridCompInitialize(gcomp2, exportState=states(2), &
                clock=tclock, rc=rc)

      IF (rc.NE.ESMF_SUCCESS) finalrc = ESMF_FAILURE
      PRINT *,'grid comp initialize , finalrc = ',finalrc

! --------------------------- END Initializatons -------------------------------

! --------------------------- Carry forward the model  ------------------------
      DO outer = 1, nouter_step
        PRINT *,'outer step, finalrc = ',outer, finalrc
        ! get inputs that update per outer time step
          ! Get the meteorological forcing:
```

```
        CALL ESMF_GridCompRun(gcomp2, exportState=states(2), clock=tclock, rc=rc)
        IF (rc.NE.ESMF_SUCCESS) finalrc = ESMF_FAILURE

! ESMF run -- merge routines called here in to the ESMF run itself
!       Let the component manage its own inner-step business.
        CALL ESMF_GridCompRun(gcomp1, exportState=states(1), clock=tclock, rc=rc)
        IF (rc.NE.ESMF_SUCCESS) finalrc = ESMF_FAILURE

        CALL ESMF_ClockAdvance(tclock, timeStep=timestep)



      ENDDO

! --------------------------- END advancing model  -------------------------

      PRINT *,'about to close out the ESMF items, finalrc = ',finalrc

! Give each component a chance to write out final results, clean up.
      CALL ESMF_GridCompFinalize(gcomp1, exportState=states(1), &
                clock=tclock, rc=rc)
      IF (rc.NE.ESMF_SUCCESS) finalrc = ESMF_FAILURE
      CALL ESMF_GridCompFinalize(gcomp2, exportState=states(2), &
                clock=tclock, rc=rc)
      IF (rc.NE.ESMF_SUCCESS) finalrc = ESMF_FAILURE

      CALL ESMF_StateDestroy(states(1), rc=rc)
      IF (rc.NE.ESMF_SUCCESS) finalrc = ESMF_FAILURE
      CALL ESMF_StateDestroy(states(2), rc=rc)
      IF (rc.NE.ESMF_SUCCESS) finalrc = ESMF_FAILURE


! --------------------------- END producing Model Output  ------------------------


!--------------- Perform ESMF last closure steps ---------------------
      CALL ESMF_CalendarDestroy(gregorianCalendar, rc=rc)
      IF (rc.NE.ESMF_SUCCESS) finalrc = ESMF_FAILURE

      CALL ESMF_Finalize(rc=rc)
      IF (rc.NE.ESMF_SUCCESS) finalrc = ESMF_FAILURE
      IF (finalrc.EQ.ESMF_SUCCESS) THEN
          print *, "PASS: ESMF_IceModelEx.F90"
      ELSE
          print *, "FAIL: ESMF_IceModelEx.F90"
      END IF

      STOP
      END
```

This structure is quite general, and it's obvious that we could establish an array of states, an array of names, and an array of grid components – so that the model could be abstracted down to calling the init/run/finalize for each of modules numbered 1 through N – without any particular knowledge of what is in each of those modules. Systems aren't that simple in practice, as some things need to be computed before others, and there may well have to be coupling steps (ESMF_CplComp) between modules, for instance to translate variables between different grids. But it is the status of the v0 and at least v1 KISS models, where the sea ice is on the same grid as the weather model, and there is no interaction between them.

8. Conclusions

It runs, it uses ESMF, it can be run without using ESMF, and we do have a full framework for further sea ice modelling – web and ftp service, model verification measures, inter-model comparison, and a good collection of inputs and outputs to work with.

Current runs of the system are displayed at `http://polar.ncep.noaa.gov/develop/icemodel/` and will include later versions of KISS and other models as they become available. The username and password are available on request from the author. As the system progresses, it will move to `http://polar.ncep.noaa.gov/seaice/icemodel.expt/` (promotion from develop, with limited distribution and password protection, to general availability – though still many experiments to be done and items to be improved).

The source for this original version of KISS is on the EMC Subversion server under projects/mmab/seaice/kiss/tags/kiss_v0_original

Beyond the obvious Version 1 of the KISS model to follow, there will be documents for a new sea ice climatology, and a document examining specifically verification measures for sea ice concentration models.

9. Acknowledgments

I thank Xingren Wu, Nicole McKee, Mark Iredell, Mike Ek, Avichal Mehra, Sean Helfrich, and Todd Spindler for their help in making the connections that were needed for framing this system.

REFERENCES

ARCUS, 2009 et seq.: Study of environmental arctic change: Sea ice outlook. Arctic Research Consortium of the US, URL `http://www.arcus.org/search/seaiceoutlook`, URL `http://www.arcus.org/search/seaiceoutlook`.

Chen, C., T. Lakhankar, P. Romanov, S. Helfrich, A. Powell, and R. Khanbilvardi, 2012: Validation of noaa-interactive multisensor snow and ice mapping system (ims) by comparison with ground-based measurements over continental united states. *Remote Sens.*, **4**, 1134–1145.

Ek, M. B., K. E. Mitchell, Y. Lin, E. Rogers, P. Grunmann, V. Koren, G. Gayno, and J. D. Tarpley, 2003: Implementation of noah land surface model advances in the national centers for environmental prediction operational mesoscale eta model. *J. Geophys. Res.*, **108 (D22)**, 8851–, doi:10.1029/2002JD003296.

Grumbine, R. W., 1998: Virtual floe ice drift forecast model intercomparison. *Weather and Forecasting*, **13**, 886–890.

Grumbine, R. W., 2009: A posteriori filtering of sea ice concentrations. Tech. rep., MMAB. NOAA/NWS/NCEP/MMAB Technical Note 282, 8 pp.

Grumbine, R. W., 2013: Long range sea ice drift model verification. *in preparation.*

Grumbine, R. W., 2013, in preparation: History 1997-2012 of ncep sea ice concentration analysis. Tech. rep., MMAB. MMAB TN.

Halliwell, G. and R. Bleck, 2001: Energy loan sea ice model. HYCOM, unpublished, 2 pp., unpublished.

Hibler, W. D., 1979: A dynamic-thermodynamic sea ice model. *J. Physical Oceanography*, **9**, 815–846.

Hibler, W. D. and J. E. Walsh, 1982: On modeling seasonal and interannual fluctuations of arctic sea ice. *J. Physical Oceanography*, **12**, 1514–1523.

Hunke, E. C. and J. K. Dukowicz, 1997: An elastic-viscous-plastic model for sea ice dynamics. *J. Phys. Oceanogr.*, **27**, 1849–1867.

Hunke, E. C. and W. H. Lipscomb, 2008: Cice: The los alamos sea ice model. documentation and software user's manual version 4.0. Tech. rep., Los Alamos National Laboratory.

Lunde, B. N. and E. F. Coelho, 2009: Implementations of the navy coupled ocean data assimilation system at the naval oceanographic office. *IEEE/MTS Oceans '09*.

Mehra, A. and I. Rivin, 2011: NCEP/EMC/MMAB.

Modeling, E. S., 2012: Earth System Modeling, http://www.earthsystemmodeling.org/, http://www.earthsystemmodeling.org/.

Murphy, A. H., 1995: The finley affair: A signal event in the history of forecast verification. *Weather and Forecasting*, **11**, 3–20.

NCAR, 2012: National Center for Atmospheric Research, URL `http://www.cesm.ucar.edu/models/cesm1.1/`, URL `http://www.cesm.ucar.edu/models/cesm1.1/`.

NCEP/EMC, 2013: NCEP/EMC, http://www.emc.ncep.noaa.gov/index.php?branch=NEMS, http://www.emc.ncep.noaa.gov/index.php?branch=NEMS.

NCEP/EMC/GCWMB, 2005: Gfs implementation 2005. NCEP/EMC.

NCEP/EMC/MMB, 2011: Nmmb. NCEP/EMC.

NRL, 2013: Naval Research Lab, URL `http://www7320.nrlssc.navy.mil/hycomARC/prologue.html`, URL `http://www7320.nrlssc.navy.mil/hycomARC/prologue.html`.

Preller, R. H. and P. G. Posey, 1989: *The Polar Ice Prediction System – A sea ice forecasting system*, Vol. 212. NOARL Report, 45 pp., nOARL, Stennis Space Center, MS.

Rothrock, D. A., Y. Yu, and G. A. Maykut, 1999: Thinning of the arctic sea ice cover. *Geophys. Res. Lett.*, **26**, 3469–3472, doi:10.1029/1999GL010863.

Saha, S., et al., 2011: The ncep climate forecast system reanalysis. *Bulletin of the American Meteorological Society*, **91**, 1015–1057, doi: 10.1175/2010BAMS3001.1.

Thorndike, A. S. and R. Colony, 1982: Sea ice motion in response to geostrophic winds. *J. Geophys. Res.*, **87**, 5845–5852.

Tolman, H. and C. Caruso-McGee, 2011: Environmental equivalence. NCEP.

Waldrop, J., 1997: Sea ice model concentration verification. *unpublished*, **0**, 13.

Winton, M., 2000: A reformulated three-layer sea ice model. *J. Atmos. Oceanic Technol.*, **17**, 525–531.

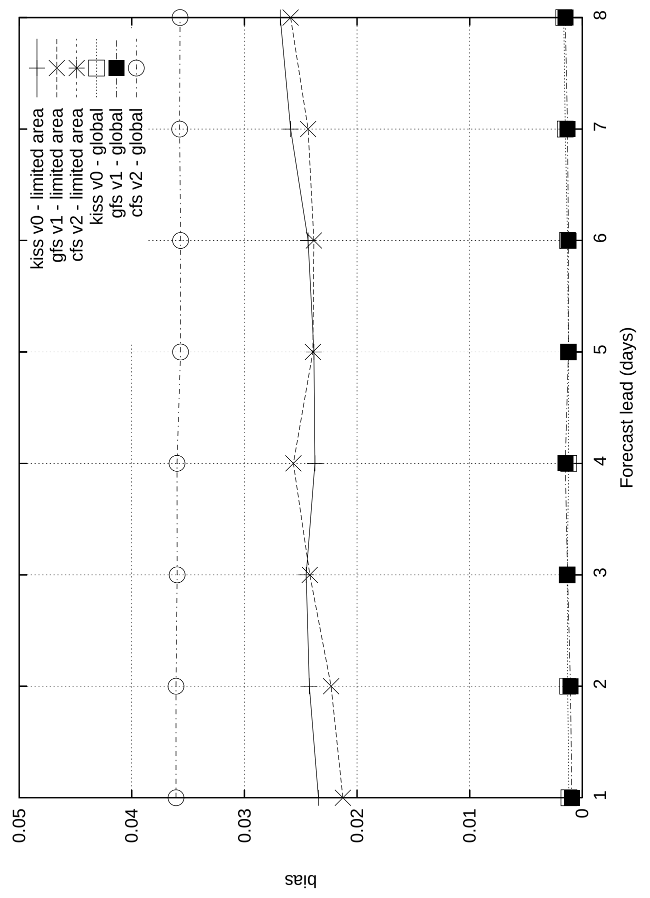
Wu, X. and R. Grumbine, 2013: Characteristics of sea ice in the ncep climate forecast system reanalysis. *Climate Dynamics*, submitted.

List of Tables

TABLE 1. Sample of Model Scores – Full Globe – Forecasts Valid 2012/11/21

Forecast Lead	Mean	RMS	A11	A12	A21	A22
01	0.004807	0.063994	0.048886	0.002865	0.008952	0.939297
02	0.004675	0.065385	0.048906	0.002887	0.008933	0.939274
03	0.004684	0.066393	0.048843	0.002873	0.008995	0.939288
04	0.004595	0.067235	0.049076	0.002990	0.008762	0.939172
05	0.004436	0.068279	0.049320	0.003238	0.008518	0.938924

List of Figures



bias
0.05
0.04
0.03
0.02
0.01
0
1
2
3
4
5
6
7
8
Forecast lead (days)
kiss v0 - limited area
gfs v1 - limited area
cfs v2 - limited area
kiss v0 - global
gfs v1 - global
cfs v2 - global

FIG. 1. Bias

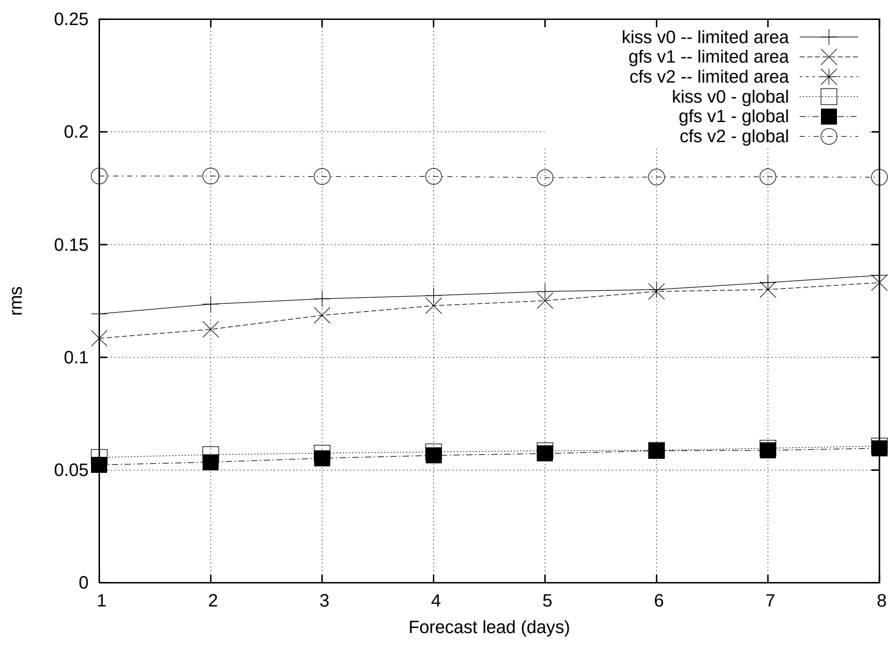

kiss v0 -- limited area
gfs v1 -- limited area
cfs v2 -- limited area
kiss v0 - global
gfs v1 - global
cfs v2 - global
rms
Forecast lead (days)
0.25
0.2
0.15
0.1
0.05
0
1
2
3
4
5
6
7
8

FIG. 2. RMS

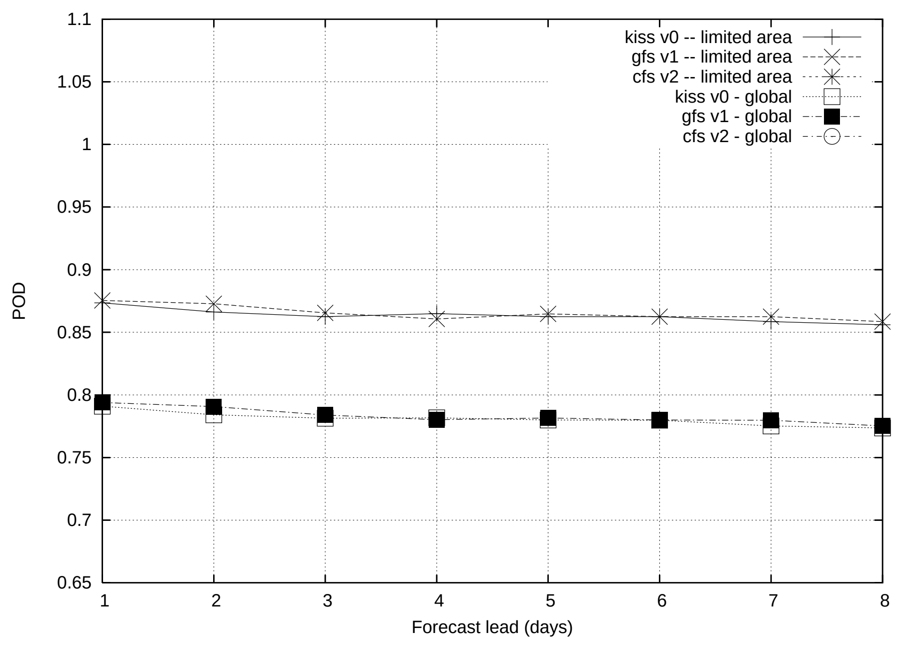


Fig. 3. Probability of Detection

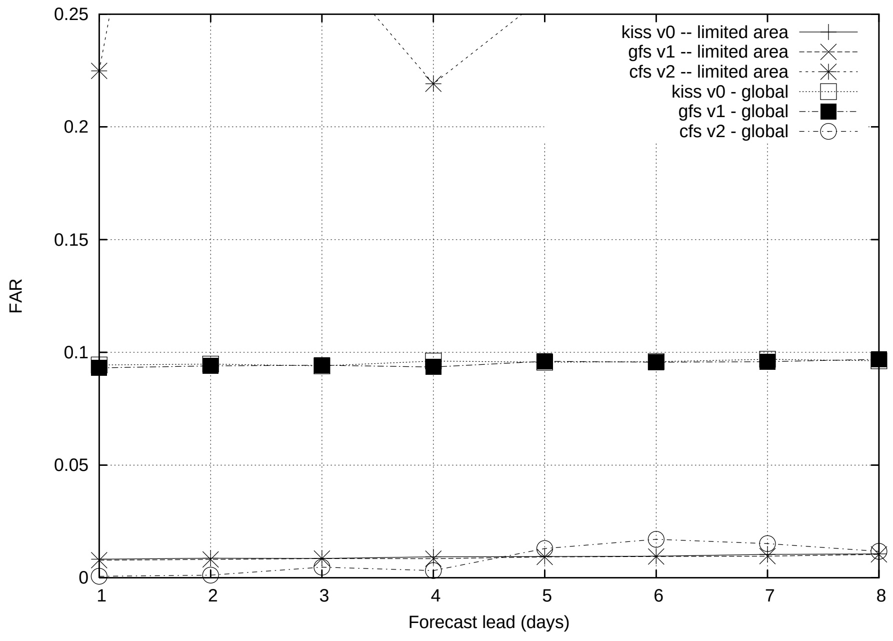


Fig. 4. False Alarm Rate

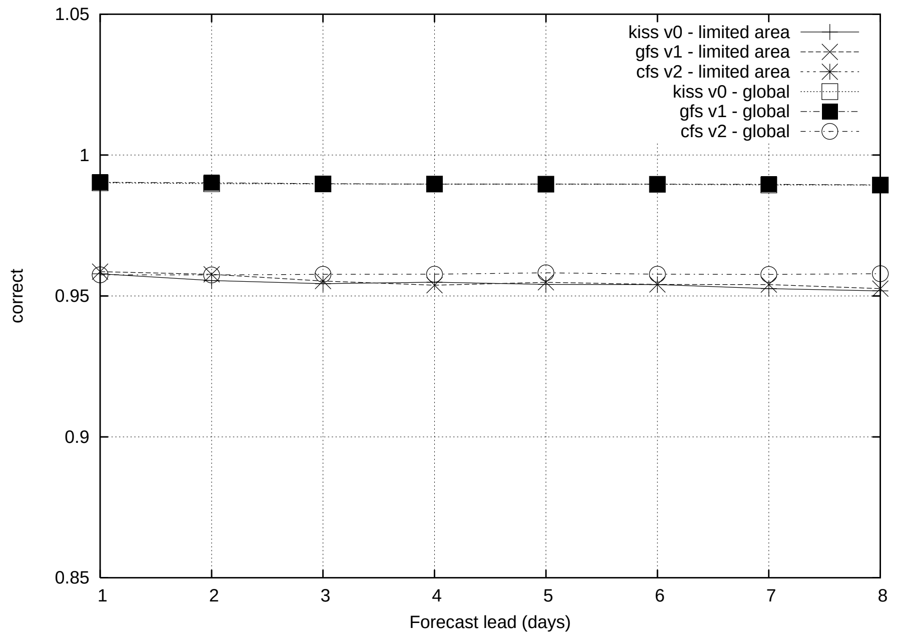


Fig. 5. % Correct