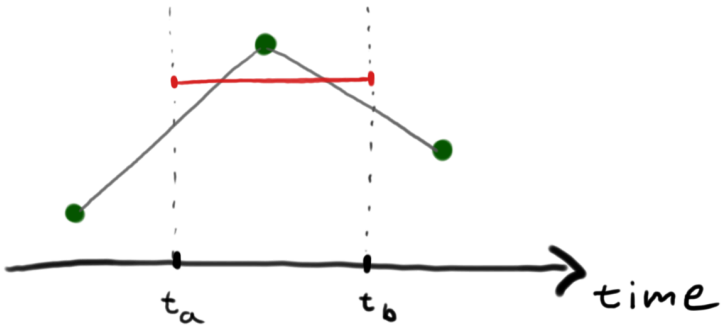
Diddling/Taylor procedure for ACCESS ESM 1.5

The following is a rough explanation of diddling based on the original PCMDI report by Taylor, Williamson, and Zwiers. The report is definitely worth looking at for a more authoritative explanation.

In the original AMIP I experiment, atmosphere models were forced with SST and sea ice ancillaries specifying monthly mean values in the middle of each month. Models would linearly interpolate between these monthly means to obtain daily values for use during running.

This approach is not ideal. If the interpolated daily SST and ice values used by the model are taken, and the monthly means of these time series are calculated, the result will not match the observed monthly means specified in the input data. In particular, amplitudes of variations will be damped:



Consider a month M running from times t_a to t_b in the above sketch. Observed monthly mean SST's for a grid cell are shown in green, and a linear interpolation between them in grey. The red line shows the monthly average computed from the linear interpolation. If observations have a local maximum in month M, the mean for month M recovered from the interpolation will be lower than the observation. The reverse will hold if there is a local minimum.

To address this issue for the AMIP II experiments, PCMDI introduced a process called "diddling" or the "Karl Taylor Procedure". Diddling modifies the monthly means in the input data so that the recovered means after linear interpolation match the observed unprocessed means.

Consider three months *i-1*, *i*, and *i+1*, all of the same length. If the observed monthly mean SST (or sea ice concentration) in month *i* is *S_i*, diddling will find modified monthly values T_{i-1}, T_i, and T_{i+1}, so that

$$S_i = \tfrac{1}{8}\, T_{i-1} + \tfrac{3}{4}\, T_i + \tfrac{1}{8}\, T_{i+1},$$

Where the right hand side is equal to the monthly mean calculated from the linear interpolation between T_{i-1}, T_i, and T_{i+1}.
Given a time period of n months, there will be n equations and $n + 2$ unknowns. When working with a climatology, periodic boundary conditions are applied making this system solvable. For a non-climatological time series, a more complicated process of adding artificial data at the boundaries is applied. The PCMDI paper has a good description of this process.

The actual calculation done in practice is a bit more complicated – most models use calendars with different length months of different lengths, and restrictions on SST and sea ice concentrations have to be taken into account (e.g. concentration between 0 and 1). This makes the equations to be solved nonlinear and an iterative method is used to do the calculation.

Available code for diddling

Two sets of diddling code are available online. One from the Met Office by Jeff Knight, and one on github repository from Duo Chan. Both sets of code appear to use code by Karl Taylor from 1997 for the actual diddling calculations, though the Met Office code is set up for a 360 day calendar and Duo's code is set up for the CESM model with a 365 day calendar.

These notes/tutorial are for a modified version of Duo's code, edited to work with ACCESS models using a gregorian calendar and regular lon/lat data (Note - currently only tested with the AMIP configuration of ACCESS ESM1.5). The code described by these notes is available on this github repository.

Diddling tutorial

Quickstart guide for running the bcgen software:

0. Start with:
    (a) A single input nc file containing sst and sea ice data, located in a folder for inputs
        - Sst variable named “SST”
        - Ice variable named “SEAICE”
        - SST in degrees C
        - SEAICE given as a percentage
        - Time given as "days since YYYY-MM-DD 00:00:00", which contains mid month values for the correct time period with no skipped months.
        - Time values need to be of type float
        - A “date” field containing the date of each timestep, given as an integer of form YYYYMMDD (with no leading zeros on the year, e.g. 5th March year 102 would be 1020305)
        - Gregorian calendar (adjustments to the diddling code are possible to change this)
        - Longitude and latitude named “lon” and “lat”

Then in bcgen_for_ACCESS.sh:
1. Set paths to input, output, and temporary directories
2. Specify name of input file
3. Set input/output time periods

Finally, run bcgen_for_ACCESS.sh. The software will produce a diddled version of the input data in a file named sstice_timeseries.nc, and a diddled climatology, sstice_clim.nc in the output directory.

Rough summary of the diddling software

The diddling code consists of the fortran code lying in the bcgen folder, and the bcgen_for_ACCESS.sh script lying in the parent directory.

bcgen_for_ACCESS.sh finds the input data, configures and saves it to the required format, and passes it to the bcgen software.

The fortran code in the bcgen directory then does the actual diddling calculations, with the main parts of the software as follows:

bcgen.f90: This reads the data, applies the diddling, and writes the output, calling several subroutines along the way. It contains a good explanation on the diddling procedure in the comments, particularly around the use of artificial buffers outside of the input's time period.

calcclim.f90 - I think this calculates the climatology for the period specified by mon1clm, iyr1clm, monnclm, and iyrnclm, and applies the diddling to this climatology.

calcfull.f90 - Carries out the diddling calculation for the whole time series (not climatology) over the user specified period.

solver.f90 - Function used by calcclim.f90 and calcfull.f90 to actually perform the calculations.

There are several other routines for things like reading and saving netcdf data which are called from bcgen.f90.

bcgen_for_ACCESS.sh

bcgen_for_ACCESS.sh sets up the input data, conforms it to the bcgen code's requirements, and calls the diddling software. This is the main file that needs to be edited to get the software running on Gadi. The user must supply the filepaths for the input.

The rough structure of this script is:

(i) - Set the user settings (file paths and desired time period)

(ii) - Move input files around
- Rename variables as required by diddling code
- Convert missing values to constants
- Combine SST and ice into a single file

(iii) - Save the modified dataset
- Compile the bcgen code
- Pass the modified dataset to the bcgen code

The specific changes bcgen_for_ACCESS.sh makes to the input data in step (ii) are:

- Changing the variable names to 'SST_cpl', and 'ice_cov'
- Changing the coordinate names to 'lon' and 'lat'
- Replacing missing values for sst with -1.8
- Replacing missing values for sea ice with 1
    - Important: I think this may just be in case the diddling code can't handle missing values. Be sure to remask the outputs from the diddling software.
- Converting the sea ice from a percentage to a fraction

- If your sea ice data is already given as a fraction, edit the following line so that it doesn't make any changes

```
ncap2 -s 'ice_cov=ice_cov/100.' temp2.nc ice_cov.nc
```

More detailed guide for running bcgen_for_ACCESS.sh

Directories and files

bcgen_for_ACCESS.sh will use of the following directories which need to be set up by the user:

- An input directory
  - Containing the input sst/sea ice file
- A Temporary directory
  - This will be filled with temporary netcdf files, but will be cleared automatically at the end.
- An output directory
  - Will be filled with the outputs from the diddling
- A tool directory
  - The directory containing the bcgen_for_ACCESS.sh script and the bcgen folder.

The user then needs to point to these directories at the start of bcgen_for_ACCESS.sh, e.g.

```
export dir_tool="/home/565/sw6175/prepping_forcings/CESM_BC"

# directory of SST and seaice data
export dir_data="/g/data/w40/sw6175/diddling/Merged_Hadley_OI/Inputs"

# Output directory
export dir_output="/g/data/w40/sw6175/diddling/Merged_Hadley_OI/Diddled"

# Directory for temporary files
export dir_temp="/home/565/sw6175/Merged_Hadley_OI/Temp"
```

The user then needs to point to the input file, which has the properties described in the checklist. The following path will be relative to dir_data

```
export input_data="sst_ice_regridded.nc"
```

Time period

The fortran code applies the diddling procedure to a time period specified by the following variables in the namelist file:

```
&cntlvars
mon1 = 1
iyr1 = 0
monn = 12
iyrn = 19
mon1rd = 1
iyr1rd = 0
monnrd = 12
iyrnrd = 19
mon1clm = 1
iyr1clm = 0
monnclm = 12
iyrnclm = 19
mon1out = 1
iyr1out = 0
monnout = 12
iyrnout = 19
```

A good explanation of each of these variables is given in both the readme file in the bcgen folder, and in the comments of bcgen.f90, but briefly:

`mon1rd, iyr1rd` are the first month and year to be read from the input file, while `monnrd` and `iyrnrd` are the final month and year read from the input file.

mon1clm, yr1clm, monnclm, and yrnclm specify the start and end period from which the climatology is calculated. E.g.

```
mon1clm = 1
iyr1clm = 1979
monnclm = 12
iyrnclm = 2000
```

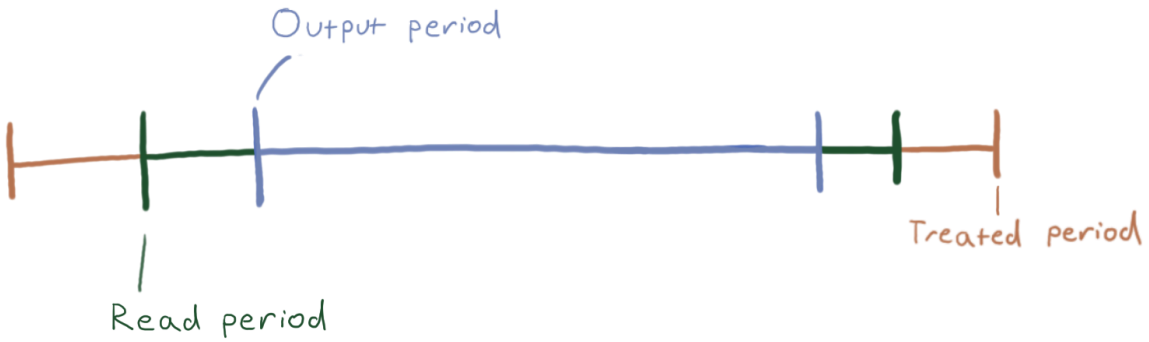
Means that the software will calculate a monthly climatology for the period between Jan 1979 and Dec 2000, and then apply the diddling to this climatology.

The diddling calculation (i.e. the calculation of modified monthly means) is mathematically tractable for the climatology as periodic boundary conditions can be applied (i.e. the first month is assumed to follow the last). When performing the calculation for the whole time series however, there are more monthly means to solve for than equations, and hence a buffer of artificial data needs to be added to the ends to make the problem solvable.

The comments at the start of bcgen.f90 explain how this artificial buffer is created, but loosely the values at the start and end of the input data are made to gradually decay to the climatology over a specified length of time, and then periodic boundary conditions are then applied to this extended dataset.

The duration of the buffer is specified by the `mon1`, `iyr1`, `monn`, `iyrn` variables. These specify the start and end of the “treated period”, which includes the read period + buffers for artificial data at the start and end. The comments in bcgen.f90 recommend a total buffer length of a year or two. The total treated period must additionally be an integer number of years (I think so that the periodic boundary conditions applied to the extended period make sense).

The values of the artificial data at the ends will affect the diddled monthly means within the actual period of interest, however their impact decays quickly with distance from the boundaries. Hence it's recommended to only write data for a subset of the read period, throwing away the first and last 2-3 months. The period which the output is written for is specified by the `mon1out`, `iyr1out`, `monnout`, `iyrnout`.



Each of these time period variables need to be specified by the user in bcgen.f90:

```
# SPECIFY the first and last month and year for period in which observed
# monthly mean data will be read. (The first month read must not preceed mon1,
# iyr1, and the last month must not follow monn, iyrn).

export mon1rd=1
export iyr1rd=1870

# Let's read 40 years
export monnrd=12
export iyrnrd=1910

# SPECIFY first and last month and year for entire period that will
# be treated (i.e., the period of interest plus buffers at ends).
# Note that the entire period treated should be an integral
# number of years.

export mon1=1
export iyr1=1869
export monn=12
export iyrn=1911

# SPECIFY the first and last month and year that will be included in
# climatological mean.  (This must be an interval within the
# observed period).

export mon1clm=1
export iyr1clm=1880
export monnclm=12
export iyrnclm=1895

# SPECIFY the first and last month and year written to the output
# file.  (Try to begin at east a few months after the mon1rd,
# iyr1rd, and end a few months before monnrd, iyrnrd, to avoid
# sensitivity to the artificial data outside the observed
```

```
# period.)

# SW: In bcgen.f90 it's recommended to exclude 2-3 months on either side
export mon1out=4
export iyr1out=1870
export monnout=9
export iyrnout=1910
```

Running the code

Once the files and time period are set up, all that needs to be done is to run CESM_E_to_F.sh (probably in a PBS job). The required modules are loaded in the script, so don't need to be added in by the user.

Output files

The software will produce a climatology, sstice_clim.nc, and time series sstice_timeseries.nc in the output directory, which contain diddled sst and ice variables placed at mid month times on a Gregorian calendar. The two files also contain copies of the original, unprocessed sst and ice in the "_prediddle" fields.

Any missing values in the input file will have been set to -1.8 deg C for sst, and 1 for sea ice. It will likely be necessary to remask out the land points using a copy of the model's land sea mask.

ACCESS ESM1.5 expects the SST ancillary to have units K, and hence the SST's will need to be converted if using this model. The helper script taylored_to_ESM_format.py on the github takes the output of the diddling software and formats it for use with ACCESS ESM1.5 - applying the masking, conversions, and setting the attributes and encodings to match those from the original AMIP ancillary files. The resulting nc files will still need to be converted to the UM format, e.g. using xancil.

Appendix

Appendix 1: Customising calendars

This version of the bcgen software has been modified to work with a Gregorian calendar.

The calendar is set up in lines 406-409 of bcgen.f90

```
 integer :: monlen(12,2)              ! length of months

   data monlen/31, 28, 31, 30, 31, 30, 31, 31, 30, 31, 30, 31, &
               31, 29, 31, 30, 31, 30, 31, 31, 30, 31, 30, 31/
```

To change back to a 365 day calendar, we just replace 29 in the second row of the monlen array with 28. (It's best to just leave monlen array as a 2D array, as the code will try to swap between the columns every leap year)

The file setup_outfile.f90 then needs to be edited to change the type of calendar in the output file. I.e. replace line 61 from

```
call wrap_nf_put_att_text (ncid, timeid, 'calendar', 'gregorian')
```
to
```
call wrap_nf_put_att_text (ncid, timeid, 'calendar', '365_day')
```

The same changes should work for a 360 day calendar, though we haven't tested this.

Warning: When making changes to the calendar, it is worth double checking that the time values (i.e. "days since reference date") are correct. Previous issues existed where the diddling calculation was performed for a Gregorian calendar, and the output was given a Gregorian calendar, however the actual time values ("days since reference date") were calculated with respect to a 365 day calendar, meaning that the output's dates drifted by one day each time a leap year occurred. This should be fixed via changes to the calculations in output_dateinfo.f90, but it's worth double checking the output if you make any changes to the calendar.

Appendix 2: Non-regular grids

The original version of the bcgen code from Duo Chan's github repository was set up for a non-cartesian grid. Zoe GIllet has made some changes which let it work with the ACCESS atmosphere's cartesian grid. If using a non-cartesian grid, revert the changes via the following modifications:

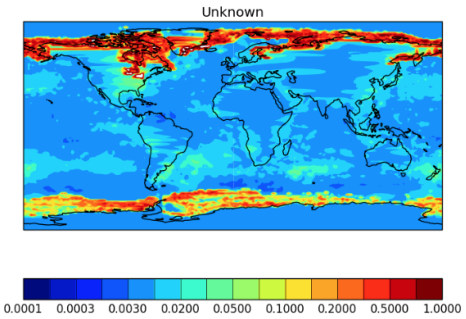
In bcgen.f90 and setup_outfile.f90, replace:
```
xlon(nlon) -> xlon(nlon,nlat)
xlat(nlat)  -> xatl(nlon,nlat)
```

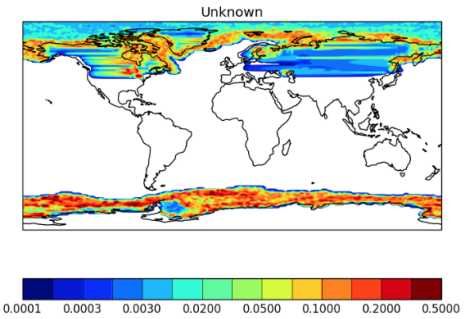
and in setup_outfile.f90, replace:
```
wrap_nf_def_var (ncid, 'lon', NF_DOUBLE, 1, dimids(1), lonid)
->
wrap_nf_def_var (ncid, 'lon', NF_DOUBLE, 2, dimids, lonid)

wrap_nf_def_var (ncid, 'lat', NF_DOUBLE, 1, dimids(2), latid)
->
wrap_nf_def_var (ncid, 'lat', NF_DOUBLE, 2, dimids, latid)
```

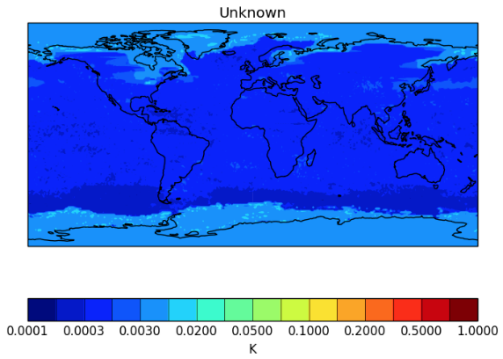
Appendix 3: Regridding and diddling

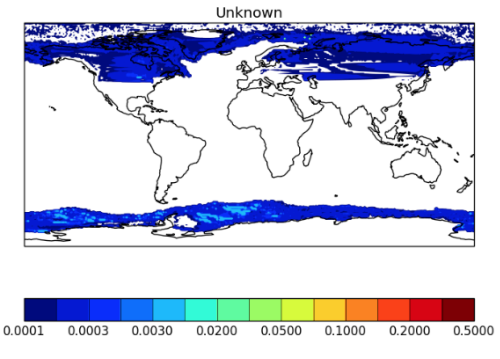
If data needs to be regridded while preparing ancillary files, regridding should always be carried out before the diddling. The following figures [from the Met Office site] shows the maximum monthly error in the recalculated means after interpolation for SST and sea ice for data that has been regridded after diddling.





The errors are significantly larger than the errors when regridding is done before the diddling:





The explanation from the Met office for why the errors are so much larger when regridding after diddling, is that the diddling calculation makes independent adjustments to each grid cell. Regridding can then mix these adjustments together, leading to the larger errors.

Appendix 4: Checking the output - Tayloring simulator code

The script taylor_test.py (located on the github) can be used to check whether the bcgen software worked.

Based on similar code from the UKMO, this script does the following:

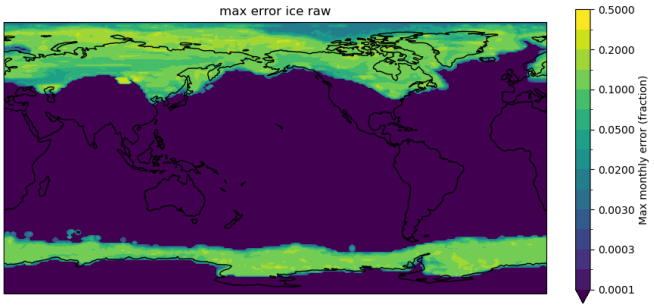
1. Take the processed SST and sea ice output from the bcgen software and linearly interpolate it from monthly to daily values.
2. Clip the data to physical values (e.g. > 271.35K for the SSTs).
3. Recalculates the monthly means from the interpolated and clipped data.

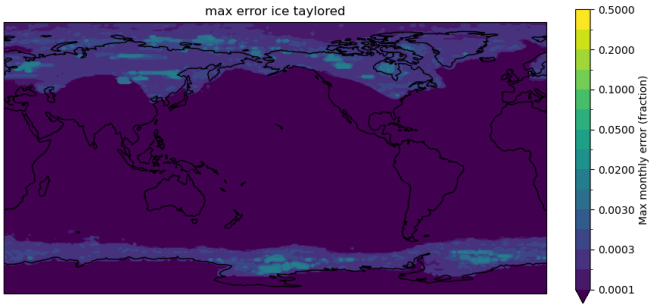
4. Plot the maximum monthly error between the recalculated monthly means and the original unprocessed data.

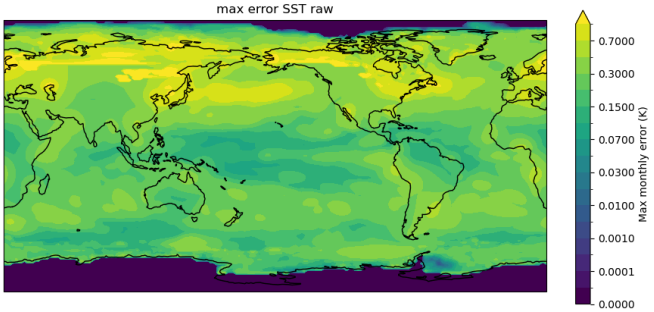
Steps 1 and 2 emulate how an atmospheric model handles the diddled data. Ideally if the bcgen software worked, the errors between the recalculated monthly means and the original data will be very small.

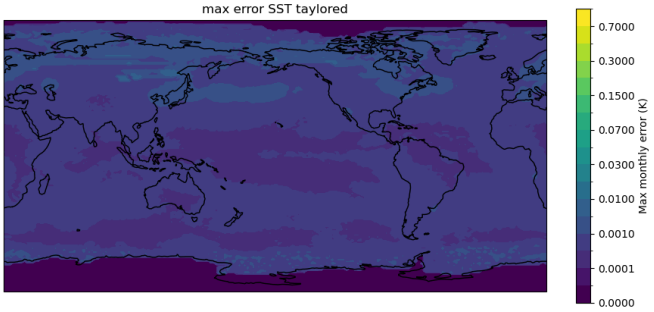
The script also applies the above four steps to the original undiddled data (held in the variables SST_cpl_prediddle and ice_cov_prediddle in the bcgen output), in order to demonstrate how the errors would look if the original data had been directly supplied to a model without applying the diddling process.

The following figures were produced after running the bcgen software and the taylor_test script on ERSST v5 SSTs and Merged Hadley-OI sea ice from 1979-2014:









It worked!