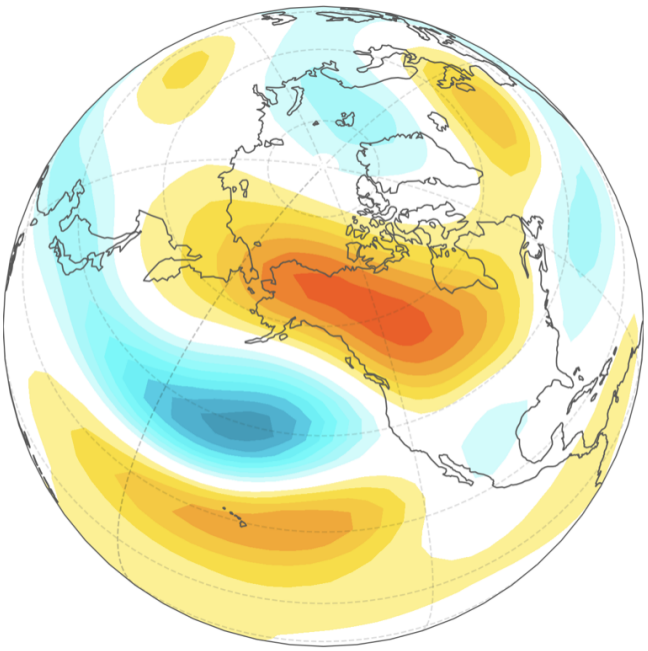


DREAM user manual

Release 8.4

nick hall, stephanie leroux

Jul 16, 2024

Chapters

1 Chapter 1: Inception - Introduction to DREAM

1.1 Preamble

In 1983, Sir Brian Hoskins published a lecture in the QJ under the title "Dynamical processes in the atmosphere and the use of models" in which he commented on the interplay between observations, sophisticated numerical models and our understanding of dynamical processes. He advocated a hierarchical approach in which dynamical models of intermediate complexity play a role (I made a contribution to this vision in Hall 2004, but I don't think anyone read it). There is always a tradeoff between physical realism and conceptual understanding. Models of varying complexity can bridge the gap.

Four decades later, numerical modelling of the atmosphere and climate is more complex than ever. It has become a truly interdisciplinary endeavour shared by communities and worked on in teams, with a wide range of applications. At the same time, at a much more modest scale, a few simple models of the atmosphere have emerged to tackle various questions in a more restricted and often more economical way. DREAM is one of these models.

The real world is complicated but our brains are limited, and indeed we are in the process of replacing them altogether with something more efficient. For those who still seek understanding, some sort of model hierarchy is needed. A model whose results are realistic enough to be useful, but whose physical specification is simple enough to offer some insight. One way to achieve this is to allow some of the solution to be constrained by data. There are several ways in which DREAM can interact with data that will be explained in this guide. This class of model is sometimes referred to as a "diagnostic model".

DREAM can produce realistic simulations of the atmosphere based on dynamics alone. It can also be used in highly idealised configurations. Because of the way its forcing can be manipulated, DREAM is not just a model. It is a hierarchy of models. DREAM stands for Dynamical Research Empirical Atmospheric Model. It is a dynamical model intended mainly for research. It is not a statistical model, but since some elements are taken from data, the word Empirical appears in the acronym. This guide is an overview of how to use the model. It is part technical manual,

part research manual as I will try to explain not only the scripts and the code, but also some diagnostic techniques that arise naturally from using DREAM.

1.2 Empirical forcing and data integration

The problem of how to force a simple dynamical model to produce an equilibrium climate solution has often been approached by using relaxation forcing. An idealised radiative convective equilibrium temperature field is specified, and the model temperature field is relaxed linearly towards it on a chosen timescale. Since the radiative convective equilibrium is hydrodynamically unstable, the atmosphere never attains this state, nor even approaches it closely (unless a very short relaxation timescale is imposed). So it is difficult to appeal to data to deduce directly what this state should be.

DREAM is based on an alternative approach first used by Roads (1987) to make a cheap forecast model. A sequence of observed initial conditions is used and the unforced model is integrated for just one timestep. The negative average of the one-timestep tendencies thus produced is then adopted as the forcing on the right hand side of the model equations. We assume that the set of initial conditions represents a stationary climate state, and that the forcing of the atmosphere can be viewed as time independent. Under these assumptions, this forcing represents the missing term that corrects the systematic errors of an unforced model.

Some form of dissipation is also included, and the combination of forcing and dissipation is mathematically equivalent to the relaxation forcing approach discussed above. But instead of specifying an unknown equilibrium state and a dissipation timescale, we just specify the dissipation and appeal to data to furnish the forcing.

The result is a GCM that is simple in conception, based entirely on dynamics, but gives realistic enough simulations to be used for a variety of climate studies. DREAM can also be used in perturbation mode. The only thing that changes is that the forcing is designed to hold a fixed basic state in place, a technique first employed by Jin and Hoskins (1995). A perturbation is then added either to the forcing, as was the case with Jin and Hoskins, or to the initial condition (see for example Hall and Sardeshmukh, 1998). The model solution arising from the perturbation can be analysed. If the perturbation is constrained to be small, the solution will be linear.

The data used with DREAM comes from four-times daily ERA-interim reanalysis from 1979-2016 (Dee et al, 2011). DREAM interacts with a version of this dataset that is written in the same spectral basis as the model output. The data can also be used to nudge selected regions of the model throughout the integration. A version of the empirical forcing with an annual cycle is also available.

1.3 How is DREAM different from other simple GCMs ?

DREAM is based on the spectral primitive equation model code first introduced by Hoskins and Simmons (1975, HS75). This code was initially used to study baroclinic wave lifecycles and quickly became one of the staple tools for dynamical studies at the Reading University Meteorology department, pressed into service for a diverse range of topics in synoptic meteorology and global climate dynamics. As a purely dynamical model, it lacks the physical source terms that maintain the climate, so for longer integrations, or for studies of perturbations on a specified basic state, a prescription is needed for the right hand side of the equations. Various solutions have been implemented, and as the model was used for more sophisticated applications, further representations of physical processes were added. At the same time a sequence of technical modifications were also made to simplify the code and make it more portable. That model is now known as the "IGCM" and the current version (Joshi et al, 2015) is shared by a consortium of users at UK universities.

DREAM has the same dynamical core as the Reading IGCM, but it has far fewer physical parameterisations available and has not been developed by a community like the Reading model. Instead the development has been more in the

direction of working with data for diagnostic modelling. The paraemterizations in DREAM are home-grown, and semi-empirical.

Here's a list of some models that fit into the simple GCM niche:

- IGCM - As mentioned above, based on code originally developed at Reading University back in the 1970s and identical to DREAM in its dynamical core. Runs as a physically based GCM with simple parameterisations.
- SPEEDY - Developed at ICTP in Trieste. Based on the GFDL spectral dynamical core. Runs as a physically based GCM with simple parameterisations.
- PUMA - Developed at the University of Hamburg. Based on the ECHAM spectral dynamical core (which in turn was derived from the ECMWF model, which originally evolved from the Reading model). Works with restoration forcing and relatively simple dissipation.
- ISCA - Developed at the University of Exeter. Based on the GFDL spectral dynamical core. Can be forced at various levels of complexity from relaxation to full radiation. Applicable to Earth and other planets.

1.4 What can you do with DREAM ?

In its simplest form DREAM is a dynamical model that either simulates a perpetual season or can be used to study perturbations about a fixed basic state, in both cases with time-independent forcing. An extension to the forcing has been developed that includes an annual cycle. In this way DREAM can be used for climate studies in which a one-to-one correspondence with historical data is required. DREAM is designed to work with the reanalysis data that is used to calculate the empirical forcing. This dataset can be consulted during a run to constrain predefined regions of the world to a time sequence of observations by "nudging" on a specified timescale. Further extensions to DREAM have been developed, including interaction with moist thermodynamics associated with the model's specific humidity variable, and the response to tropical SST anomalies. DREAM has been used for a wide range of applications, including fundamental properties of the midlatitude jets and baroclinic waves, easterly waves over West Africa, teleconnections from tropical convection, the annual cycle and seasonal prediction of continental rainfall. A complete list of publications is given in Appendix E.

1.5 Some references

- Dee D.P., et al, 2011: The ERA INTERIM reanalysis: Configuration and performance of the data assimilation system. Quart. J. Roy. Meteor. Soc., 137, 553–597.
- Hall, N.M.J., 2000: A simple GCM based on dry dynamics and constant forcing. J. Atmos. Sci., 57, 1557-1572.
- Hall, N.M.J., 2005: The atmospheric response to boundary forcing and the use of diagnostic models. ERCA6, EDP sciences, C. Boutron ed. J. Phys IV, France, 121, pp 125-137.
- Hall, N.M.J., S. Leroux and T. Ambrizzi, 2019: Transient contributions to the forcing of the atmospheric annual cycle: A diagnostic study with the DREAM model. Climate Dyn. https://doi.org/10.1007/s00382-018-4539-y
- Hall, N.M.J. and P.D. Sardeshmukh, 1998: Is the time mean Northern Hemisphere flow baroclinically unstable ? J. Atmos. Sci., 55, 41-56.
- Hoskins, B.J., 1983: Dynamical processes in the atmosphere and the use of models. Quart. J. Roy. Meteor. Soc., 109, 1-21.
- Hoskins, B. J., and A. J. Simmons, 1975: A multi-layer spectral model and the semi-implicit method. Quart. J. Roy. Meteor. Soc., 101, 637–655.
- Jin, F., and B. J. Hoskins, 1995: The direct response to tropical heating in a baroclinic atmosphere. J. Atmos. Sci., 52, 307–319.

- Joshi, M., M. Stringer, K. van der Wiel, A. O'Callaghan and S. Fueglistaler, 2015: IGCM4: a fast, parallel and flexible intermediate climate model. Geosci. Model Dev., 8, 1157–1167.
- Roads, J. O., 1987: Predictability in the extended range. J. Atmos. Sci., 44, 3495–3527.
- Simmons, A. J., and D. M. Burridge, 1981: An energy and angular-momentum conserving vertical finite-difference scheme and hybrid vertical coordinates. Mon. Wea. Rev., 109, 758–766.
- Simmons, A. J., and B. J. Hoskins, 1978: The lifecycles of some nonlinear baroclinic waves. J. Atmos. Sci., 35, 414–432.

2 Chapter 2: Eyes Wide Shut - A Quick Start Guide to Running DREAM

2.1 Accessing the model

The DREAM code and a selection of data is currently hosted on GitHub: https://github.com/dream-gcm/DREAM

You need to join the private DREAM project on github, so if you don't already have a github account, the first step is to create one. Then contact us through the link on our web site to be invited to the DREAM private github page. Once you have been invited to the page, you'll be able to download the model. Go to the model page, click on the green Code button to Download ZIP. Unzip it and you're done.

2.2 Compilation on your system

Now you're on your computer and you have either downloaded and unzipped the repository or `git clone` it. If what you downloaded is named "DREAM-master" then rename this folder as DREAM, put it wherever you want, and go into this directory.

First you'll need to recompile the model libraries. Go into the `dream_model/source` directory to compile the model libraries on your computer (you should only have to do this once):

```
cd ./lib
./makelib.csh
cd ../..
```

and now you should be back in the dream_model directory.

2.3 Your first test simulation

Go into the DREAM/`dream_model/jobs` directory and open the script `runmodel_v8.4.ksh` (at time of writing).

Modify the root path $KD so it points to the DREAM directory: KD=`/Users/yourname/`DREAM - for example

We recommend you compile the model with gfortran (so gfortran must be installed on your computer at this stage, along with its netCDF dependencies). Make sure that $FC is set to the name with which to call the fortran compiler on your machine.

The script writes a file that contains two namelists: SETUP and INITIAL. They contain the basic run parameters and a vast array of model parameters. For a first test run set:

- KRUN=`320` (your test experiment will be run for 320 timesteps, i.e. 320/64 = 5 days),
- RUNTYPE=`"PERPETUAL"` (the model will run with fixed time independent forcing),
- THERMTYPE=`"DRY"` (no moist thermodynamics).

Everything else should just behave itself and you will discover all the other options as you gain experience.

The model also reads a number of files as input data and these can be seen in the last part of the script, assigned to the following fortran channels:

Essential:

- `10` - the initial condition - this can be anything you want - spectral
- `13` - the basic forcing - this will drive the model to simulate the season you choose - spectral
- `16` - the reference state - this must match the season chosen for the forcing - spectral
- `19` - the land-sea mask - grid

Optional:

- `20` - a nudging state - spectral,
- `15` - a forcing perturbation - spectral,
- `17` - either the full SST or a perturbation (SSTA) - grid,
- `18` - the climatological mean or reference SST - grid.

These data files all reside in DREAM/`dream_data` and its subdirectories. If there is a problem and the run doesn't complete it will probably be because one of these files is specified incorrectly.

To run the model execute the script in a terminal:

```
./runmodel_v8.4.ksh
```

The model will run, developing over five days from its initial condition, subject to the forcing you have specified.

2.4 Postprocess the model output

If the model has run without errors, some output files have now been produced and stored in DREAM/`dream_results/$EXPDIR` (the default name for $EXPDIR is test_run).

The post processing code resides in DREAM/`dream_model/diagnostics`. There are many scripts here that produce a large variety of diagnostics, including sequences, time means and fluxes with output options for netCDF and binary grid.

The simplest operation is to write a time sequence of netCDFs on model levels, using defaults for the choice of variables and levels. Edit the script run_output.ksh so it contains your path (set KD) and choose the values of logical switches lmean - if you want to calculate the time mean of your output and lseq if you want a sequence.

Execute this script from within its directory

```
./run_output.ksh
```

and if all goes well, you will find some netCDFs in your experiment directory DREAM/`dream_results/test_run/netcdfs`

The probability of all going well is less than unity ! Typically there is some work to do to make sure your version of netCDF is compatible with your fortran compiler.

The netCDFs are labelled with reference to resolution, variable name and level. So for example `dreamT31L15_u_250.nc` is output from the model run at T31 with 15 levels, zonal wind at 250 mb
(but it's not really millibars - in fact it's output on the sigma=0.25 level, but it is close to data on a pressure surface because DREAM runs without orography and the surface pressure in the data is actually mean sea level pressure - see Appendix A).

Now you can plot these variables and compute any diagnostics you want with your favourite tools and programs for dealing with netCDF datasets.

2.5 Further notes on installation

gfortran is strongly recommended, with netCDF libraries and big endian option.

Example of the compilation line for the model code:

```
gfortran -fdefault-real-8 -fconvert=big-endian -fno-align-commons -w -O3
 ↪-I/opt/local/include/ -L/opt/local/lib -lnetCDF -lnetCDFf
```

With gfortran version > 10.1.0 you need to add the option `-fallow-argument-mismatch` to the compiler when you compile the diagnostics code calling for the netCDF libraries.

Example of the compilation line :

```
gfortran -fdefault-real-8 -fconvert=big-endian -O3 -frecord-marker=4 -w -
 ↪fallow-argument-mismatch -fno-align-commons specan_W2G.f -o a.out -I/
 ↪usr/local/include/ -I./include -L/usr/local/lib -lnetCDF -lnetCDFf -L/
 ↪yourpath/DREAM/lib -lfft -lblas -lutil -laux
```

2.6 So what else do you need to know ?

- I want to know where are all the files are stored and what are they're all for - see Chapter 3.
- I want to go further with diagnostics and understand the data structure and timing - see Chapter 3 and Appendix A.
- What is the physical and numerical specification of this model ? - see Appendix A.
- What's the difference between a GCM run and a perturbation experiment ? - see Chapter 4 and Appendix B.
- How is the basic forcing calculated ? - see Chapter 4.
- How do I make an anomaly forcing file ? - see Chapter 4.
- What are all those parameters in the namelist for ? - see Appendix C.
- I want to run ensemble forecasts, with an annual cycle, and nudging, and SST anomalies, and deep convection - whoah, calm down, one step at a time, it's all in Chapter 4.
- I want to know everything about how the model works - I'm afraid you have no choice but to look at Appendix D, good luck.

and finally…

- I've successfully used DREAM and my results are astonishing ! I think this is a major contribution to atmospheric science which will launch my career. My first draft is ready - Congratulations ! Now don't forget about the rest of us - see Chapter 5.

3 Chapter 3: The Matrix - Description of the DREAM File Structure

The sub-headings in this chapter refer to directory names under the DREAM root directory. As well as giving you a guide to the file structure, we'll also have a look inside and see what data there is, and what code is available for pre-processing and manipulating the data, running the model and post processing the output. This chapter is mainly a guide to the files available with a few details where appropriate, especially for the data files. A more comprehensive guide to running the model and diagnosing the output follows in Chapter 4.

3.1 `dream_data`directory

The first of three main branches from DREAM is `dream_data`. There are four sub-directories as the data is split into two resolutions, and into spectral and grid data. So T31 spectral data goes with G96 grid data, and T42 spectral data goes with G128 grid data.

3.1.1 Spectral files

Let's look at spectral data first. These files are all in the same format and contain the base variables of the model: vorticity, divergence, temperature, log surface pressure and specific humidity. Details of the file structure are given in Appendix A, but for now let's look at the files in the different subdirectories. Note that the file names can be the same for T31 or T42 so it is important to keep them in their correct directories.

i) ave: Average

Contains fortran binary files (extension .b) for time-mean climatologies. The annual mean of the analysis period 1979-2016 is given (ANN) and the four seasons DJF, MAM, JJA and SON. For example

ERAi_ave4x_1979-2016_DJF.b is ERA-interim, average of 4x-daily data, from 1979 to 2016, for the December, January February season. These files are a single record. They are often used as reference states for the model (channel 16) and sometimes for the initial condition (channel 10), especially for perturbation stationary wave experiments (see Chapter 4, section 1).

As well as these mean climatologies, there are some idealised fields, which are derived from the mean fields, for example the global mean resting basic state ERAi_ave4x_1979-2016_ANN_GM_REST.b which is derived from the annual mean, but with globally uniform fields of temperature and humidity and no winds. There are thus no horizontal gradients, but the stratification is retained.

Another example ERAi_ave4x_1979-2016_DJF_ZM.b is the time-mean zonal mean from DJF and ERAi_ave4x_1979-2016_DJF_ZM_SYM.b is the same but symmetrical about the equator. The code to create these files is outlined in the next section.

ii) cyc: Cycle

ERAi_cyc4x_1979-2016_RM41.b contains the mean annual cycle over the analysis period. This file is 4x daily data over one model year of 365.25 days. So it has 1461 records. It is based on a simple mean by calendar date which has then been smoothed with a 10-day running mean (RM41 for 41-point running mean).

iii) fan: Forcing Anomaly

Still in the same data format, but these files are actually tendencies rather than states. They are mostly idealised heating fields but can be used to add a perturbation forcing to any model variable, read into channel 15. They are usually just one record for a time independent perturbation but they can also be a sequence. So for example CPac_EQ180E_40x15_Deep_2dpd_fan.b.

Is a Central Pacific heating anomaly centred at the equator and 180 degrees longitude. It is elliptical in shape with semi-major and minor axes of 40 degrees in longitude and 15 degrees in latitude. The horizontal distribution is a cosine squared bell shape. The heating is injected into a deep convective profile (peaking at sigma=0.35) with a vertical average heating rate of 2 degrees per day.

_fan files can be more sophisticated, with for example a canonical MJO sequence. The model can be set to read sequential records at a user-defined rate. When the model reaches the end of the file it will return to the beginning for a cyclic perturbation.

iv) fbs: Forcing the Basic State

These are single-record files that contain tendency information in the standard spectral format. They are read into the channel 13 to provide the basic forcing for the model. If an _fbs file is chosen to force the model it should be used in conjunction with the correct initial condition, which corresponds to the desired basic state. This is because an _fbs file contains exactly the value required to cancel the tendency that the unforced model would have if it were integrated one timestep forward from that basic state. So if the basic state is used as an initial condition in conjunction with its _fbs file, there will be no development in the model. It will step forward in time and its state will remain the same as its initial condition. So if you read fbs/ERAi_ave4x_1979-2016_DJF_fbs.b into channel 13, and use ave/ERAi_ave4x_1979-2016_DJF.b in channel 10 and channel 16, the model will just sit on that state until rounding errors grow to finite amplitude through instability (this usually takes a few hundred days).

Of course this configuration is intended for use in experiments where there is a perturbation either to the forcing (a _fan file) or to the initial condition. More on this in Chapter 4, Section 4.2e.

v) fcm: Forcing a Circulation Model

This is technically the same as the _fbs file described above, but it will not maintain any basic state. If you read an _fcm file into channel 13, the model will develop regardless of the initial condition you choose in channel 10. But note that it is still important to use the correct associated reference state in channel 16.

This is the simple GCM forcing, which can be used for long runs or forecasts to get the model to behave like a GCM. The model will develop a fully turbulent eddy field with a mean state and transient fluxes that resemble those of a real GCM (or even the real atmosphere). Often used in experiments where there is a long perturbed equilibrium simulation, to be compared with a long control run. Forcing for perpetual runs simulating each of the four seasons is provided.

Most of the files in this directory are single record files, designated ave4x, but there is one that is designated `cyc4x`, which has 1461 records. This is the annual cycle forcing described in Chapter 4, section 2d. If the annual forcing cycle is chosen, then the reference state must be the mean annual cycle. The model will read forcing and reference data sequentially every six hours and return to the beginning of the year at the end of the files.

vi) inst: Instantaneous data snapshots

Useful as initial conditions to get the model started in a realistic way, or to throw some multi-scale noise at an idealised simulation to break symmetry. These are single record data files where the date is specified.

vii) seq: Sequence

These are multiple record data files that can be used directly for pre-processing, for calculating model forcing functions, for nudging, or for ensemble initial conditions. For example `ERAi_seq4x_1979-2016_ANN.b` contains the entire dataset from 01/01/1979 to 31/12/2016: 55520 records. This has been split into datasets with sequential discontinuous chunks for the four seasons: `ERAi_seq4x_1979-2016_DJF.b`, etc.

There are also sequences for a given date: `ERAi_seq_1979-2016_01_01.b` contains 38 records with every first of January, useful for ensemble forecasts. And `DREAM_dry_seq_120_28d_interval.b` is actually output from a long perpetual DJF integration of DREAM, sampled once every 28 days. So there are 120 records of independent data sampled from the model's own climate: useful for drift-free idealised ensemble work.

3.1.2 Grid files

The only essential grid file the model needs is the land-sea mask, `LSmask_G96.b` for T31 and `LSmask_G128.b` for T42. Apart from that the most important grid-based input to the model is the SST data. Single record climatologies are given for the four seasons, and a monthly climatology is also provided in a 12-record file. These climatologies are used in conjunction with either realistic or idealised SST anomalies to calculate the associated atmospheric heating anomaly. This is still subject of research.

3.2 `data_process` directory

In the second main directory, dream_model, we have a set of routines to manipulate the data and prepare data files and forcing for the model.

3.2.1 In `/data_manip/`:

Here we have a selection of fortran programs for a wide variety of tasks. They are stand-alone without scripts to run them, so it's up to you to compile them:

```
gfortran -fdefault-real-8 -fconvert=big-endian
```

and link the right input files to the appropriate channels and to rename the output as required. This means you'll have to understand how the programs work, and possibly edit them to your needs. None of them are very long. They all have very specific purposes. Most of them work on spectral "history" files which may be reanalysis data or model output. Some work on grid data which is from model output of fields that are not spectrally analysed in the model, such as rainfall.

Here's the complete list:

- `add2_hst.f` - adds two model spectral history files together
- `checkhst.f` - simply reads a spectral history file to check the data
- `checkhst_modenorm.f` - reads a history file and calculates the norm used in "modefinder" (see Chapter 4, section 4a)

- `compare_hst.f` - compares two history files
- `concat.f` - concatenates two history files into one
- `daily_means.f` - calculates daily means from 4x-daily spectral data
- `extract_10yr.f` - extracts a chosen decade from the data sequence
- `extract_38x1Jan.f` - extracts a sequence of 1st of Januarys from the data sequence
- `extract_38x1st_of_Month.f` - same but for the first of any chosen month
- `extract_DJF.f` - to create a sequence of DJF 4x-daily data from the full sequence
- `extract_IC_for_ensemble.f` - to create a sequence of states by reading a long model run at fixed intervals to provide ensemble initial conditions
- `extract_last_40_days_same_IC.f` - from model output of any length, extracts the last 40 days of 4x-daily data and writes it out with the original initial condition - useful for analysing modes
- `extract_season_means_cyc.f` - to make four seasonal means from a mean annual cycle
- `extract_seasons_spec.f` - extracts the four seasons from the full sequential dataset - spectral
- `extract_seasons_grid2d.f` - same but for 2d grid fields (variables related to precipitation)
- `extract-label_years.f` - relabel spectral data with information about the year and daynumber
- `global_mean.f` - creates spectral data with the global mean of the input file but no horizontal variations
- `global_mean_resting.f` - same but with no wind at all
- `growth.f` - calculates growth rates in a quadratic vorticity norm (for analysis from the modefinder, see Chapter 4, section 4)
- `make_inst2.f` - extracts a single record from a sequence to furnish a singe initial condition
- `make_inst.f` - same with KOUNT and DAY set to zero
- `modify_metadata.f` - to update timing information on history files
- `pack_history_grid.f` - deals with a bug in writing out grid data where some of the data were not output from the model
- `rephase_MJO_fan48d.f` - shuffles MJO forcing anomaly sequence to change initial phase
- `subtract2_hst.f` - subtracts one history file from another
- `subtract_BS.f` - subtracts a basic state from a history file and adds a resting state in its place, useful for looking at anomalies on a fixed basic state.
- `subtract_CTL.f` - subtracts a time-dependent control experiment and adds a resting state, useful for tangent anomaly development.
- `subtract_IC.f` - subtracts the initial condition from a history file and adds a resting state, useful for anomaly forecasts.
- `TILS.f` - Time Independent Linear Solution. Complicated routine for extrapolating a set of three solutions found at different additional damping rates back to zero additional damping (see Appendix B,section 3).
- `time_filter_history.f` - creates a block mean history file for crude low-pass filtering.
- `time_mean.f` - creates a custom time-mean from the sequence data after skipping an initial period.
- `truncate_T42toT31.f` - creates T31 history record from T42 input
- `visualise_forcing_T42.f` - creates a history file that differs from the initial condition by an input forcing rate (from a `_fan`, `_fcm` or `_fbs` file) multiplied by one day to allow visualisation of tendency fields in units per day
- `W2G2W_sector_mean.f` - uses the model's spectral analysis code to do geographical manipulations on spectral data in grid space, notably the sector mean.
- `zonal_mean+WN123.f` - takes the zonal mean of the input data, with options to also make it symmetric about the equator, and to retain wavenumbers 1, 1 and 2 or 1,2 and 3.

3.2.2 In /prep_cyc/:

A small number of fortran programs specifically focussed on computing cycles. The names of these programs are quite self explanatory (except the ones that refer to components of the annual cycle CT, TT, MC and CC which are highly specialised):

- `annual_cycle_spec/grid` - calculates the mean annual cycle from a sequence - note that there is no smoothing filter applied in this code, that is done separately afterwards using cyclic_running_mean_spec/grid.
- `composite_n_day_cycle_spec/grid` - used for diagnosing aggregate cyclic responses to cyclic forcing (like the MJO) in long model runs.

3.2.3 In /prep_fan/:

Scripts and programs for preparing idealised forcing anomalies. First look at makefan.ksh. It compiles and runs makefan.f to produce a gridpoint forcing anomaly for either temperature or vorticity. This is then spectrally analysed at T42 using `specan`ANOMT42`.f` and written to a file called temp_fan.b (to be renamed as needed). T31 anomalies can be made by changing the parameters and using `specan`ANOMT31`.f` (or by downscaling the T42 result using `truncate_T42toT31.f`).

The program `makefan.f` starts from scratch and can be edited for the desired properties of the forcing anomaly. It will take the form of an ellipse with a cosine squared bell shaped horizontal distribution. You can specify its location and radius in x and y directions, its heating rate and its vertical profile. Examples are given in the file `Notes_for_forcing_anomalies.rtf`.

More complex shapes can be defined from gridpoint input using makefan_ReadGrid.f and sequences of forcing can be produced using `makefan_seq.f`. The model will read through a forcing anomaly sequence at a user-defined rate until it reaches the end and then it will repeat to give a cyclic forcing anomaly.

3.2.4 In /prep_lsm/:

Code for creating the land-sea mask in model format from gridpoint data

3.2.5 In /prep_seq/:

Basic code for creating the ERA-interim dataset as a sequential binary file in the model format.

3.2.6 In /prep_sst/:

Code for manipulating and visualising SST data and idealised SST anomalies:

- `check_grid_SST.f` reads binary SST data in model format and prints it on the screen to check it's OK.
- `makessta.f` creates an elliptical SST anomaly in model grid format in a similar way to makefan.f

3.3 source directory

In the /source directory you'll find the model: at time of writing it's dream_v8.4.f. Appendix A, section 6 and Appendix D take you through the code in some detail. It is easy to edit the code but not recommended ! If you do want to hack it for some special reason, just make sure you keep a safe original copy. The model calls some library routines in /lib but once compiled these should not be touched. It also reads a lot of parameters and common block variable declarations from the /include directory. Note that this is set up to work at two resolutions, T31 and T42, with the associated grid resolutions of 96 and 128 points around a latitude circle (and 24 and 32 latitudes per hemisphere). Switching between resolutions is transparent for the code, and to a large extent also for the associated data files. It's all set up in the job script, as described in the next section. So you have very little reason to visit this source directory.

Also in the include subdirectory is a setup file. This contains a few edits to the code to alter its behaviour depending on some choices made in the job script namelists. The idea is to have some standard use cases, but we haven't gone very far down this road as in general everyone's use case is different.

Finally there is a change-log which contains notes on changes made between versions of DREAM. As such it is a nice chronological summary of the development history which complements this user guide.

3.4 jobsdirectory

This is where you'll spend a lot of your time. We've already had a look at the job script to run the model in Chapter 2, and we'll go into much more detail in Chapter 4. Here's just an overview of the files in this directory.

- runmodel_v8.4.ksh - your basic script for running the model multirun.ksh - a bog standard script to sequentially run several experiments - edit at will.
- makefrc.ksh - the script for making forcing files _fbs and _fcm. How to do this is explained in Chapter 4 section 2. It calls fortran routines: calcfrc_T31.f or calcfrc_T42.f.
- makefrc_cyc.ksh - calculating a forcing function with an annual cycle is a protracted and elaborate business, see Chapter 4, Section 2d and Appendix B, section 7 . This script is only part of the procedure, along with fortran routines: calcfrc_cycADV.f and calcfrc_cycTEND.f.
- makefed.ksh - if you want to diagnose the transient eddy part of the forcing, it is the difference between a _fbs file and an _fcm file (see Appendix B). This script works it out using calcfed.f.
- run_ensemble.ksh - a very useful script for organising an ensemble forecast and then calculating an ensemble mean history record. It calls the script make_ensemble_mean.ksh, and fortran routines: ensemble_ic.f and ensemble_mean.f or ensemble_mean_dry.f.

3.5 results directory

So the model has run without crashing and you have some history files. They will have been put into your experimental directory under DREAM/dream_results. Here you will find for reference a record of how you set up the experiment in the form of the following files:

- runmodel_v8.4.ksh - a record of the script you used to run the model
- namelist_data - the changes you made to the model namelist defaults
- results - a text file with details of the run including namelist values etc.
- dream_v8.4.f and setup.f - a clone of the model code you used

- `parameters.f` - a record of your parameter (i.e. the resolution)
- `prog` - the executable file for the model
- `history` - the dynamical output from the model
- `history_grid2d` and `history_grid2d` - precipitation-related gridpoint output if appropriate `restart.11` and `restart.12` - restart files periodically during the run (11) and at the end (12).

When you run the diagnostics you'll also get:

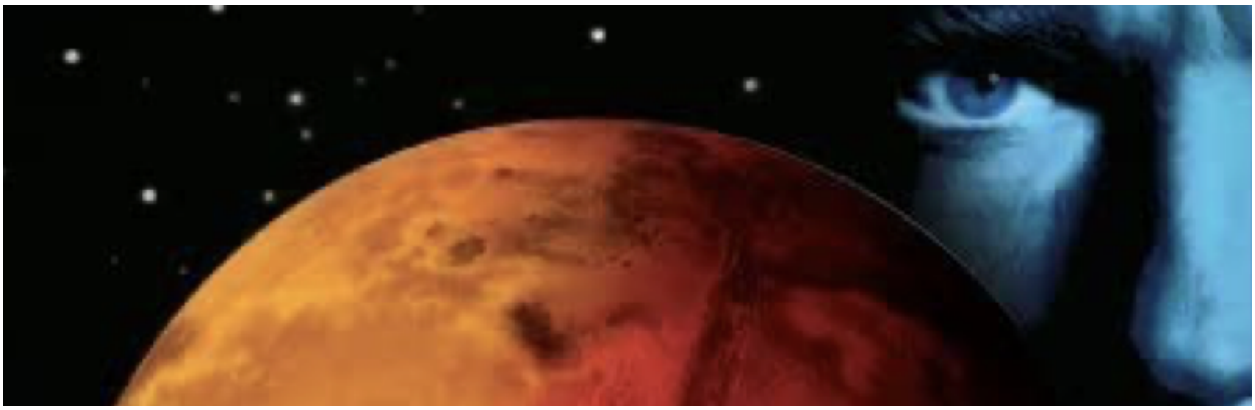
- `run_output.ksh` and `model_output.ksh` - a record of the diagnostic scripts you used `/netcdfs` - your netCDF output goes here
- `/binary_grid` and `/binary_grid3d` - binary diagnostic output if you selected it.

3.6 `diagnostics` directory

Another directory where you'll spend a lot of your time. Again, this is an overview of the files. A comprehensive guide to diagnostics is given in Chapter 4.

- `specan_W2G.f` - the core fortran program that everything depends on, this routine uses the spectral analysis from the model to take a model history file and write grid diagnostic fields for dynamical variables into binary and netCDF formats.
- `gridan2d.f` and `gridan3d.f` do a similar job but for model grid output, i.e. fields that are related to model precipitation.
- `time_mean_spec_grid.f` - calculates a time mean from your sequential output, called by the script if requested.
- `run_output.ksh` - this is your basic script for running the output, it calls `model_output.ksh` and `file_renamer.ksh` as well as the diagnostics programs cited above.
- `run_output_flux.ksh` - an extended script that does the same job as `run_output.ksh` but will also provide quadratic quantities with time-filtering, such as heat, moisture and momentum fluxes, geopotential variance and eddy kinetic energy. To do this it calls `model_output.ksh`, `model_flux.ksh`, `hp_flux.ksh`, and the fortran programs cited above. For handling fluxes and time filtering it also calls: `time_filter_history.f`, `time_mean_flux2d.f`, `gridan_flux2d.f` and `hp_flux2d.f`.
- `/include` - contains parameters and common blocks. Note that the common block files are not simply copies of the ones in the source directory.
- `check_grid3d.f` - this is just a verification program to print 3d grid data on the screen.

4 Chapter 4: Total Recall - The Many Different Ways of Using DREAM



4.1 Running DREAM

4.1.1 A tour of the job script

Let's look in more detail now at `runmodel_v8.4.ksh`.

First you choose your resolution with RESSP and RESGR. They should match, i.e. T31G96 or T42G128. In fact if you choose a different combination the model will still run, so if for example you want to degrade the resolution of the grid-point calculations compared to the spectral resolution, or vice versa, it is possible but an unlikely use case.

The next block of commands sets up the working directories and the fortran compiler and then we're into the namelists. A complete specification of namelist variables is given in Appendix C, and any one of them can be included here but the standard job script only contains the most commonly used.

- First we have SETUP, in which we specify the use cases RUNTYPE,THERMTYPE and SSTZONE. These are controlled in the include file `setup.f` mentioned above.
- RUNTYPE has five presents: TRAIN-to calculate forcing data, CYCLE and PERPETUAL for the type of forcing used, UNFORCED to run without forcing and CHANNEL which will execute a perpetual run with zonally uniform forcing (and optionally a zonally uniform initial condition, and there are further options for the type of symmetry or for adding long waves - but you'll need either to edit the setup file or set the namelist manually).
- THERMTYPE refers to moist thermodynamics. There are three options inspired by formula 1 racing: DRY, WET and INTER. DRY is just dry dynamics where the specific humidity variable is strictly a passive tracer with a source and sink in the forcing and so it has a reasonable climatology. WET has the model's large scale

rain and deep convection schemes switched on. The model will produce rainfall and output it into grid files. And the heating from condensation will feed back onto the dynamics. INTER runs with dry dynamics but the rainfall schemes are switched on. So the model will produce rainfall and the humidity variable will not be just a passive tracer. But the feedback onto the heating is cut, so the dynamical development is the same is in a DRY run.

- SSTZONE works with the SST forcing input and restricts it to a pre-defined zone, always in the tropics - either the entire tropical band or one of the three ocean basins.
- Also in SETUP we have the basic run length KRUN. This is in timesteps and there are 64 timesteps per day. So if you want to run for 10 days, set KRUN=640. But what if you want to run for the entire month of January, with the initial condition at 00Z on 1st Jan and the final record at 18Z on 31st Jan ? That's 31 days right ? Well, no not really. If you set KRUN=64x31=1984 and you use the default value of KOUNTH=16 (the output frequency in timesteps for 4x daily output) then your resulting history file will have 125 records, the first of which is your initial condition and the last one will be 00Z on 1st Feb. If you want to finish in Jan, you'll need to set up the run to deliver 124 records, which means setting KRUN=1984-16=1968. It's up to you of course, you do whatever you want. Are you more of an Orwellian bent, or more in the spirit of peace and love ?
- Finally in SETUP we have the important variable KTFIN. This is only needed when using the model to generate a forcing file (RUNTYPE=TRAIN). Set it to 1 to make a single-shot _fbs file. Set it to something else to scan through a longer set of initial conditions: values are given in the comments.
- The other namelist is INITIAL and there are many choices here. Whatever you put here explicitly will override the defaults, and it will also override the use case scenarios provided in the setup file. So you can tinker here to your heart's content. KOUNTH and KOUNTFAN are intervals for writing history records (channel 9) and reading spectral forcing anomaly data (channel 15) respectively. A multi-record forcing anomaly file will be read sequentially until the end and then rewound and read gain.
- KBEGYRSST and KBEGMNSST are the year and month at which the model will start reading SST data when it is using SST data with metadata - for use in seasonal forecasts/hindcasts. This date will also be assigned to model output.
- LSST controls whether the model reads SST data and the following switches determine what it does with it, including the type if transfer function to atmospheric heating and some timing options. Note that SSTs can heat the atmosphere even in DRY or INTER runs.
- LFAN controls the reading and implementation of a spectral forcing anomaly, with scaling and timing options.
- LNUDGE activates nudging.
- LSTAB activates a uniform damping on all degrees of freedom, for use in stability calculations and derivation of time independent solutions when the basic state is unstable.
- LMODE activates the modefinder, which turns the model into a crude eigensolver to get the fastest growing mode for a given basic state by timestepping.

After this we are into linking input and output files. We've already covered the input files in Chapter 2, and the rest is fairly self explanatory. The main file you will work with is the history file which is a binary output with sequential records describing the model state as spectral coefficients of the state variables. For diagnostic fields related to precipitation the model writes history_grid2d and history_grid3d as needed. These outputs are controlled by namelist options. Restart files are written periodically (restart.11) and at the end of the run (restart.12). These files contain two consecutive timesteps so in principle can be used to restart the model and continue the run. Some basic information about the run is written into the text file results.

That's the basic overview - let's look at some specific examples.

4.1.2 Running DREAM as a simple GCM in perpetual mode

This is pretty straightforward. You want to simulate an equilibrium climate. You'll need to decide how long you want to run it. Bear in mind that the model can clear about 50-days per minute at T31 and about half that at T42 - your mileage may vary ! I suggest doing a few short test runs before you plunge into your long reference runs. Run it for a few hundred days and do some diagnostics. You probably want to have a control simulation, and the some sort of experiment in which you perturb something. You'll need to decide what perpetual season you want to simulate.

So first set your resolution and set RUNTYPE="PERPETUAL". For a 100-day run you'll set KRUN=6400. Is this a dry run or do you want explicit condensation processes ? Do you want them to affect the dynamics ? Set THERMTYPE accordingly. If you choose a WET or INTER run the model will automatically write `history_grid2d` files for the precipitation.

The rest is just a matter of choosing files. Your initial condition would normally be an instantaneous field. Your forcing and reference files must agree: for example `ERAi_ave4x_1979-2016_DJF_fcm.b` and `ERAi_ave4x_1979-2016_DJF.b` in channels 13 and 16.

When you come to do diagnostics you should skip a set number of spinup days and look at the mean (and maybe fluxes) over the remainder of the run. A spinup of about a month is usually sufficient. For statistically significant results for a climate signal you'll have to experiment. The model can display some very low frequency variability so be prepared to do long runs. The length of the runs you'll need to do depends entirely on how strong the perturbation you are testing is, and how strongly the model dynamics responds to it.

4.1.3 Running DREAM as a simple GCM with an annual cycle

This is again a GCM-style experiment so again we're talking control and perturbation. But now you have an annual cycle, you'll have to do a bit more work on the diagnostics, extracting seasonal means etc. And inevitably you'll have four seasons to consider.

Technically all you have to do is set RUNTYPE="CYCLE" and choose your initial condition, forcing and reference states accordingly. The model will normally start at the beginning of the calendar year so your initial condition should be a first of January or at least some sort of boreal winter state, and you should consider a spinup. The forcing will be `ERAi_cyc4x_1979-2016_fcm_RM41.b` and the basic state will now also be an annual cycle sequence file `ERAi_cyc4x_1979-2016_RM41.b`. Both these files will step forward once every 6 hours, or 16 timesteps and will return at the end of a 365.25-day model year.

It is possible to start the annual cycle at the beginning of any month by setting KBEGMNSST to the desired month between 1 and 12. The model will skip through the forcing and reference data until it gets to the right part of the annual cycle and start from there. And you don't need to actually read SSTs or enable SST forcing in order to play this trick. If you do this the history records will have the correct time signature and your ensuing netCDFs will flag the correct calendar date.

4.1.4 Running DREAM as an ensemble forecast model

Ensemble forecast runs can be made in any configuration: perpetual, cycle or even idealised stationary wave runs with a fixed basic state. All that is required is to use a script that runs the model multiple times and then takes an average of the output. Such a scrip is available and it is called `run_ensemble.ksh`.

The comments section at the beginning of this script is extremely comprehensive so just follow it. Basically you have to decide how many members your ensemble has, and how to initialise each member. The script assumes that all your initial conditions are gathered in a single file, one per record. It can read the initial conditions sequentially, or it can skip records at a fixed interval. This is useful if you're using a previous long run to supply initial conditions. Some options for the initial condition file are already provided in /seq directory but you can also make your own. The script will copy each initial condition to a file called initial_condition which is overwritten as you run each member of the ensemble.

A subdirectory is created in your experimental directory for the history records from each ensemble member. Each will have a run number appended to the file name: `history_n`. When all the members have run, the ensemble mean history is calculated and deposited up in your experimental directory. So note that the file in your experimental directory at the end that is called "history" is not a direct product of the model, it is the ensemble mean. This can also all be done for `grid2d` files if you have rainfall but you may need to uncomment those lines from the script.

4.1.5 Running DREAM as a perturbation model with a fixed basic state

This is the story of a dynamical model with a climatological initial condition, and of the one forcing that will cancel all its tendencies. It's Arrested Development.

You should try this first: set RUNTYPE="PERPETUAL" and THERMTYPE="DRY" and then initialise with the climatology `ERAi_ave4x_1979-2016_DJF.b` in channel 10 and also as a reference state in channel 16.

Then use: `$DATADIRSP/fbs/ERAi_ave4x_1979-2016_DJF_fbs.b` as the forcing in channel 13.

Make sure LSST and LFAN are both set to `.F.` Run for about 10 days and you'll get the idea. There should be no development. The model reports its 100th spectral coefficient of vorticity every time it outputs history, and you'll see that number scroll down the screen, unchanging, except on the last couple of decimal places at machine precision. This tiny development is due to rounding errors and if your basic state is unstable, which it will be if it's a mean winter state, then this will grow like Lorenz' butterflies until it reaches finite amplitude, usually a few hundred days into the run.

In the meantime you've got plenty of time to do perturbation experiments. There are two ways to perturb the run so that it develops an anomaly. You can change the initial condition (more on that in section 4i), or you can add a perturbation to the forcing. Here we will concentrate on the latter.

Forcing perturbations can be introduced either through a direct perturbation in spectral space, setting LFAN=`.T.` and reading a `_fan` file into channel 15, or through an SST anomaly, reading SST data into channels 17 and 18.

Try repeating the run you just did, and adding one of the forcing anomalies in `dream_data/T31/fan`. A deep heating in the tropics will set off a classical chain of events: tropical Kelvin and Rossby waves going east and west, then a Rossby wave propagating out into the extratropics and finally baroclinic waves on the extratropical jets that propagate eastwards and grow exponentially. You can diagnose all these effects by looking at the difference between the solution and the initial condition / basic state.

You can also control the amplitude of the perturbation directly from the job script using SCALEFAN. Double the forcing anomaly by setting SCALEFAN=2. Change its sign of by setting SCALEFAN=-1. If you set SCALEFAN to a very small number, say 0.0001, the model response will be correspondingly small and will develop as a linear perturbation around the basic state. You can scale it back up for presentation in the diagnostics.

By default the forcing anomaly is fixed and persistent. As already discussed you can make it time-dependent by using a sequential file, leading to a cyclic forcing perturbation. More simply, you can also create a short-lived initial pulse of forcing anomaly with LPULSE, using KPULSE to determine the length of time in timesteps over which the initial pulse grows and decays symmetrically in time following a sin-squared curve. The integrated amplitude is normalised so that it is equivalent to having a fixed amplitude pulse over the same time interval.

If you wish to perturb a fixed basic state with an SST anomaly instead there are various considerations about how the model responds to SST data that are covered in section 4iv, but the basic principles are the same, and there are also options for scaling the perturbation in this case.

To diagnose perturbation runs it's best to subtract the climatology from the model state, and this can be done in the diagnostics package, furnishing netCDFs that look like anomalies on a resting state. Of course you can also do this independently of the code presented here by using your own software to compare netCDFs generated by the perturbation run with a netCDF of the basic state. As always, it's up to you.

4.2 Calculating the forcing

The theory of how a set of one-timestep tendencies can represent either diabatic and transient eddy forcing is laid out in Appendix B. Simply put, the `_fcm` forcing represents just the diabatic terms that are normally parameterised in a GCM. The `_fbs` forcing represents the sum of this diabatic forcing and the systematic effect of transient flux convergence, often termed "transient eddy forcing". The latter is simpler to calculate so we'll start with that.

4.2.1 Forcing a perturbation model with a fixed basic state

Suppose you have a basic state that you want to use for perturbation experiments. You want the model to develop as a response to your perturbation, in conjunction with this basic state. The first thing you need is a forcing that will stop the basic state from evolving.

To find this forcing, you take your basic state, use it as an initial condition (channel 10) and also as a reference (channel 16) and run the model with no forcing at all, for just one timestep. To achieve this, all you have to do is set RUNTYPE="TRAIN" and KTFIN=1, and make sure all your perturbations are switched off (LFAN=.F. and LSST=.F.). This setup will impose dry dynamics, switch off the basic forcing and run the model for one timestep.

When your run is complete (it happens very quickly) you need to use the information from the initial condition and the result after one timestep. These are both in your experimental directory as history and restart.12. In the job directory the script `makefrc.ksh` consults these two files to find the tendency and thence the forcing. Remember to set KTFIN=1 in this script in the namelist `ktfin_input_data` and set the call to the `calcfrc.f` fortran program to the correct resolution. Then all you have to do is make sure your output is renamed to the appropriate file, which should finish with `_fbs.b`. This can be in your experimental directory for specific applications or if you want to keep it across multiple experiments put it in dream_data in the appropriate `/fbs` directory.

To check that it worked, try running the model in PERPETUAL mode (forcing will be switched back on and the model will run to whatever you have set as KRUN). You should get zero development as described above (be careful interpreting the `Z(100)` output on the screen for very simple basic states - it is likely to be zero so you'll get some very small numbers that dance around at machine precision).

Remember - every time you change anything in the model, damping parameters for example, you will need to recalculate the forcing.

4.2.2 Forcing a simple GCM in perpetual mode

The forcing for a simple GCM is fundamentally different because it represents only the diabatic forcing and not the transient eddy part. In a fully turbulent GCM simulation the transient eddies are explicit and so their mean fluxes are represented explicitly in the simulation, they are not present in the forcing. But to calculate this `_fcm` forcing we follow exactly the same procedure as for the _fbs forcing, and use the same code.

The only thing that differs is the value of KTFIN. Now we have to scroll through a long list of initial conditions, running for one timestep from each of them. In principle it doesn't matter what order the initial conditions come in, but in practice it is usually a time sequence.

There is a big loop in the model that scrolls through this sequence when training mode is activated. The initial condition should therefore be a `_seq` file for whatever season or composite you are interested in. The reference data should be set appropriately to the seasonal mean. And KTFIN should be set to the number of records in the sequence (some values are given in the job script for the data provided). Don't forget to set KTFIN to the same value in the job script and in the makefrc script. After looping for a while, first to run the model and then to find the average tendency, you'll get a `_fcm.b` file - make sure you put it in the right place, i.e. in a `/fcm` directory.

There is no simple way to check that it worked this time. You'll just have to use it as a forcing for a long run of the simple GCM, with whatever initial condition you want, and a suitable spinup period, and then validate.

4.2.3 An aside on reference data

Every time you do an operation like this you need to use the appropriate reference climatology in channel 16. This seems like a pain, and to be honest it is, and it's not strictly necessary if all the dissipation in the model is linear. This is just the way it evolved. In principle, any reference file could be substituted in and give the same result, because the forcing that you calculate would simply redress the large tendency associated with whatever reference state you use. DREAM could therefore be configured to avoid this reference state altogether and have one less input file to worry about. But that's not the way it has been done, and in fact the reference state does play a role in some very nonlinear bits of the model, especially concerning deep convection and the response to SSTs.

4.2.4 Forcing a simple GCM with an annual cycle

The procedure for finding the annual cycle forcing is not so straightforward, and the theory is outlined in Appendix B. The forcing consists of an advective part, calculated with the model, and a tendency part, calculated directly from the data. To calculate the advective part the procedure for running the model is the same as above with a very long sequence of consecutive years. But the time averaging is now cyclic in nature so the resulting forcing file is a sequence. The tendencies are calculated in the script makefrc_cyc.ksh but the cyclic averaging must be done manually using the code in data_process/prep_cyc (see Chapter 3, Section 2). The full procedure to calculate the annual cycle forcing is:

First step:

- Calculate the annual cycle of the data: `annual_cycle_spec.f`
- Smooth it: `cyclic_running_mean_spec.f`
- Use it to calculate the tendency part: `calcfrc_cycTEND.f`

Second step:

- Calculate the advective part using the model: `makefrc_cyc.ksh` and `calcfrc_cycADV.f`
- Calculate its annual cycle: `annual_cycle_spec.f`
- Smooth it: `cyclic_running_mean_spec.f`

Final step:

- Add the two cycles together (the tendency and advective parts): `add2_hst.f`
- Smooth it all again: `cyclic_running_mean_spec.f`

4.2.5 How to calculate forcing perturbations

There are many ways to perturb the model. Let's start with a simple idealised heating perturbation. In `data_process/prep_fan` open the script `makefan.ksh`. This script will help you to make a spectral forcing perturbation file `_fan.b` by specifying the shape and amplitude of your forcing in grid space.

To do this you will need to edit the fortran program `makefan.f`. This program is set up to generate idealised perturbations either to temperature T or vorticity Z. The first thing it does is set up the amplitude. Then you decide on your vertical profile. There are many to choose from (see `notes_for_forcing_anomalies.rtf`) or you can invent your own. Make sure it integrates to unity between sigma=0,1. Then you specify four coordinates: the longitude and latitude of the centre of the anomaly, and its radius in longitude and latitude directions. The program will fill this ellipse with a cosine-squared bell function and put zeros outside it. Note that makefan.f works on a T42 Gaussian grid but the output can still be used to make T31 forcing anomalies.

Back to the script and the output from makefan.f is fed into a spectral analysis routine `specanANOMT42.f` (or T31 if you want). Use the resulting `temp_fan.b` file in the model (channel 15) and see what it does. You can have no end of fun.

If you want a time-dependent forcing perturbation then you can create a time sequence with `makefan_seq.f`, which will allow your idealised source to move. It is set up to translate uniformly from a beginning position to an end position. It can also grow, shrink or change shape as it goes, but for the moment the vertical profile is fixed.

Sequences of forcing anomaly data are read by the model at a frequency determined by the namelist variable KOUNTFAN until it reaches the end of the sequence, then it will rewind and repeat. If you want to be a bit less idealised, then you can read the forcing anomaly from grid data using `makefan_ReadGrid.f`. These programs should serve as an example of how it's done and form this basis you can develop your forcing anomalies as you please.

4.3 Diagnosing the output

4.3.1 Time-mean diagnostics

So you've run the model and successfully produced a history file with all the model dynamics. And if you ran the wet version you might have `history_grid2d` and `3d` with rainfall and diabatic heating as well. Well done, you're about half way there. It remains to convert this information into useable form and then visualise it.

There are a few basic ways you'll want to visualise the output. The diagnostics package is very focussed on horizontal maps on selected model levels. These can be written straight to netCDF, and there is a fairly comprehensive set of diagnosed variables. For vertical sections you'll have to delve into the binary 3d output and do it yourself. And then there are a few basic arithmetic operations you'll want to do: time sequences and time averages; anomalies from a reference state, and possibly time-mean second order transient fields, with various time-filters. This is all possible in the diagnostics package. Needless to say this is all in the `dream_model/diagnostics` directory.

Let's start with the simplest option: a time sequence and/or time mean of the basic dynamical diagnostics on selected model levels. Have a look at `run_output.ksh`. After choosing your resolution you then select the records to be included in your time mean: you skip nspinup records and then average over nmean records. If nmean is greater than the number of remaining records the average will be just over the remainder of the sequence (note that these two parameters only affect the time mean, the sequence output will go through the whole history record regardless). If you want to take the mean of the whole sequence just set them to `nspinup=0` and `nmean = something big`.

After this we have the usual directory allocation as in the model job script. Then there are various choices on what kind of output you want and where to send it.

`HISTID` and `FILTER` are strings appended to history files and output directories to distinguish them from one another. For example it might be useful to set `HISTID="_DJF"` and `FILTER="_mm"` or `"_hp"` if you're dealing with data from various seasons in the same directory, and you want to treat monthly mean or high pass components. More on that later.

REFSP and REFGR are reference states for spectral and 2d grid point data that you might want to subtract from your results to display an anomaly. To do so set lsubtractsp and lsubtractgr to true. There is also an option ltangent which will enable a time dependent anomaly calculation. In this case your reference files must also contain sequences of data to be subtracted from your sequence of results.

You can choose to calculate a time mean and/or a sequence by setting lmean and lseq. If you have 2d or 3d grid data from the run and you want output from it set `lreadgrid2d` and `lreadgrid3d` accordingly. Remember the 2d grid comprises fields related to precipitation and the 3d grid data is just the associated diabatic heating.

`run_output.ksh` is a high level script and most of the action takes place one level down in the script it calls: `model_output.ksh`, let's have a look. After a bit of file handling you see that this script is split into two sections: the time mean output and the sequential output.

For the time mean output there are two namelists. The first passes information that we've already set up about the averaging period to the fortran code. The second namelist is used to for output options. The levels for output are chosen with LEVOUT. Set it to true for each level required: the list of levels is given in the comment. This namelist also picks what kind of binary grid output you want, if any. `LBINDYN` and `LBINPPT` will send basic dynamical fields and precipitation information respectively to binary output. If you want full 3d information rather than just

level by level then activate LBIN3D. Note that it is assumed that netCDFs are always wanted on the levels chosen so there isn't a way to switch that off !

The script then calls the fortran program `time_mean_spec_grid.f` to do the averaging operation on the spectral and grid history files. This produces single-record average history files: `history_ave` and `history_grid2d_ave`, in your experimental directory. These history files are then fed to the main analysis programs: `specan_W2G.f`, and if needed `gridan2d.f` and `gridan3d.f` (details of the binary file structures are given in Appendix A).

`specan_W2G.f` calls the spectral analysis routines from the model and then does some analysis in grid space to produce a number of diagnostics which it sends to netCDF and binary grid output. The fields it produces are labelled as follows:

- chi` - velocity potential - m2/s
- `div` - divergence - s-1
- `gph` - geopotential height - m
- `omega` - pressure vertical velocity - mb/hour
- `psi` - streamfunction - m2/s
- `q` - specific humidity - kg/kg
- `sp` - surface pressure - mb
- `T` - temperature - deg C
- `u` - zonal wind - m/s
- `v` - meridional wind - m/s
- `vort` - vorticity - s-1.

Apart from geopotential height and vertical velocity, which are calculated in the diagnostics program, these variables are all direct from the model's state variables.

The gridan programs are more limited in scope and only concern grid output from model runs with moist thermodynamics enabled.

- `ppt` - precipitation rate - mm/day
- `vimc` - vertically integrated moisture convergence - mm/day
- `vicw` - vertically integrated column water - mm
- `visf` - vertically integrated saturated fraction - nondimensional
- `pptdeep` - precipitation rate from the deep convection scheme - mm/day
- `condheat` - condensation heating (3d variable) - degrees/day.

4.3.2 Sequence diagnostics

The second half of model_output.ksh repeats these operations for the sequential model output. So it will produce netCDFs with a time dimension. There is nothing new to say about this, it is just a repeat of the calls used for the time mean. The only thing worth mentioning is that the namelist input is written again here - so you have the opportunity to make different choices for time mean and sequential output. You could, for example use different levels for time mean and sequential data, or plot full fields for the time-mean and anomalies for the sequence, or request binary output only for the time-mean, or whatever combination you desire. All these choices are for the moment in this lower level script as I expect them to be fairly standardised, and thus choose not to clutter the top level scripts.

4.3.3 Time-filtered transient fluxes, variance and eddy kinetic energy

It remains to discuss time filtering and second order products. This is a fairly recent development and it is all dealt with from the script `run_output_flux.ksh`. This starts off the same way as run_output.ksh to do the basic mean and sequence diagnostics. Then the script deals with filtering and fluxes. The sequence of operations is as follows:

1. Run model output on history file, skipping spinup, with sequence and mean output. A copy of run_output.ksh so you only need to run this one script. Note that lbinflux is set to true, so further sequence and mean diagnostics are output to binary grid. These extra files include eddy kinetic energy, momentum flux and zonal and meridional fluxes of temperature and specific humidity, as well as geopotential height and vertical velocity.
2. Create low pass and monthly mean history records. Calls `time_filter_history.f` to produce sequential history files with block mean values. So each record of the resulting history files is a contiguous mean of NBLEN records from the original history file. If this block length NBLEN=`12` for a low pass filter, this corresponds to consecutive 3-day means. Diagnostics from this sequence will be like from the original sequence with a crude 6-day low pass filter applied. If NBLEN=`120` then the sequence will contain consecutive monthly means. These reduced block-mean history sequences are labelled `_lp` and `_mm` respectively.
3. Run model output for low pass and monthly mean, for sequence output only. Low pass and monthly mean sequence diagnostics are produced for further calculations.
4. Run model flux level by level for total flux, low pass and monthly mean. The script `model_flux.ksh` takes the time mean of the sequences just produced, squaring the geopotential height and vertical velocity as it goes to output mean standard deviations of these quantities. These mean quantities are written to netCDF providing the following diagnostics:

- `gph` - standard deviation of geopotential height - m
- `omega` - standard deviation of vertical velocity - mb/hour
- `eke` - eddy kinetic energy - m2/s2
- `uv` - momentum flux - m2/s2
- uT - zonal temperature flux - deg C m/s
- vT - meridional temperature flux - deg C m/s
- uQ - zonal specific humidity flux - kg/kg m/s
- vQ - meridional specific humidity flux - kg/kg m/s.

This is done level by level for the total unfiltered quantities, low pass and monthly means. The three levels chosen are 250, 500 and 850.

5. Calculate high pass fluxes level by level. It remains to subtract the low pass quantities from the total to give the high pass component.
6. Run model flux level by level for high pass netCDF output. And output netCDFs of that.

Finally, if you do decide to use the filtered-transient package, bear in mind that it is relatively untested so pay attention to all the switches and directory names along the way. You'll probably have to be quite hands-on and it would be best if you had a good look at the code rather than using it as a black box.

4.4 Going further

4.4.1 Diagnosis of normal modes and time-independent solutions

Linear analysis around a fixed basic state is one of the things that DREAM has been used for quite a bit. For mathematical details see Appendix B, section 3. The trick of fixing the basic state in a time-stepping model allows you to track the response to a perturbation in the forcing for a limited time, usually about 20 days. After this, the instability of the basic sate starts to manifest, in the form of exponentially growing modes that usually look like midlatitude wavetrains. They can be a nuisance, because they are pretty independent of whatever carefully designed perturbation you are trying to study. On the other hand they can be interesting in their own right as a fundamental property of the basic state.

DREAM has provisions for addressing both these concerns. So if you want to study the structure and growth rate of the fastest growing normal mode on a given basic state, you can enable the modefinder. Choose your basic state. Set the forcing to maintain it as explained above. Then set `LMODE=.T.` in the namelist. Oh and make sure you initialise with a state that has enough multi-scale information to break the symmetry of whatever basic state you're using, especially if it is zonally uniform, because the modes you're looking for won't be zonally uniform and they have to grow from something.

The model will call the subroutine SCALEDOWN, which compares a mid-level mean mean-squared vorticity norm with a standard value. For a growing perturbation on an unstable basic state, every time the ratio creeps up above 10, the difference between the model state and the basic state will be scaled back down. This is also done to the tendency so the model will continue seamlessly. To start with, this perturbation will be a mess, because it will contain many different modes. But you'll need to run it for a good long time, and as the run progresses, the fastest growing normal mode will eventually dominate, and the signal will be a pure shape-preserving mode that cycles sinusoidally between phases and grows exponentially. You can analyse it using the simple program `data_process/data_main/growth.f` (remember the norm is quadratic).

If your basic state is stable, then you'll have no long term growing modes but don't worry. Instead of scaling down, the model will scale up, and you'll find the slowest decaying mode.

The stability or otherwise of the basic state depends on both its three-dimensional structure and on the dissipation parameters chosen in the model run. If instability is an undesirable property, then we can do our best to eliminate it but adding extra damping to the model. If you set `LSTAB=.T.` some extra linear damping will be added equally to every degree of freedom of the model, thus preserving the modal structure of the basic state. The damping rate is set in days with TAUSTAB. If you've already diagnosed the growth rate of the fastest growing mode of your basic state with the modefinder, then you know what value of TAUSTAB you need to use to kill it. But remember, every time you mess with the model, in this case by altering its dissipation, you will need to recalculate the `_fbs` file that maintains the basic state.

What if you're interested in the time-independent response to a given forcing anomaly but your basic state is unstable ? Time stepping the model will not converge to a fixed anomaly solution, but rather degenerate into instability and chaos. But if you stabilise your basic state you'll find that you can then repeat your perturbation experiment and a long run will converge to a time independent state. You don't get something for nothing though, because the extra damping will considerably weaken the response so it won't be the same as the imagined time-independent response with standard damping parameters. A crude fix to this is to use various values of TAUSTAB and then extrapolate back to zero extra damping. The fortran program `data_process/data_main/TILS.f` will calculate the solution using a quadratic extrapolation with three values of TAUSTAB (note that the extrapolation is actually done in terms of damping rate, not timescale). `TILS` stands for Time Independent Linear Solution. It will only be linear if by construction your perturbation is small. Otherwise it's nonlinear but it will still work.

4.4.2 Nudging

Set LNUDGE=.T. to activate the nudging option. This is a linear relaxation towards a prescribed state that takes place in grid space. This makes it possible to nudge specific regions whilst leaving the solution unconstrained over the rest of the globe. It's useful to evaluate the remote impact of some observed sequence of events.

The nudging state is separate from the rest of the input data and so can be different to the model's reference state. It can be a single-record fixed state or a sequence. The model will read the next record from this file periodically, at a frequency that can be set to KOUNTNUDGE timesteps (default 16, i.e. 6 hours). If it reaches the end of the file it will rewind and start again. Each time the nudge state is read it is transformed to grid space and the variables u,v,T,q at all levels and sp are relaxed towards this state on a timescale specified by TAUNUDGE (default 6 hours).

The region in which nudging occurs is defined by the grid coordinates IWNDG, IENDG for the western and eastern boundaries and JNNDG,JSNDG for the northern and southern boundaries which define the mask RMASKNDG (see Appendix A for details of the model grid). The strength of the nudging is halved on the grid boundaries for a smooth transition to un-nudged regions.

4.4.3 Moisture, condensation and convection

When you choose the setup option THERMTYPE="WET" you are activating schemes that will condense water vapour to produce precipitation and heating. The conditions under which this happens, and the amount and distribution of precipitation and heat requires some explanation, as the user has a great deal of control over these routines. The three essential switches that are thrown to positive are LCHX, LLSR and LDEEP. Let's take them in reverse order.

Deep Convection:

LDEEP activates the model's deep convection scheme. This is a semi-empirical home-grown scheme that first decides for a given grid point whether deep convection takes place, then calculates the rate of precipitation produced, and accordingly modifies the tendencies of specific humidity and temperature. Before it calls the deep convection scheme the model does a quick check on how much water there is, by calling a routine called PWATER. For a given grid point this routine calculates the amount of water in the column in mm, and the amount of water there would be in the column if it were saturated. If the option LTRUNCQ is enabled (by default it is not) these calculations are done with a smoothed humidity field. PWATER also calculates the vertically integrated moisture flux convergence (mm/day) at that instant in the model and also for the model reference state.

Then the model calls its convection scheme DEEPTEND. The first task is to decide whether to do anything. Convection is triggered only if the following criteria are met:

The column integrated moisture flux is convergent. The moisture flux convergence is stronger than the reference value by more than a certain threshold (the default for this threshold is zero so in fact it's a simple comparison with the reference state). In fact this criterion is evaluated with a smoothed version of the moisture flux convergence when LTRUNC is enabled (by default it is). The flux convergence is spectrally truncated to an equivalent wavenumber T15 (this can be modified by setting NNTRUNC). So the model convection scheme sees the large scale moisture flux convergence, not the fine details. The low level temperature difference between the bottom two layers and the next two layers up exceeds the reference value (in the sense of the boundary layer being less statically stable than the reference state).

The third criterion is optional and is only activated when LBLSI=.T. (by default it is false).

If deep convection is triggered then the amount by which the tendencies need to be adjusted is calculated with reference to a condensation time scale TAUCOND (default value 90 minutes). The amount of precipitation that will fall within this period is set equal to the moisture flux convergence multiplied by this time period. So in fact the value of TAUCOND does not affect the deep convective rain rate. But there are a few modifications to this simple rule. First, it is the smoothed flux convergence that is used in this calculation. Second, we set limits to how much water can fall in the period TAUCOND. It can't exceed the total column water. And it can't exceed an imposed threshold

rain rate PPTCAP (mm/day), but is only permitted to approach it asymptotically. The default value for PPTCAP is 15 mm/day but it is of course a user tuneable parameter.

The reason for imposing all these controls on the amount of condensation associated with the divergent flow is that we need to tackle the beast known as CISK (Convective Instability o fthe Second Kind). CISK plagues simple models with condensation heating schemes that are linked to the divergent flow. It can lead to runaway grid-point instability and this has been observed in DREAM under certain parameter regimes.

All that remains is to calculate the actual tendencies for the specific humidity and the temperature. This is tackled in the subroutine PROFILEHEAT. Again if the option LTRUNCQ is enabled (by default it is not) these calculations are done with a smoothed humidity field. The rate of loss of water in the column is distributed proportionally to the humidity content. The associated diabatic heating is distributed into a deep convective profile, meaning there is an implied unresolved vertical moisture transport in the column. By default the heating profile peaks at sigma=0.35. But you guessed it, the peak level of the heating is a user definable parameter, set with PRHEATMAX in the namelist.

Large Scale Rain:

Once the deep convection scheme as done its job, whatever super-saturation is left gets mopped up by the large scale rain scheme, activated by the option LLSR. The subroutine LSRTEND calculates the difference between the the specific humidity and its value at saturation. If the option LTRUNCQ is enabled (by default it is not) these calculations are done with a smoothed humidity field. If the specific humidity exceeds the saturation value then this difference is condensed in-situ, for every grid location at every level. The timescale over which the condensation is to take place is again given by TAUCOND (note that for large scale rain the parameter TAUCOND does influence the rain rate). Just as with the deep convection scheme, there is some protection here against runaway CISK. The maximum in-situ condensation is limited to a value QDIFFCAP, which is currently hard wired in the code to a value that limits the large scale rain rate to 50 mm/day for the default value of TAUCOND. The appropriate negative tendency is applied to the specific humidity, and the associated diabatic heating is applied in-situ to the temperature tendency.

Forcing Modification:

All these calculations for tendencies of humidity and temperature based on parameterisations of physical processes are bound to have a non-zero time-mean effect. There will be a net sink of moisture and a net source of heat. This is clearly in opposition to the DREAM philosophy of deriving climatological sources and sinks of state variables from data, using residual tendencies from the unforced dry model. We are effectively double counting, and so some correction must be made for this in the basic forcing. In fact it turns out to be crucial to obtaining realistic simulations. The rain schemes don't have much work to do if realistic time-mean sources and sinks are already provided on the right hand sides of the equations.

One objective way to proceed is to hand over all the condensation to the explicit schemes, while continuing to provide the evaporative sources. This is what happens when LCHX is enabled. The forcing for specific humidity is retained where it is positive, but set to zero wherever it is negative. When LCHX is enabled, the humidity variable quickly develops to supersaturation, and the explicit condensation processes kick in.

There is also a modification to the temperature forcing to account for exactly the amount of condensation that has been removed from the humidity forcing. Note that this correction to the temperature forcing depends on the value of the latent heat of condensation. This can be modified by the user through the factor SCALELHEAT. In particular, if THERMTYPE is set to INTER, then SCALELHEAT=0 and we recover dry dynamics with working rainfall schemes. The moist thermodynamics is present, but completely decoupled from the dynamical solution. In this case only the specific humidity forcing is altered, not the temperature forcing. But whatever the value chosen for SCALELHEAT, be it zero, unity something in between, or even greater, the model will incorporate it into the LCHX correction made to the temperature forcing.

This pragmatic forcing strategy has lead to some reasonably realistic rainfall climatologies, and so forms the basis of a hybrid dynamical-physical-empirical general circulation model. The ability to simulate rainfall opens up a lot of applications that were not possible for a simple dry GCM.

Adding sea surface temperature effects

How the atmosphere reacts to an anomaly in the sea surface temperature is the subject of major importance and a vast literature. Because of the coupled nature of surface fluxes, convection and radiation it is difficult to derive empirical rules. But this doesn't stop us from trying. We restrict ourselves to the tropics where we can more safely argue that the ocean forces the atmosphere, and to time scales long enough to hope we can apply empirical rules. The causal logic is that the SSTA leads to an anomaly in precipitation which in turn is translated into diabatic heating and fed into the model. Much of the architecture of the model's convection scheme can be co-opted into this problem to take an anomalous precipitation rate and feed it into a deep convective heating profile. This is dealt with in the subroutine SSTTEND.

Set LSST=.T. and the model will expect to read two gridded SST datasets. Into channel 18 it will read a long term climatology, either fixed or developing. The default assumption is that it changes once per month but this can be modified with KOUNTSSTC. The SST data read into channel 17 can be either a realistic weekly value (this period can be modified with KOUNTSST) or an anomaly. Either way, the model is interested in the SST anomaly. How the model interprets channel 17 depends on the value of ISSTREAD. If ISSTREAD=1, then the model treats channel 17 as SST data and calculates the anomaly as the difference between this and the climatology (channel 18). If ISSTREAD=2 the model will interpret channel 17 directly as an SST anomaly.

Having obtained the SSTA, and with an SST climatology as well, the task is to translate this temperature anomaly in degrees C into a rain rate anomaly in mm/day. There are four options, depending on the value set for the transfer function switch ISSTTRAN.

ISSTTRAN=1 triggers a direct linear correspondence between SSTA and the precipitation rate anomaly. There is a dummy parameter DQDT, the rate of change of precipitation rate with SSTA, which is just set to 1. The translation thus follows geophysical constants to provide 1 mm/day for every degree of SSTA.

ISSTTRAN=2 activates the nonlinear transfer function. This is a highly tuned exponential function that depends not only on the SSTA but also on the climatological SST. The same SSTA has more impact in warm ocean than in cold. And the 200mb divergence also plays a role, attenuating the function in regions where you would not normally see upper level divergence. There are currently six tuneable parameters in this transfer function and it is still under development. Generally it does a better job than direct linear correspondence at mimicking observed interannual variations of precipitation in relation to interannual variations of SSTA.

Note that the precipitation anomaly thus calculated is passed into profileheat in the same way as it is from the convection scheme, and the diabatic heating will be placed into the same deep convective vertical profile. This anomaly is fixed, and it can be negative. So don't be alarmed by negative values of mean precipitation in your long climate run. The SSTA-induced precipitation anomaly will impact the tendencies of moisture and temperature. So it interacts with the moisture budget of the model a well as the heat budget. Finally - be warned that the SSTA-induced heating will be affected by alternative specifications for the latent heat of condensation. So if you set THERMTYPE to INTER, don't expect the model to respond to SSTAs !

ISSTTRAN=3 is a short cut if you want to force the model directly with an observed precipitation anomaly. The values read from channel 17 are interpreted directly in mm/day and passed to PROFILEHEAT.

ISSTTRAN=4 is a completely different approach more akin to a conventional GCM. It bypasses the calculation of anomalous precipitation in SSTTEND and instead adds the SSTA as a modification to the flux at the bottom boundary in VDIFFTEND. Fluxes of temperature and specific humidity are modified by changing their values in the imaginary sub-surface layer. The SST anomaly is added to the temperature and a quantity QDIFF is added to the specific humidity. QDIFF is the difference in saturated humidity between the climatological temperature normally ascribed to the sub-layer and the temperature with the SSTA added. So it is a nonlinear function of temperature and it's quite important in determining the model response to an SSTA. This option removes problems of double counting and negative precipitation associated with the anomaly-transfer-function approach.

There are some options for controlling your anomaly experiments. SCALESSTA can change the amplitude or sign of the SSTA in a similar way to SCALEFAN. SCALEPPTA does the same thing to the derived precipitation anomaly. If the transfer function is nonlinear it will not have the same effect. LSSTMASK can be switched on to limit the SSTA to the tropics, or to separate ocean basins in the tropics. These are controlled by SSTZONE, which can be set to TROIPICS, PACIFIC, ATLANTIC or INDIAN. It chooses useful values for the box boundaries of the mask, using the same code as for the nudging mask.

Temporal behaviour can also be controlled from the job script. LPERSIST will force the model to stick on the first record of SST that it reads. LOOPSST controls looping behaviour for cyclic anomalies. KSTOPSST assigns a timestep KOUNT at which to stop progressing through the SST file: useful for honest hindcasts. Finally the metadata on the historical weekly SST data is very useful for allowing the model to scroll through the data and start a seasonal hindcast at a specific date. This metadata is not present on every file so if you want to read it you must set LREADMETA=.T. The structure of the metadata is described in Appendix A.

5 Chapter 5: A Midsummer Night's Dream - Working Together

We hope that this manual provides you with a way into using DREAM, and that you find it is more than a user guide, also a source of scientific ideas for your diagnostic modelling. At the time of writing I think it would be a bit of an exaggeration to talk about a DREAM "community", but there is a loose affiliation of people who have used the model in one form or another. In Appendix E there is a list of publications that have used DREAM or its earlier variants, and hopefully this list will continue to grow. To keep in touch, and participate in our ongoing activity there are some collaborative resources in place.

5.1 The DREAM GitHub repositories

It's here: https://github.com/dream-gcm

This is where you have downloaded the code. It also hosts the DREAM web site, which is occasionally updated with the latest ongoing science. Also check on this web site for the latest contact information. Various other resources are also available on GitHub including some python code for generating plots, not covered in this user guide.

5.2 Slack channel

If you want to join the slack group (https://dream-fcq1579.slack.com), contact us via the web site and we'll let you in. It's a more dynamic and open ended way to collaborate than posting issues on GitHub or just exchanging emails. At this point, where the numbers involved are quite manageable, all means of communication are welcome.

5.3 Copyright, credit and collaboration

Downloading and using DREAM implies that you accept the copyright and terms of use of the GNU General Public License. Further to this, we request that users of DREAM proceed in a collaborative way. As well as the simple courtesy of recognising the effort put into development, a more collaborative approach is more productive for everyone concerned. Methods of using DREAM, and problems encountered are best discussed within the DREAM community, and as such they can be addressed far more effectively. This doesn't mean that everyone has to be involved in everything all the time. That would be grossly inefficient. But there is a risk that an individualistic (or even in one instance unauthorised) use of the model can (and did) lead to mediocre science. We are stronger together, and we can help and inspire one another. So whatever you do, keep in touch, and give credit where it is due.

6 Appendix A: DREAM Works - Model Code Basics and Data Structure

6.1 A1. The primitive equations and the semi-implicit timestep

DREAM solves the primitive equations expressed in terms of vorticity and divergence. These are written down in Hoskins and Simmons (1975, HS75) but let's deciphier this presentation by approaching them from the momentum equations, using subscript notation for partial derivatives:

$$u_t - fv = -uu_x - vu_y - wu_z - p_x/\rho \qquad (A1),$$
$$v_t + fu = -uv_x - vv_y - wv_z - p_y/\rho \qquad (A2).$$

Then:

$$\frac{\partial}{\partial x}(A2) - \frac{\partial}{\partial y}(A1)$$

gives the vorticity equation:

$$\xi_t + fD + \beta v = -u\xi_x - v\xi_y - \xi D - (wv_z + p_y/\rho)_x + (wu_z + p_x/\rho)_y,$$

which leads to an equation for the tendency of absolute vorticity:

$$\frac{\partial \zeta}{\partial t} = \frac{\partial}{\partial x}\{-u\zeta - wv_z - p_y/\rho\} - \frac{\partial}{\partial y}\{-v\zeta - wu_z - p_x/\rho\}.$$

The terms in curly brackets can be designated by the vector (F_u, F_v) and this vorticity equation is the curl of this vector. Likewise the tendency of the divergence can be deduced from the momentum equations:

$$\frac{\partial}{\partial x}(A1) - \frac{\partial}{\partial y}(A2)$$

eventually leading to a divergence equation as another expression in F_u and F_v with an added source related to the kinetic energy:

$$\frac{\partial D}{\partial t} = \frac{\partial}{\partial x}F_u + \frac{\partial}{\partial y}F_v - \frac{1}{2}\nabla^2(u^2 + v^2).$$

These equations are written in sperical coordinates in HS75 and thermodynamic, continuity and hydrostatic equations are added in sigma vertical coordinates as shown in fig. A1. Also shown in Fig. A1 are some of the variable names used in the model for these terms: spectral variables in blue and grid point variables in red. In these equations T' is a temperature deviation from a reference value which in DREAM is just set to 250K, $p*$ is the surface pressure, which in DREAM is calculated as the mean sea level pressure, and ϕ is the geopotential. The divergent part of the flow, expressed in equations 2-5 in fig. A1, can be written in summary form as a set of tendencies generated by flux convergences and other source terms. Eliminating for the divergence D gives a wave equation in with a source term. The semi-implicit time stepping approach is used to filter these gravity waves, by discretising this equation so that the wave equation operator acts on the centred average divergence $\bar{D}$, effectively filtering the fast gravity modes and allowing a longer time step. This development is summarised in fig. A2, including some more variable names from the code.

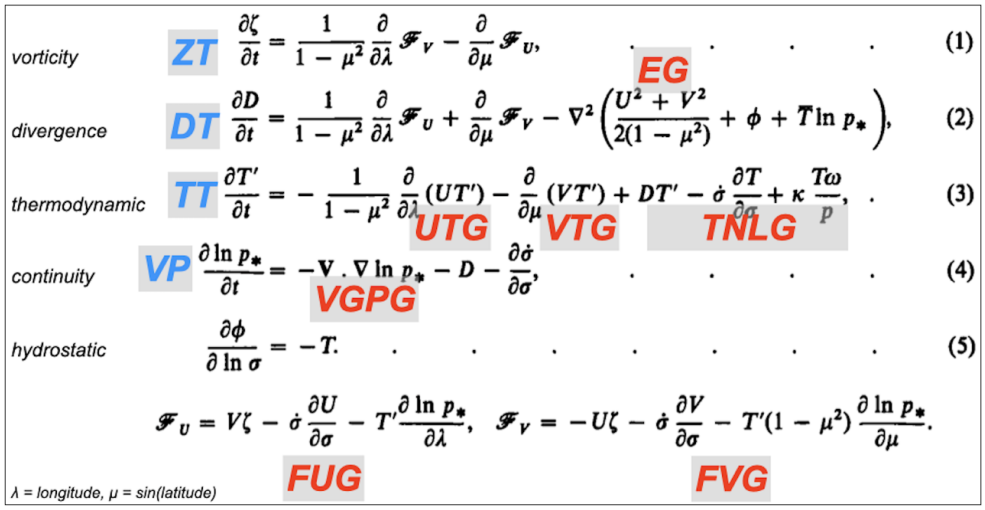


Fig. A1: The primitive equations as presented in HS75.

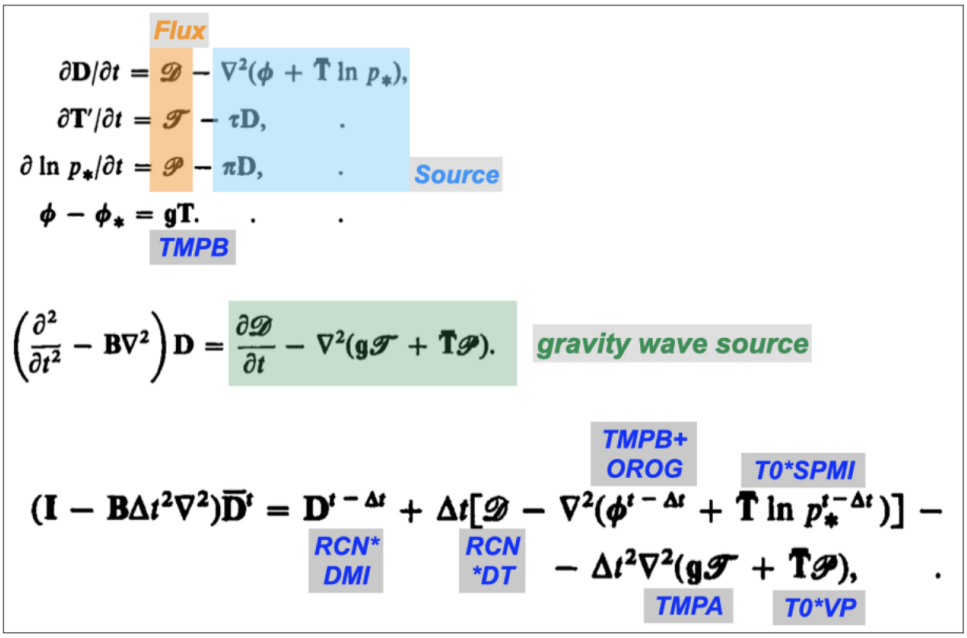


Fig. A2: Some more equations culled from HS75, outlining the semi-implicit timestep.

6.2 A2. Spectral truncation and data organisation

The model proceeds by calculating tendencies and then applying these tendencies to the model state to find the next model state using a semi-implicit centred difference timestep. Linear calculations for the tendencies are carried out directly in spectral space. Spatial derivatives are calculated as part of the transformation between spectral and grid space. Nonlinear advective terms are calculated in grid space. Model state variables therefore have both spectral and gridpoint arrays asigned to them, and in this section we will briefly outline how this data is organised.

Model variables are projected onto Fourier transforms in the zonal direction and Legendre polynomials in the meridional direction:

$$X = \sum X_n^m P_n^m(\mu) e^{im\lambda}$$

where m is the zonal wavenumber, and n is the meridional wavenumber (i.e. the number of zeros between the poles). The number of coefficients is limited by a "jagged triangular" truncation, which has the property of istoropy on a sphere (the resolution independent of direction or latitude), and equal numbers of even and odd coefficients with respect to symmetry about the equator, in total and also individually for each zonal wavenumber. This is illustrated in fig. A3 which shows how many coefficients you have for triangular truncation to an odd number, T5 and an even number T4. DREAM can easily be run at T31 or T42.

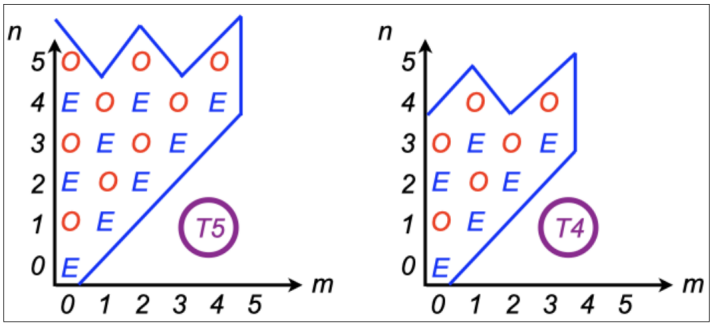


Fig. A3: Jagged triangular truncation for spectral coefficients.

For each model level, these complex coefficients are stored in increasing values of the meridional index n, embedded within increasing values of zonal wavenumber m. But this is done separately for even and odd coefficients. So for example, referring to fig. A3, the divergence, temperature and pressure variables at T5 would be stored in the following order: EEE,EE,EE,E,E,OOO,OO,OO,O,O. This data organisation makes it easy to create a state that is symetric about the equator: you just set all the odd coefficients (the second half of the binary record) to zero. This works because these fields are naturally symmetric about the equator (one does not expect divergence or temperature to change sign on the eqator). To make it work in the same way for the absolute vorticity, which is naturally antisymmetric, the vorticity coefficients are stored in the oppostie order: OOO,OO,OO,O,O,EEE,EE,EE,E,E. The jagged triangular truncation also provides a rapid way to impose cyclic symmetry in the zonal direction by selecting m as a multiple of the order of symmetry desired. Six-fold symmetry was used in the original baroclinic wave lifecycle experiments (Simmons ans Hoskins, 1978), but this feature has not yen been used with DREAM.

When the model variables are transformed to grid space they are stored on a Gaussian grid in latitude pairs, closing in towards the equator from the most polar latitudes to the most equatorial. There are MG equally spaced longitude points around the globe, with the first one situated on the Greenwich meridian. And there are JG different latitudes in each hemisphere, situated between the pole and the equator. Grid data is written for a given level in the following order:

- first latitude pair: `j=1 (north): i=1,MG, 0,0, j=JGG (south): i=1,MG,0,0`
- second latitude pair `j=2 (north): i=1,MG, 0,0, j=JGG - 1 (south): i=1,MG,0, 0`
- and so on until final latitude pair (closest to the equator) `j=JG (north): i=1,MG, 0,0, j=JG + 1 (south): i=1,MG,0,0.`

Note that the total number of latitudes `JGG=2JG`, and two dummy zeros are written per latitude so the longitiude index in the array is actually `MGG=MG+2`.

Gridpoint operations are carried out one latitude-pair at a time for all levels. So at any given moment only one latitude pair exists in grid space. The calculations proceed in zonal-vertical slices. Note that in some routines the data is in grid space in the meridional direction but in Fourier coefficients in the zonal direction. At this stage the data is still complex, but when the data is fully into grid space it is real. Real grid data shares array space in common blocks with hybrid latitude-Fourier data, so declarations can change from one subroutine to another with the same variable names. Eeek !

6.3 A3. Model variables and dimensions

The basic model variables are vorticity and divergence (s^{-1}), temperature (degrees C), surface pressure (Pa) and specific humidity (kg/kg). These quantities are non-dimensionalised using physical constants:

- **Time**: the angular velocity of the earth WW=Ω=2π /23.93 $\times$ 3600(^{-1}$). This immediately gives a scaling for vorticity and divergence, giving model variables:
  - `Z = Absolute vorticity / WW`
  - `D = Divergence / WW`
- **Distance**: the radius of the earth RADEA=a = 6371000 m This gives us a scaling for **Speed**: CV =RADEA*WW so in grid space UG and VG equal u and v divided by CV.
- **The gas constant**: GASCON=R = 287 J/(kg K) or $m^2/(s^2$ K). This gives us a scaling for temperature CT=CV*CV/GASCON. The temperature variable used in the model is thus `T = (Temperature in Kelvin - 250) / CT`.

The surface pressure appears in the model equations as SP=`ln(p*)`. Since DREAM does not have explicit orography, this `p*` is actually the mean sea level pressure referenced to 1000 hPa. This is calculated from the temperature and the geopotential height at 1000 hPa according to :

$$\ln(p_*/1000) = \left(\frac{gz}{RT}\right)$$

- Specific humidity Q is already dimensionless as kg of water vapour per kg of air.
- Other physical constants set and used in the model are:
  - Gravitational acceleration GA=g= 9.81 m^2/s^2,
  - The ratio of the gas constant to the specific heat capacity at constant pressure AKAP =$\kappa = R/Cp$ = 0.286,
  - The latent heat of condensation RLHEAT= $L = 2256476$ J/kg

Special attention needs to be paid to the nondimensionalisation of time intervals, timescales and rates. Since time is nondimensionalised in terms of an angular frequency Ω, this means that a non-dimensional day actually has a day length of 2π. So if a timescale is specified in days, for example the dissipation timescale in the free troposphere TAUFT=`20 days`, then it's non-dimensional value will be 20 $\times 2\pi$. The associated dissipation rate FTFR is the reciprocal of this: FRFT=`1./(PI2*TAUFT)` . Note that timescales are generally specified as TAUXX `(days)` and the associated rates as FRXX. The same logic applies to the nondiensional length of the model timestep DELT=`PI2/TSPD (=2*PI/64)`. For details of the reporting of time in model output and data see *Section A7* of this appendix.

6.4 A4. Vertical structure

DREAM currently uses 15 σ-levels in the vertical. They are referenced to the mean sea level pressure, as calculated from temperature and pressure at the 1000 hPa level as shown above. So model levels in DREAM are quite close to pressure levels, because there is no explicit orography. This does not mean that there is no orographic forcing in the model, because the absence of explicit orography is compensated for automatically in the empirical forcing of momentum. But it does mean that DREAM cannot simulate the interaction of transient systems with orography, because our empirical forcing is not flow-dependent.

The sigma levels used in DREAM have been chosen to be as close as possible to the ECMWF standard pressure levels on which the data was originally provided, minimising interpolation errors. An exact correspondence is not possible because the model sigma levels must fall at the mid point of model sigma layers. It is, in fact, the boundaries between sigma layers that are specified, not the mid-points. The layer boundaries start at zero at the top of the atmosphere and finish at unity at the bottom. Sigma levels are then calculated as the mid-points between these layer boundaries. The layer boundaries in DREAM have been chosen so that we end up with model sigma levels centred at the following fifteen values:

$\sigma \times$ 1000 = 37.5, 100,150, 200, 250, 312.5, 400, 500, 600, 700, 791.67, 850, 883, 925, 975.

The model uses the Simmons and Burridge (1981) angular momentum conserving vertical scheme. The term "vertical scheme" refers to the way in which the geopotential is calculated on sigma levels in the gravity wave source term shown in fig. A2. For a quick discussion of this see the appendix of Hall (2000).

6.5 A5. Dissipation

Scale selective hyperdiffusion is applied in spectral space to all the model's 3-d state variables (Z,D,T,Q) independently of vertical level. Any order of the Laplacian operator can easily be used because it reduces to a simple multiple of the spectral coefficients. A timescale is also specified. Currently the default horizontal diffusion is set to 12-hour ∇^6 .

Further level-dependent vertical diffusion and damping is added in grid space to all 3-d state variables (U,V,T,Q). Linear diffusive vertical fluxes are calculated from vertical gradients at sigma-level boundaries. Their convergence is then calculated at sigma levels, using linear centred differences. The boundary conditions act to damp the system as if the top and bottom surfaces mirrored the reference value in the adjacent layer, like a sponge at the top of the atmosphere or a fixed SST at the surface. The timescales used to calculate the fluxes depend on the model level, and are much shorter at the lowest levels. The damping rate (the reciprocal of the time scale) follows a linear profile from the surface to a specified boundary layer height. Standard parameters are shown in fig. A4. The mean rate in the boundary layer corresponds to a time scale of 16h. There is an optional doubling of the lowest layer vertical diffusion coefficient over land if `LLSD` is to to true. This is currently enabled by default at T42 but disabled at T31. Above the boundary layer, in the free troposphere, the vertical diffusion rate is fixed with a default timescale of 20 days.

In addition to difusion, there is a level-indepenent linear damping on temperature only, with default time scales of 10-days (T31) and 12-days (T42). This is intended as the crudest radiation scheme imaginable. In principle, simple in-situ linear damping like this could be added either in spectral or grid space. On the other hand, the vertical diffusion must take place at a geographical location in grid space. There is an option to vary the boundary layer coefficient as a function of latitude, with a different value at the equator that smoothly approaches the standard extratropical value at specified latitudes. Finally there is the optional uniform damping applied to all degrees of freedom in spectral space. This is only indended for use in stabilising fixed basic states.

The empirical forcing always acts hand in hand with the prescribed dissipation. Whenever a damping or diffusion parameter is changed, the forcing must be recalculated. If you find you are tinkering a lot with the dissipation parameters for simple GCM runs, but you don't want to go through the long process of finding a new `_fcm` forcing every time you make an adjustment, there is a short cut. If the dissipation is linear, the transient eddy part of the

forcing: `fbs-fcm` will not change. And it is quick to find the forcing for a fixed basic state _fbs. So if you want, you can calculate the transient forcing just once, and then go ahead and subtract it from a newly calculated `_fbs` forcing every time you change parameters. This will give you a new _fcm every time, which should be identical to the `_fcm` you would have calculated the long way by stepping through lots of inital conditions. For this to work your basic state must of course be the mean of your long dataset. See Appendix B for a mathematical explanation.

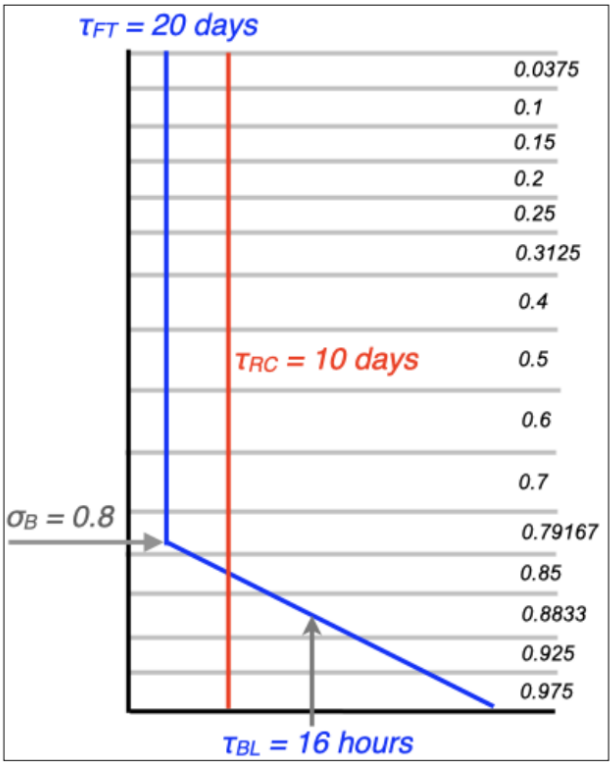


Fig. A4: Vertical profiles of diffusion and damping with associated time scales.

6.6 A6. Code structure

Most of the presentation of the code is in Appendix D, but here I will walk you through the main program to give you an idea of how one timestep unfolds. Have the source code open in front of you while you read this.

6.6.1 Model setup

A few subroutines are called to set up things that are not going to change thoughout the model run. `INISET` to set up namelist variables, nondimensionalisation constants and factors for Fourier transforms etc. `INIGAU` calculates Gaussian weights and latitudes. `INISI` sets up sigma levels, the vertical scheme and the semi-implicit scheme. INIRES defines damping coefficients from timescales given in the namelist. `INIVAR` initialises some model variables such as grid point fields, masks etc, and initialises the spectral state variables to zero.

6.6.2 The training loop

The loop from 1 to KTFIN starts next and is bounded by line number 111. It is only used for training runs (LTRAIN=.T.). It actually runs the model multiple times for one timestep from a sequence of initial conditions.

6.6.3 Initialisation

State variables are reinitialised to zero for good measure, and then read from the input data in INIIC. Then if necessary we skip through the reference data and cyclic forcing files to the correct calendar date to get ready to read in the basic forcing. READFCE reads the forcing and reference data and then, conditionally on options set in the namelsit we read the forcing anomly, nudging data and SST data.

Once this is all done the intial condtion is written out as the first record of the model's spectral history file at RKOUNT=0. For consistency in the number of records per file, gridpoint history records are also output at this point. But the first record of the grid history file will only contain the intial zeros, because no gridpoint information has yet been calculated.

6.6.4 The time loop starts

This is a loop over KOUNT=1 to KRUN bounded by line number 1000. The real DAY is incremented each timestep. First there is a conditional re-read of all the forcing, nudging and SST data contingent on the update periods that have been specified. This is also where the spectral humidity flux convergence is periodically re-intialised because it is needed in the calculation of the reference value in grid space.

6.6.5 Calculation of advective tendencies

The tendencies are first initialised to zero. Then the first set of transforms from spectral to grid space takes place. This is a short piece of the main program but much goes on here.

In W2GA there are two steps. First a call to LTI transforms Legendre polynomials to the Gaussian latitude grid. Along the way meridional and zonal pressure gradients are calculated, and vorticity and divergence are used to calculate zonal and meridional winds, streamfunction and velocity potential. At this point the data is still spectral in longitude. Conversion to the longitude grid then takes place with multiple calls to FFT991.

The subroutine ADVECTION computes all the advective tenencies by multiplying winds with gradients. These tendencies correspond to the advective terms on the right hand side of the primitive equations. We do not yet have tendencies for model state variables.

The variables associated with these terms are then transformed back to spectral space in G2WA. This performs multiple Fourier transforms (FFT991) and then direct Legendre transforms (LTD), reconstituting the desired terms in the equations from various zonal and meridional gradients along the way.

6.6.6 The semi-implicit timestep

Having calculated all these terms, the spectral coefficients of state variables can now be stepped forward in time. The subroutine TSTEP contains vertical integral calculations to furnish the gravity wave source term, and updates the state variables with both linear and advective tendencies. The job is not complete, even for a purely dynamcial time step, because we are still inside a time filter and the diffusion is yet to be added.

6.6.7 Calculation of diabatic tendencies

But first, lets find all the tendencies due to simple parameterisations in grid space. Spectral tendencies are first reinitialised to zero in INITEND and some preliminary operations are carried out in spectral space (SPECPREP) to smooth (truncate) the moisture flux convergence.

Then there is another round of spectral transforms (W2GD). Once in grid space the routine DIABATIC calls all the individual schemes to find tendencies associated with nudging, deep convection, SST anomalies, vertical diffusion, land-sea drag, large scale rain and radiative relaxation. It then adds all these tendencies together into a grand total that is passed back for transformation into spectral space (G2WD) to give diabatic tendencies for the model's state variables.

Before doing this it also records some purely diagnostic gridpoint variables associated with precipitation, putting them into a more convenient format for output (SHUFFLE takes latitude pairs and writes them into a grid that is sequential from north to south).

Finally SPECPOST tidies up some of the spectral tendencies (makes sure there is no global diabatic tendency applied to vorticity, divergence or surface pressure).

6.6.8 Spectral forcing anomalies, damping and diffusion

Subroutine FANDAMP adds the forcing anomaly that has been read from the _fan file to the spectral tendencies. It also adds stabilising damping on all degrees of freedom if requested. Then DIFUSE calculates the scale selective hyperdiffusion and adds this to the tendencies, all in spectral space.

6.6.9 Completing the timestep and adding the empirical forcing

Subroutine DSTEP updates the state variables with the accumulated diabatic tendencies. This is much more straightforward that TSTEP as no extra calculations are requred at this point. The empirical forcing is then applied directly to the state variables in FSTEP. Then in TFILT there is a final adjustment to the values of the state variables on their previous timestep (ZMI etc) to close the time filter PNU.

6.6.10 Output a the end of the timestep

If you're looking for normal modes, the modefinder is engaged here by calling SCALEDOWN, after the timestep has completed. This is because it is a direct intervention on the state variables, not on the tendencies.

Then we check to see if we are on a timestep for reporting the output (a multiple of KOUNTH). After some timing calculations to get the model year in the right format, history records are written (spectral and grid), and occasionally a restart record as well.

The time loop ends and then if we are at the end of the run, all that is left is to write a final restart record. If we're in a training loop, this is done for each initial condition until the training loop also ends.

6.7 A7. Data timing

6.7.1 Reading, writing and initialising

A history record containing the complete state of the model is periodically written as a binary file during the run as follows:

```
WRITE(9)RKOUNT,RMYR,DAY,Z,D,T,SP,Q,YEAR
```

where:

- KOUNT is the current integer time step and RKOUNT is the corresponding real number.
- DAY = RKOUNT/TSPD is the real day number
- RMYR is the current model year, counting up from the initial condition in YYYY.DDDDD format, with a 365.25-day year.
- YEAR is a real number YYYY.DDDDD that is initialised from to the reference data. It keeps track of leap years, so it is different to RMYR and is useful for keeping track of time in the real world from the ERAi dataset. In practice the implementation in the model is incomplete so there is room for improvement here.

When the model is initialised it takes YEAR from the value provided by the initial condition. It then sets RMYR to the same thing for a cycle run, but to 100.00000 for a perpetual run. RMYR can also be set by SST metadata, which takes priority over the information in the initial condition.

From then on, in a model run, YEAR is only changed when it is updated by reading it from the reference state. Note that if this reference state is the mean annual cycle then YEAR will be set to 100.DDDDD.

RMYR, on the other hand, increments every time a history record is written. So if RMYR=YYYY.DDDDD, then YYYY is a model year that increments by one every 365.25 days, and DDDDD is the value of DAY times 1000. So RMYR will count round to YYYY.36500 and then go on to YYY(Y+1).00000.

At the end of a model run (and at certain long intervals during the run) a restart record is written. This contains all the information necessary to restart the run as a smooth continuation of the integration:

```
WRITE(12)RKOUNT,RMYR,DAY,Z,D,T,SP,Q,YEAR,ZMI,DMI,TMI,SPMI,QMI,RNTAPE
```

where: ZMI,DMI etc denote the values of state variables at the previous timestep. For a restart record RNTAPE=100. and serves as a check on correct reading of the binary file.

6.7.2 The DREAM dataset

The dataset provided with DREAM currently spans the calendar years 1979-2016. Four-times daily data is included at 00Z, 06Z, 12Z and 18Z. This is all assembled into one history record in the same format as the model output. This is the file ERAi_seq4x_1979-2016_ANN.b.

The entire dataset amounts to 55520 records for 13880 days. At T42 one record is 901864 bytes and the entire dataset is 50 GB. At T31 one record is 499752 bytes and the entire dataset is 28 GB.

A single annual cycle of four-times daily data for 365.25 days contains 1461 records. This is the file ERAi_cyc4x_1979-2016_RM41.b.

Since the 38-year dataset is written in model output format, some of the model counters have been included. It is written as:

```
RKOUNT,YEAR,DAY,Z,D,T,SP,Q,RMTAPE
```

In general RNTAPE takes different values depending on the type of data: RNTAPE = 100 (model runs), 200 (ERAi data), 300 (empirical forcing) or 400 (forcing anomaly). The way the ERAi data is organised, the value of RKOUNT increments throughout the real calendar year as if it were model output, but resets to zero at 0Z at the beginning of every new calendar year. DAY also increments throughout the year resets at the new year.

Output from a long run of the model would look slightly different because it would report as:

```
RKOUNT,RMYR,DAY,Z,D,T,SP,Q,YEAR
```

and RMYR would not exactly follow the value of YEAR because the model year has a fixed length of 365.25 days.

For example, the Intergovernmental Panel on Climate Change (IPCC) was created on 6th December 1988. This was the 341st day of 1988. Let's pick the time 12Z. Since the first day of the year is always day 0, this would lead to a label for the year as YEAR = 1988 + 340.50/1000 = 1988.34050.

This would be record number 14511 of the ERAi dataset: 4 x (7x365+2x366+340) + 3 = 14511. A long model run starting at 0Z on 1 Jan 1979, with 4x-daily output, would, at record 14511, set

DAY=(14511 - 1)/4 = 3627.5 and RMYR = 1979 + INT(DAY/365.25) + (the remainder from DAY/365.25)/1000 = 1988.34025.

The six-hour discrepancy between RMYR and YEAR is just because of fixed-length years compared to leap years. It doesn't matter because there is no diurnal cycle in the model. Table A1 shows further examples of how to keep track of model years and calendar years. Note that although the dataset spans 38 calendar years exactly, the last two records enter the 39th model year.

record	record mod 1461	year	month	date	hour	DAY	DAY mod 365.25	RKOUNT	RKOUNT mod 23376	YEAR	RMYR
1	1	1979	Jan	1	0	0,00	0,00	0	0	1979,00000	1979,00000
2	2	1979	Jan	1	6	0,25	0,25	16	16	1979,00025	1979,00025
3	3	1979	Jan	1	12	0,50	0,50	32	32	1979,00050	1979,00050
4	4	1979	Jan	1	18	0,75	0,75	48	48	1979,00075	1979,00075
5	5	1979	Jan	2	0	1,00	1,00	64	64	1979,00100	1979,00100
1459	1459	1979	Dec	31	12	364,50	364,50	23328	23328	1979,36450	1979,36450
1460	1460	1979	Dec	31	18	364,75	364,75	23344	23344	1979,36475	1979,36475
1461	1461	1980	Jan	1	0	365,00	365,00	23360	23360	1980,00000	1979,36500
1462	1	1980	Jan	1	6	365,25	0,00	23376	0	1980,00025	1980,00000
1463	2	1980	Jan	1	12	365,50	0,25	23392	16	1980,00050	1980,00025
14511	1362	1988	Dec	6	12	3627,50	340,25	232160	21776	1988,34050	1988,34025
55517	1460	2016	Dec	31	0	13879,00	364,75	888256	23344	2016,36500	2016,36475
55518	1461	2016	Dec	31	6	13879,25	365,00	888272	23360	2016,36525	2016,36500
55519	1	2016	Dec	31	12	13879,50	0,00	888288	0	2016,36550	2017,00000
55520	2	2016	Dec	31	18	13879,75	0,25	888304	16	2016,36575	2017,00025

Tab. A1: Timing in the DREAM dataset for 4x-daily data over 38 calendar years.

6.7.3 The SST metadata

SST data is read in binary grid form and some datasets contain metadata before the SST on every record. This metadata contains timing information, like a poor man's netCDF. To recognise this and read it properly there is an option LREADMETA. It only applies to SST (channel 17) and should only be enabled if the metadata is present. SST data without metadata (idealised anomalies for example) should be read with LREADMETA=.F., and climatological SSTC data (channel 18) does not contain metadata.

The metadata interfaces with the namelist specifications for KBEGYRSST and KBEGMNSST. When the SST is initialised with metadata enabled, it will scroll through the data file until it reaches the first record with the specified year and month. It will then read this SST data and set the model variable RMYR to the specified year and the first day of the specified month in a real calendar. This ensures a correct start date in NetCDF output for forecast runs.

The metadata consists of six integers before the SST data:

```
MYEAR,MMONTH,MREMAIN,IYEAR,IMONTH,IDAY,SST
```

where IYEAR, IMONTH and IDAY are the calendar year, month and start date of the one-week period covered by the SST data record. This is for information only and is not used by the model.

MYEAR and MMONTH are the year and month for which this week-long SST state would apply if a forecast were started on the first of the month. In this way the model will pick the right SST to start its forecast, even if the week in question straddles the first of the month, which is usually the case. As long as LPERSIST is not enabled, the model will update its SST state every seven days. But at the beginning of a forecast the model will need to know how long to stay on the first SST state it reads: probably less than seven days. This information is provided by MREMAIN, which is used to offset the calculation for DIFKSST in the main body of the code and control the call to READSST.

7 Appendix B: Field of Dreams - The General Theory Behind DREAM Forcing and Solutions

7.1 B1. Quick guide

Whenever I try to explain the way DREAM is forced I seem to run into difficulties. Is it because it's really complicated ? Is it because I'm bad at explaining things ? Is it because people are generally a bit thick ? I don't really accept any of these reasons ! But I know it's impossible to explain it in a short conference talk, and even in a longer presentation it is tempting to go into too much depth. But here we can relax and look at all the implications of a forcing method that I still believe to be simple and effective.

Let's start with notation that is intuitive for people who were bought up in the geosciences. Consider the development of some variable, say the potential vorticity q, in the real atmosphere:

$$\frac{\partial q}{\partial t} + \mathbf{v}.\nabla q = \mathcal{F}(t) - \mathcal{D}(q) \qquad (B1)$$

What we have here is tendency, advection, external forcing and state-dependent dissipation. We can simulate the advection with our model, and we can specify damping and diffusion and tune it as we please. The biggest challenge is to find a way of specifying the forcing $\mathcal{F}$. We know what it is physically. It represents source terms like radiative heating, boundary fluxes or condensation. But it's a tough job to specify it in a simple way.

What we know for sure is that if we ran the model without any forcing, from any realistic initial condition, it would quickly develop large systematic errors. So one prescription for the forcing might be a source term for q that corrects the average systematic error of an unforced model. Let's assume that we are in a statistically steady climate state (i.e. we consider one season). And let's also decide to impose an external forcing that time independent. This is not realistic, but it does lead to the great simplification that our external forcing is just the time mean of $\mathcal{F}$. We can readily see this by taking the time mean of (B1)

$$\overline{\mathcal{F}} = \overline{\mathbf{v}.\nabla q} + \overline{\mathcal{D}(q)} \qquad (B2)$$

Evidently if we use this to force a model that develops with the same advection and dissipation, i.e.

$$\frac{\partial q}{\partial t} + \mathbf{v}.\nabla q + \mathcal{D}(q) = \overline{\mathcal{F}} \qquad (B3)$$

then any statistically steady integration of this model will also satisfy equation (B2). This does not mean that the model's climatology will be identical to the observed climatology. There is more than one way to satisfy (B2), because at least the advection term is nonlinear. The time mean advection contains contributions from advection of the mean flow and transient fluxes. Let's have a look:

$$\overline{\mathcal{F}} = \overline{\mathbf{v}}.\nabla\overline{q} + \overline{\mathbf{v}'.\nabla q'} + \mathcal{D}(\overline{q}) \qquad (B4)$$

So you see the same forcing could balance different partitions between the first two terms on the right hand side, associated with different model climatologies. A model forced in this way may have systematic errors, just like any GCM. Only the time mean generalised total flux is constrained to conform to observations. Note that in (B4) we have assumed that the dissipation is linear.

It remains to find a way of calculating this time-independent forcing. If you've read Chapter 4, part 2 then you already know the answer. The time mean of $\mathcal{F}$ is just the exact opposite of the time mean of all the tendencies that would be generated by the unforced model if it were initialised from a representative set of observed states. So the trick is to take the model and initialise it many times, from a range of initial conditions, without any forcing. Run it for one timestep for each initial condition. Take all those one-timestep tendencies and average them together. $\mathcal{F}$ is just minus that.

7.2 B2. Forcing a simple GCM

I tried to keep the quick guide quick. If you're still reading, it's because you want more. Be careful what you wish for. We're now going to switch to a more generalised notation in which the instantaneous state of the atmosphere is represented by a state vector $\mathbf{\Phi} = (\zeta, D, T, q, p_* \dots)$, which represents vorticity, divergence, temperature, specific humidity and surface pressure in the same basis as the model. The development of the observed atmosphere can then be written as

$$\frac{d\mathbf{\Phi}}{dt} + (\mathcal{A} + \mathcal{D})\mathbf{\Phi} = \mathcal{F}(t) \qquad (B5)$$

where $\mathcal{A}$ is a nonlinear advection operator. Let's separate the state vector into time-mean and transient components.

$$\frac{d\mathbf{\Phi}'}{dt} + (\mathcal{A}+\mathcal{D})(\overline{\mathbf{\Phi}} + \mathbf{\Phi}') = \overline{\mathcal{F}} + \mathcal{F}' \qquad (B6)$$

As before, the time mean of this equation can be written

$$(\mathcal{A}+\mathcal{D})\overline{\mathbf{\Phi}} + \overline{O(\mathbf{\Phi}'^2)} = \overline{\mathcal{F}} \qquad (B7)$$

This is the generalised form of (B4). We see that time-mean advection is balanced by transient eddy fluxes and forcing. We could move the second term to the right hand side and call it the "transient eddy forcing". Subtracting (B7) from (B6) we obtain the transient budget with respect to the time mean

$$\frac{d\mathbf{\Phi}'}{dt} + \mathcal{L}\Phi' + \left[O(\mathbf{\Phi}'^2) - \overline{O(\mathbf{\Phi}'^2)}\right] = \mathcal{F}' \qquad (B8)$$

In (B8) all the terms have zero time mean, so there is no large cancellation. This means that the time-mean state is a realistic basis for perturbation experiments. The development of small perturbations is conditioned by the linear operator $\mathcal{L}$, which itself depends on the time mean state. And the structure of these small perturbations may be relevant to observed transient systems. For deeper philosophical insight into these matters see Hall and Sardeshmukh (1998).

Now back to the practical problem of forcing a simple GCM. Let's introduce a model, with a state vector $\mathbf{\Psi}$ in the same basis as $\mathbf{\Phi}$. If it is forced with a constant source field, it will develop according to:

$$\frac{d\mathbf{\Psi}}{dt} + (\mathcal{A}+\mathcal{D})\mathbf{\Psi} = \mathcal{G}_{cm} \qquad (B9).$$

As before let's set our simple GCM forcing $\mathcal{G}_{cm} = \overline{\mathcal{F}}$. To find it we run the model with no forcing for one timestep, providing us with a set of tendencies:

$$\frac{d\mathbf{\Psi}}{dt} + (\mathcal{A}+\mathcal{D})\mathbf{\Psi} = 0 \quad \Rightarrow \quad (\mathcal{A}+\mathcal{D})\mathbf{\Psi} = -\frac{d\mathbf{\Psi}}{dt}$$

Do this many times with a set of observed states $\mathbf{\Phi}_i$ as initial conditions, and take the average of all the tendencies:

$$\mathcal{G}_{cm} = \frac{1}{n}\sum_{i=1}^{n}(\mathcal{A}+\mathcal{D})\mathbf{\Phi}_i \quad .$$

If we use this forcing to perform a long integration of the model we can compare our simulation with the dataset we used to generate the empirical forcing. And we find that this prescription for the forcing guarantees that the total generalised flux from the model simulation will be the same as in the observations, i.e.

$$\overline{(\mathcal{A}+\mathcal{D})\mathbf{\Psi}} = \overline{(\mathcal{A}+\mathcal{D})\mathbf{\Phi}} \quad .$$

But for reasons already explained above, it does not guarantee that the simulated time-mean flow will be realistic, i.e. $\overline{\Psi} \neq \overline{\mathbf{\Phi}}$.

Neither does it guarantee that the transient fluxes will be realistic. This is because the balance of terms in:

$$(\mathcal{A}+\mathcal{D})\overline{\mathbf{\Psi}} + \overline{O(\mathbf{\Psi}'^2)} = (\mathcal{A}+\mathcal{D})\overline{\mathbf{\Phi}} + \overline{O(\mathbf{\Phi}'^2)}$$

can be achieved differently on the two sides of the equation, as already discussed above directly in terms of mean flow and transient fluxes (equation B4).

So we have a forcing that is time-independent. It formally corrects the average initial systematic error that you'd get if you had no forcing, and it ensures that the generalised flux convergence in the model solution is identical to that in the observational dataset. But it does not guarantee a realistic climate, or realistic transient fluxes. The model will typically have systematic errors, like any other GCM, associated with compensating errors in mean and transient fluxes, and between transient fluxes on different timescales.

7.3 B3. Forcing a perturbation model with a fixed basic state

To maintain a climatological mean state against time-mean advection requires a combination of diabatic forcing and transient eddy fluxes. The sum of these two effects could be equated to the first term in (B7). So let's do that, and define a new forcing:

$$\mathcal{G}_{bs} = (\mathcal{A}+\mathcal{D})\overline{\mathbf{\Phi}}$$

From (B7) we see that this forcing is the same forcing we had before ($\mathcal{G}_{cm}$) plus the "transient eddy forcing" alluded to previously. And no, that's not what "bs" stands for, it stands for "basic state". What would happen if we forced the model with $\mathcal{G}_{bs}$ and initialised it with the climatology $\mathbf{\Psi} = \overline{\mathbf{\Phi}}$? Well, of course, nothing would happen ! The forcing would exactly cancel the tendency produced by the initial condition and the integration would proceed with

no development. It's actually much easier to find this forcing that to find the simple GCM forcing, because all you have to do is run the model for one timestep from a single initial condition. In this example that initial condition would be the time-mean state, but this can work for any basic state you want. What is the point of doing this ? Well, you can then study the development of of the solution due to perturbations, either in the initial condition or in the forcing.

If we add a perturbation $\mathbf{\Psi}_1$ to the initial condition, the development equation becomes

$$\frac{d\mathbf{\Psi}_1}{dt} + \mathcal{L}\mathbf{\Psi}_1 + O(\mathbf{\Psi}_1^2) = 0$$

We can make sure that $\mathbf{\Psi}_1$ is small (and remains small) we have a linear perturbation model

$$\frac{d\mathbf{\Psi}_1}{dt} + \mathcal{L}\mathbf{\Psi}_1 = 0 \qquad (B10)$$

The solution to this equation gives the normal mode structure associated with the climatology $\overline{\mathbf{\Phi}}$ (or any other basic state, with appropriate $\mathcal{G}_{bs}$). If $\mathcal{L}\mathbf{e}_n(x,y,z) = \lambda_n \mathbf{e}_n(x,y,z)$ and $\lambda_n = \sigma + i\omega$

then $\mathbf{\Psi}_1 = \mathbf{e}_n(x,y,z)e^{(\sigma+i\omega)t}$ or $\mathbf{\Psi}_1 = [A(x,y,z)\cos\omega t + B(x,y,z)\sin\omega t]\, e^{\sigma t}$, an exponentially developing normal mode with a spatial structure that cycles periodically between A, B, -A, -B.

Instead of adding a perturbation to the initial condition we add also perturbation to the forcing (to obtain the linear response it is sufficient to multiply some realistic forcing perturbation by a small factor, and then scale the solution back up by the same factor for diagnsotics). If the forcing anomaly is added to the right hand side of (B10)

$$\frac{d\mathbf{\Psi}_1}{dt} + \mathcal{L}\mathbf{\Psi}_1 = \mathbf{f}_1$$

the solution is:

$$\Psi_{1j}(t) = \frac{f_{1j}}{\lambda_j}\left(e^{\lambda_j t} - 1\right)$$

for jth projection of $\mathbf{\Psi}_1$ and $\mathbf{f}_1$ onto $\mathcal{L}^T$ and jth eigenvalue of $\mathcal{L}$. If the eigenvalues \lambda_j all have negative real parts σ, the basic state is stable for the model's current set of dissipation parameters. Under these circumstances the model can be used to integrate forwards in time and find asymptotic time-independent perturbation solution $\mathbf{\Psi}_1 = \mathcal{L}^{-1}\mathbf{f}_1$.

But basic states that have realistic zonal winds are usually unstable, and so the normal modes described above will usually emerge after about 20 days regardless of how the model is perturbed. Structures and growth rates of normal modes can be elegantly teased out of a long integration using the modefinder (Chapter 4, section 4a).

But by their nature such normal mode solutions have little to do with the details of a forcing perturbation, so unless they are themselves the object of study, they can be a nuisance. For stationary wave perturbation studies we often need a stable system, so that a time integration will converge to a steady response. An approximate way to get this is to damp the entire system for a selection of damping rates. The trick is to damp all degrees of freedom equally, so as not to interfere with the structures of the eigenmodes. The procedure is also outlined inChapter 4, section 4a, section 4i.

7.4 B4. A word on damping and restoration

As discussed in Chapter 1, section 2, a popular way to force simple advective models is through "restoration" to a radiative-convective equilibrium state or a reference climatology. We imagine this as a spring, which returns the atmospheric state to what it would be if there were no dynamical fluxes, or at least prevents a model from straying too far from a realistic state. It's worth briefly exploring the connection between restoration forcing and our empirical forcing approach in which we specify the dissipation, but not the equilibrium state.

Let's rewrite the model development equation (B9):

$$\frac{d\mathbf{\Psi}}{dt} + \mathcal{A}\mathbf{\Psi} = \mathcal{G}_{cm} - \mathcal{D}\mathbf{\Psi} = R(\mathbf{\Phi}^* - \mathbf{\Psi})$$

On the right we have restoration towards a specified fixed equilibrium state $\mathbf{\Phi}^*$ at rate R. We can identify this state as $\mathbf{\Phi}^* = \mathcal{G}_{cm}/R$ and the rate R is simply the local damping rate $\mathcal{D}$. So it appears that the two approaches are mathematically identical. Instead of specifying a damping rate and an ad-hoc restoration state, we specify a damping rate and an objective empirical forcing.

Strictly, for this equivalence to hold, our dissipation needs to be just a local damping, so $\mathcal{D}$ is linear and diagonal. The dissipation in DREAM actually has off-diagonal elements associated with diffusion. We could still calculate the associated restoration state if we wanted, it's just hard to think of a scientific motivation for doing it.

7.5 B5. Nudging

Imagine you want to examine the influence of an observed sequence of events $\mathbf{\Phi}_i$ in a specified region on the large scale circulation. It is possible to artificially constrain the model within this specified region by nudging. This is another kind of restoration forcing but it does not correspond to any real physical process. Think of it as an additional term in (B9):

$$\frac{d\mathbf{\Psi}}{dt} + (\mathcal{A} + \mathcal{D})\mathbf{\Psi} = \mathcal{G}_{cm} + \left(\frac{\mathbf{\Phi}_i - \mathbf{\Psi}}{\tau}\right)$$

where τ is the nudging timescale. The procedure for implementing nudging is outlined in Chapter 4,section 4b.

Note that you need to be careful if you try to use nudging as a linear perturbation. This is normally achieved by scaling down the forcing perturbation with some small factor ϵ. But in this case you can't just multiply the nudging term by , because if the model state is always close to a specified basic state, then the nudging term above will just become an additional constant forcing, not a spring. So instead of nudging towards the state $\mathbf{\Phi}_i$, we need to nudge towards a state $\mathbf{\Phi}_i^*$ whose departure from the climatology $\overline{\mathbf{\Phi}}$ is scaled down $\mathbf{\Phi}_i^* = \overline{\mathbf{\Phi}} + \epsilon(\mathbf{\Phi}_i - \overline{\mathbf{\Phi}})$.

If we do this, then the nudging term becomes a small perturbation forcing and a damping on the anomaly response. The nudged version of (B10) can be written as:

$$\frac{d\mathbf{\Psi}_1}{dt} + \mathcal{L}\mathbf{\Psi}_1 = \epsilon\left(\frac{\mathbf{\Phi}_i - \overline{\mathbf{\Phi}}}{\tau}\right) - \frac{\mathbf{\Psi}_1}{\tau}$$

Fig. B1: Say no more !

7.6 B6. Diagnosing nonlinear and transient forcing

There are many studies in which some kind of perturbation is added to a dynamical model, and the response is analysed in terms of the direct response to the perturbation, plus an additional response to the modified transient fluxes. The direct response might be linear or nonlinear (the time independent nonlinear component is sometimes called the "stationary nonlinearity').

If the transient fluxes have been modified in a perturbation run, then this is surely a finite amplitude nonlinear solution. In this case the direct response to the implied change in "transient eddy forcing" can also be diagnosed, and used to explain how the total response is different to the direct response.

A lot of studies start to get a bit qualitative at this point. One can look at eddy fluxes, E-vectors, their divergences, the local tendencies due to the anomalies in these fluxes, etc. And one can reason that this is why, for example, the PNA pattern is stronger in the GCM than it is in the simple stationary wave model. This is all good, traditional, midlatitude dynamics.

The neat thing about DREAM is that it is relatively easy to be more quantitative. In fact, the direct response to any forcing anomaly is not just local. This is equally true for a transient eddy forcing anomaly as it is for the original forcing anomaly that triggered the direct response. With DREAM, you can diagnose every source, and separately calculate every response, both local and non-local.

Consider a simple GCM experiment with a forcing perturbation. First do a control run (B9) that yields a model climatology $\overline{\mathbf{\Psi}}_c$. Now do a perturbation run :

$$\frac{d\mathbf{\Psi}_p}{dt} + (\mathcal{A} + \mathcal{D})\mathbf{\Psi}_p = \mathcal{G}_{cm} + \mathbf{f}_1$$

Take the time mean and subtract from the time mean of the climatology run, and you get:

$$(\mathcal{A}+\mathcal{D})\overline{\mathbf{\Psi}}_p - (\mathcal{A}+\mathcal{D})\overline{\mathbf{\Psi}}_c = \mathbf{f_1} + \mathbf{TE}$$

Where **TE** is time-mean difference between quadratic terms in the transients from the two runs. This is the transient eddy forcing anomaly, and it is easy to find for the entire model state vector. We know the time mean for both runs and we know the forcing perturbation. So we can work out **TE** using the same techniques as we used to find above. Now we know both $\mathbf{f}_1$ and **TE**, we can use them together and separately in perturbation experiments about the fixed basic state $\overline{\mathbf{\Psi}}_c$. You'll probably find that the linear solution about this basic state to the forcing perturbation $\mathbf{f_1} + \mathbf{TE}$ is pretty close to the difference between the two GCM runs. I told you it was neat !

7.7 B7. Forcing the annual cycle

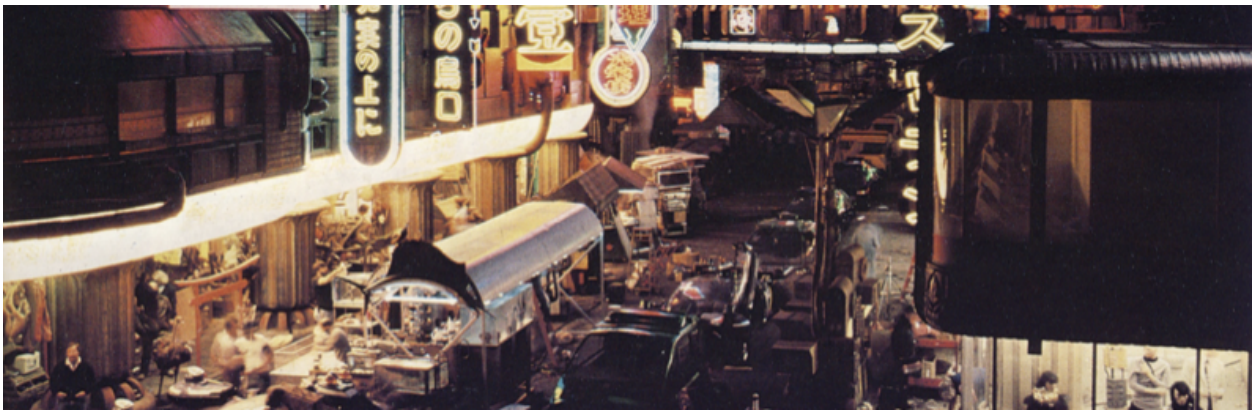
Some uses of DREAM require a direct correspondence to real-world events, like SSTs or nudging data. For these use cases the model simulation must have a simple way of changing with the seasons. The constant forcing described up to now needs to be extended to include a repeating annual cycle. If we use the notation tilde to represent such a cycle, which contains the annual frequency and its harmonics, then the forcing we seek can be expressed as $\mathcal{F} = \overline{\mathcal{F}} + \widetilde{\mathcal{F}}$, and we can consider a full spectrum of time variation in observed states $\mathbf{\Phi} = \overline{\mathbf{\Phi}} + \widetilde{\mathbf{\Phi}} + \mathbf{\Phi}'$.

Where prime now denotes any variation that does not repeat annually. If we substitute this into (B5) then our expression for the forcing (equivalent to B7) becomes

$$\overline{\mathcal{F}} + \widetilde{\mathcal{F}} = \frac{d\widetilde{\mathbf{\Phi}}}{dt} + (\overline{\mathcal{A}} + \widetilde{\mathcal{A}})(\overline{\mathbf{\Phi}} + \widetilde{\mathbf{\Phi}} + \mathbf{\Phi}') + \mathcal{D}(\overline{\mathbf{\Phi}} + \widetilde{\mathbf{\Phi}})$$

The main change is the first term on the right hand side which must be calculated directly from the data. The rest is calculated from one-timestep unforced model runs as before. The complete procedure is described in Chapter 4, section 2d. The second term contains a large number of timescale interactions which are exhaustively explored in Hall, Leroux and Ambrizzi (2019). They use a dummy multiplier technique in the data to isolate each term, but this is beyond the scope of this user guide. If you just want to run with an annual cycle then it is sufficient to calculate $\overline{\mathcal{F}} + \widetilde{\mathcal{F}}$ as a single diagnostic.

8 Appendix C: Do Androids Dream of Electric Sheep ? - Default values in namelist.



All the namelist variables are listed here with their default values and a brief explanation.

8.1 C1. Setup

- KRUN = 0 - The timestep on which the run will end (eg set to 320 for a 5-day run).
- KTFIN = 1 - Number of times the model is run when LTRAIN=.T. - should be set equal to number of records in initial condition sequence.

The following three string variables engage packages of namelist parameters - there is no default.

- RUNTYPE - Specifies the type of run in terms of forcing or symmetry: CYCLE, PERPETUAL, UNFORCED, or CHANNEL.
- THERMTYPE - Specifies the type of moist thermodynamics: DRY, WET or INTER.
- SSTZONE - Specifies the region in which SSTs are read: TROPICS, PACIFIC, ATLANTIC or INDIAN.

8.2 C2. Initial

8.2.1 Physical constants

- GA = 9.81 - Acceleration due to gravity - m/s^2
- GASCON = 287. - The gas constant R - J/kg K or m^2/s^2 K
- RADEA = 6371000. - Radius of the earth - m
- AKAP = 0.286 - Kappa, the ratio gas constant to specific heat capacity: R/Cp
- WW = 7.292E-5 - Angular velocity of the earth - rad/s

8.2.2 Numerical and run control

- BEGDAY = 0. - The day on which the run starts.
- TSPD = 64. - Number of timesteps per day.
- PNU = 0.015 - Time filter coefficient for the leapfrog scheme.
- TDISS = 0.5 - Dissipation timescale (days) for spectral hyperdiffusion.
- NDEL = 6 - Exponent of the gradient operator for spectral hyperdiffusion.
- T0 = 15*250. - Reference temperature (K) for the fifteen layers.
- NCOEFF = 0 - Maximum wavenumber of printed coefficients (only relevant for output text in channel 2).
- NLAT = 16 - Number of latitudes in printed gridpoint fields (only relevant for output text in channel 2).
- LSTRETCH = .F. - Legacy. Engages the original sigma level spacing from HS75.
- LLSD = .F. for T31 otherwise .T. - Activates "Lucy in the Sky with Diamonds" mode, in case you are hallucinating enough to believe that adding Land-Sea Drag will improve model performance. Doubles the drag over land in the lowest layer.

8.2.3 Forcing type

- LTRAIN = .F. - Switch to put the model run into training mode, to make successive forecasts from a sequence of initial conditions. Only used when generating a new forcing file.
- LFCE = .T. - Switch to apply spectral forcing from channel 13.
- LCYC = .F. - Switch to enable annual cycle and sequentially update forcing and reference fields.

8.2.4 Grid output for thermodynamic diagnostic variables

- LGRIDOUT2D = .T. - Controls output of 2-d grid diagnostics related to precipitation.
- LGRIDOUT3D = .F. - Controls output of 3-d grid condensation heating.

8.2.5 Counters

- RNTAPE = 100. - Variable for checking correct read/write of spectral history and restart files and also identifying type file: 100 for model output (restart); 200 for ERAi data; 300 for forcing files _fcm and _fbs and 400 for forcing anomaly files _fan.
- KOUNTH = 16 - Interval in timesteps at which history records are written.
- KOUNTR = 64000 - Interval in timesteps at which restart records are written.
- KOUNTREF = 16 - Interval in timesteps at which reference fields are updated from channel 16.
- KOUNTNUDGE = 16 - Interval in timesteps at which nudging fields are updated from channel 20.
- KOUNTFAN = 16 - Interval in timesteps at which forcing anomaly fields are updated from channel 15.
- KOUNTSSTC = 1948 - Interval in timesteps at which Climatological SST fields are updated from channel 18 (1948 corresponds to one average calendar month).
- KOUNTSST = 448 - Interval in timesteps at which SST fields are updated from channel 17 (448 corresponds to one week).
- KBEGYRSST = 0 - Start year for reading SST YYYY. Works with SST metadata.
- KBEGMNSST = 0 - Start month for reading climatological SST MM. Can also be used to force the model to start its annual cycle at the beginning of any month.
- KSTOPSST = 0 - Value of KOUNT at which to stop updating the SST and the SSTC and fix it at its current value. If set to zero this is inactive and SST and SSTC will update as normal. See also LPERSIST.

8.2.6 Tuneable parameters for simple physics

- TAUBL = 0.6667 - Average vertical diffusion timescale in the boundary layer in days.
- TAUBLEQ = 0.6667 - Value of TAUBL at the equator for runs with modified vertical diffusion in the tropics.
- PHITROPIC = 45. - Latitudinal extent over which TAUBL is modified by TAUBLEQ.
- TAUFT = 20. - Average vertical diffusion timescale above the boundary layer (free troposphere) in days.
- TAURC = 10. for T31 otherwise 12. - Timescale in days for Newtonian linear restoration of temperature throughout the atmosphere.
- SIGMAB = 0.8 - Sigma level that defines the top of the boundary layer for vertical diffusion coefficients.
- TAUCOND = 0.0625 (i.e. 90 minutes) - Timescale used to convert amount of water to be precipitated into a rain rate for the tendencies of T and Q in convection and large scale condensation schemes.
- TAUNUDGE = 0.25 - Timescale in days for linear nudging to nudge state read from channel 20.
- TAUSTAB = 0. - Timescale in days for uniform basic state stabilisation.
- NNTRUNC =15 - Wavenumber for spectral truncation of moisture variables in moist thermodynamic calculations, see also LTRUNC and LTRUNCQ.
- VIMCONTHR = 0. - Threshold for anomaly in vertically integrated moisture flux convergence compared to climatology required to trigger deep convection.
- BLSITHR = 0. - Threshold for anomaly in boundary layer static instability (bl vertical temperature gradient) compared to climatology required to trigger convection, see also LBLSI.
- PPTCAP = 15. - Maximum precipitation rate allowed for the deep convection scheme in mm/day. This limit is approached smoothly.
- PRHEATMAX = 0.35 - Sigma level at which normalised vertical profile for convective heating is maximum. This profile is used in the convection scheme and also in the response to SST anomalies.
- QGPFAC = 1.0 - Multiplying factor to boost continental humidity flux from the bottom boundary condition of the vertical diffusion scheme. Only applied over land. The sub-layer climatological value of specific humidity is multiplied by QGPFAC when MASK=1.

8.2.7 Forcing anomaly

- LFAN = .F. - Switch to apply spectral forcing anomaly from channel 15 on temperature or other variables.
- LPULSE = .F. - Causes the model to read a spectral forcing anomaly as a temporary pulse at the beginning of a run, with a squared sinusoidal temporal amplitude signature (normalised).
- KPULSE = 64 - Duration in timesteps of the anomaly forcing pulse.
- LSTAB = .F. - Enables damping that is the same for all degrees of freedom - used to stabilise basic states.
- LMODE = .F. - Enables modefinder, which successively rescales anomalies, for use in diagnosing unstable structures and growth rates from a basic state.
- LNUDGE = .F. - Enables nudging which reads a sequence of nudging data from channel 20.

8.2.8 Condensation scheme options

- LDEEP = .F. - Switch on the deep convection scheme.
- LLSR = .F. - Switch on the large scale in-situ condensation scheme (large scale rain).
- LCHX = .F. - Switch to set any negative values of forcing on humidity to zero. Also modifies temperature forcing. To be used in conjunction with explicit condensation and diabatic heating schemes.
- LTRUNC = .T. - Enables truncation of vertically integrated moisture flux convergence (VIMC) when used as a criterion for triggering deep convection - see also NNTRUNC.
- LTRUNCQ = .F. - Enables truncation of specific humidity when used for thermodynamic calculations - see also NNTRUNC
- LBLSI = .F. - Switch to enable a boundary layer static stability criterion in the decision to trigger deep convection.
- LPPTCAP = .T. - Enables limit to deep convective precipitation rate.

8.2.9 Sea surface temperature options

- LSST = .F. - Switch to read SST data from channels 17 and 18, which can be used to introduce heating anomalies.
- ISSTREAD = 1 - Instruction for how to interpret the SST data: 1 for full SST and 2 for SST anomaly.
- ISSTTRAN = 1 - Instruction for how to convert the SST anomaly into a precipitation anomaly: 1 for linear (one degree per day heating per degree of SSTA); 2 for nonlinear transfer to precipitation; 3 for direct reading of precipitation data; 4 to introduce the SST anomaly as a boundary condition in the vertical diffusions scheme.
- LSSTMASK = .T. - To apply the SST mask to the SST anomaly.
- LPERSIST = .F. - Will read one record of SST as directed by KBEGYRSST and KBEGMNSST if using SST metadata, otherwise just the first record. But will not update the SST after this, so the run will have a persisted SSTA.
- LREADMETA = .T. - Instructs the model to look for metadata in the SST file in channel 17.
- LOOPSST = .F. - Tells the model whether to return to the beginning of the file when it comes to the end of the SST data (channel 17) or SSTC data (channel 18). If set to false the model will stick on the last values read of both SST and SSTC if it gets to the end of either file.

8.2.10 Anomaly scaling factors

- SCALEFAN = 1. - Factor that multiplies anomaly forcing from a _fan file (channel 15). It can be used to double, halve, or reverse the sign of a forcing anomaly. It can also be used to scale down an anomaly (normally use a factor of 0.0001) to provide linear perturbation solutions (the anomalies in the solution can then be scaled back up for presentation).
- SCALESSTA = 1. - Factor that multiplies the SST anomaly directly.
- SCALEPPTA = 1. - Factor that multiplies the precipitation and thus the diabatic heating directly associated with an SST anomaly. If the transfer function is nonlinear this will have a different effect to SCALESSTA.
- SCALELHEAT = 1. - Factor that multiplies the specific latent heat of condensation. Normally = 1. Set it to zero if you want to force dry dynamics whilst retaining condensation processes (as in RUNTYPE=INTER) or to something else if you want to play.

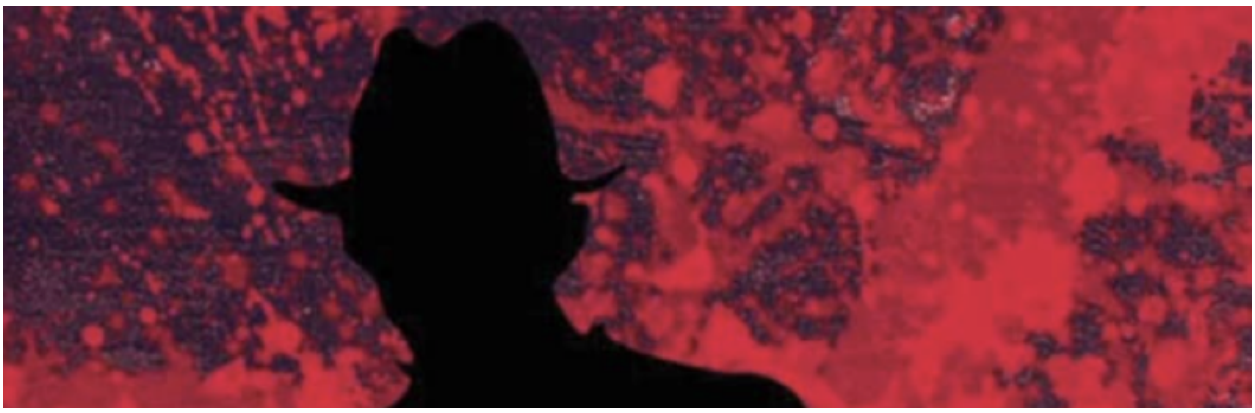
8.2.11 Zonal averaging options

- LZMFC = .F. - Takes the zonal mean of the forcing and reference data to provide zonally uniform forcing (but not for anomaly forcing).
- LZMIC = .F. - Takes the zonal mean of the initial condition.
- ISYM = 1 - Set this to zero to make these zonal mean calculations symmetric about the equator.
- IWAVE = 0 - Set this to 1 to add wavenumber 1 to these zonal mean calculations. Set it to 2 to add wavenumbers 1 and 2. Set it to 3 to add wavenumbers 1,2 and 3, etc.

8.2.12 Definition of regions for reading SSTs and nudging

- JNSST = 25 (reset to 19 for T31) - Sets northern latitude for the SST mask in a N-S Gaussian grid.
- JSSST = 40 (reset to 30 for T31) - Sets southern latitude for the SST mask in a N-S Gaussian grid.
- IWSST = 0 - Sets western longitude for the SST mask in a regular grid that starts on Greenwich.
- IESST = MG+1 (129 or 97) - Sets eastern longitude for the SST mask in a regular grid that starts on Greenwich.
- JNNDG = 25 (reset to 19 for T31) - Sets northern latitude for the nudging mask in a N-S Gaussian grid.
- JSNDG = 40 (reset to 30 for T31) - Sets southern latitude for the nudging mask in a N-S Gaussian grid.
- IWNDG = 0 - Sets western longitude for the nudging mask in a regular grid that starts on Greenwich.
- IENDG = MG+1 (129 or 97) - Sets eastern longitude for the nudging mask in a regular grid that starts on Greenwich.

9 Appendix D: Nightmare on Elm Street - The model subroutines.



The scope of this user guide is to give an indication of how things work without explaining every line of the code. The main body of the program has already been presented in Appendix A. Here the subroutines get a similar treatment, in the order they currently appear in the source code.

- INISET: Called just once at the beginning of the run to set up basic constants and default values of namelist variables. It then reads the SETUP namelist to override the defaults. SETUP has just five variables which are used for basic run types. The user modifiable code setup.f is then included. It uses the namelist SETUP, and overrides the defaults in the namelist INITIAL. Then INITIAL is read and this overrides both defaults and whatever has been set in the setup file. This way, what you see is what you get in the job script. The subroutine goes on to calculate basic indices, horizontal diffusion coefficients and non-dimensionalising factors and coefficients for calculating derivatives in from spectral coefficients. Finally it sets up all the non-dimensional time scales and associated rates for model parameterised processes.

- INIGAU: Called just once at the beginning of the run. Calculates Gaussian weights and latitudes.

- INISI: Called just once at the beginning of the run. Sets up the sigma levels from specified sigma half levels. Then sets up alpha weights used for calculating the geopotential in the gravity wave source term, see fig. A2.

- INIRES: Called just once at the beginning of the run (the name is historical, this routine is no longer used for restoration forcing). Contains some setup for the vertical profile of vertical diffusion coefficients. These are calculated according to the specified timescales and introduced into the profile PRDAMP. Also sets up the vertical profile for convective heating using input parameter PRHEATMAX. This profile is written into PRHEAT.

- INIVAR: Called just once at the beginning of the run. Initialises model variables. First single-level grid variables: masks, precipitation diagnostics and SSTs. Then multi-level grid variables. Then reads in the land-sea mask. Sets up a latitude-dependent function CD for the boundary layer vertical diffusion. Then defines regional masks for SSTs and nudging. Finally (and redundantly) initialises all spectral variables to zero, including tendencies.

- INIIC: Called at the start of each run, which is once per initial condition if we are in a training loop, but only once for a normal run. Initialises all spectral variables to zero, including tendencies. Initialises gridpoint output diagnostics to zero. Reads the initial condition from channel 10 and sets up the model year RMYR. Initialises previous timestep spectral variables to be the same as the initial condition.

- INITEND: Called at the beginning of the adiabatic part of the timestep and again at the beginning of the diabatic part. Initialises all spectral tendencies to zero.

- INIQCON: Called each time the forcing and reference state is updated and also at the beginning of the adiabatic part of the timestep. Initialises the spectral moisture flux convergence to zero.

- READFCE: Called at the start of each run and each time the forcing and reference state is updated. Reads forcing (channel 13) and reference states (channel 16). Takes their zonal mean if LZMFC enabled. Performs calculations to modify the forcing of humidity and temperature if LCHX is enabled (wet run). This entails wave to grid transformations using spectral workspace variables and gridpoint moisture and temperature variables QG and TG temporarily. The modified forcing is defined in grid space as the positive only source of Q and an adjusted heating for T. Wave to grid transformations are also made to store gridpoint values of the reference state. Moisture flux convergence is truncated to smooth it before finally going back to grid space to store a gridpoint reference value of the moisture flux convergence.

- READNUDGE: Called at the start of each run and each time the nudging state is updated. Reads the spectral nudging state from channel 20.

- READFAN : Called at the start of each run and each time the forcing anomaly is updated. Reads the spectral forcing anomaly from channel 15. Rewinds if at end of file.

- INISSTC: Called at the start of each run. Initialises grid climatological SST by reading from channel 18. Scrolls through the data until it picks up the correct record for the month KBEMNSST.

- READSSTC: Reads grid climatological SST from channel 18 each time it is updated. Will stick on the last record unless LOOPSST is enabled in which case it will rewind at the end of the file.

- INISST (MREMAIN): Called at the start of each run. Reads grid SST data from channel 17. If the SST data contains metadata the it initialises SST at correct year and month specified by KBEGYRSST and KBEGMNSST. Otherwise it just reads the first record of SST data. Calculates the right value of RMYR to pass the correct calendar date to the diagnostics. Passes back MREMAIN, the number of days to wait before reading again, to the main program to offset the counter for reading SST data.

- READSST Reads grid SST from `channel 17` each time it is updated. Will stick on the last record unless LOOPSST is enabled in which case it will rewind at the end of the file. Handles SST data with or without metadata.

- SPECPREP: Called just before the diabatic tendencies are calculated. Calculates smoothed values of Q and Q flux convergence by spectral truncation.

- SPECPOST: Called immediately after the diabatic tendencies are calculated. Sets global mean spectral coefficients of diabatic tendencies of vorticity, divergence and surface pressure to zero.

- FANDAMP: Called before the diabatic timestep. Adds in spectral tendencies from the forcing anomaly. If LPULSE is enabled a sin-squared temporal pulse is imposed on the anomaly amplitude at the beginning of the run, otherwise it is time-independent. If LSTAB is enabled an additional damping is applied equally to all spectral tendency coefficients.

- DIFUSE: Called before the diabatic timestep. Adds spectral tendency due to scale-selective hyperdiffusion.

- DSTEP: The diabatic timestep. Updates spectral variables with accumulated diabatic tendencies.

- FSTEP: Called immediately after the diabatic timestep. Updates spectral variables with basic empirical forcing tendencies.

- TFILT: Called at the end of the model timestep. Completes the time filter by modifying the previous values of the spectral variables, ZMI etc.

- HANAL: Called from numerous places during grid to wave transformations, notably the subroutine LTD and its derivatives. Used in direct Legendre transforms in to calculate Legendre coefficients for the latitudinal structure from variables in latitude-Fourier space. Different types of transform are used to obtain the fields or their derivatives for variables with differing symmetries.

- HANAL1: As HANAL but for single-level fields.

- HEXP: Called from numerous places during wave to grid transformations, notably the subroutine LTI and its derivatives. Used in inverse Legendre transforms in to calculate variables on Gaussian latitudes / Fourier space from spectral space. Different types of transform are used to obtain the fields or their derivatives for variables with differing symmetries.

- HEXP1: As HEXP but for single-level fields.

- LGNDRE: Called from INIGAU. Calculates Legendre polynomials and their derivatives as a function of latitude.

- LTD: Called from G2WA. Similar routines are called elsewhere during grid to wave transformations. Stands for Legendre Transform Direct. Transforms from latitude-Fourier space to grid space for a number of variables in order to calculate spectral tendencies for the first part of the time step. Includes multiple calls to HANAL to achieve this, with various requirements to take meridional derivatives as part of the transformation.

- LTI: Called from W2GA. Similar routines are called elsewhere during wave to grid transformations. Stands for Legendre Transform Indirect. Transforms from grid space to latitude-Fourier space for a number of variables in order to provide basic gridpoint fields for the calculation of tendencies in grid space. Includes multiple calls to HEXP to achieve this, with various requirements to take meridional derivatives as part of the transformation.

- MATINV: Called from INISI when setting up arrays for the semi-implicit scheme. Inverts a matrix.

- W2GA: Called from the adiabatic part of the timestep. Collects together the wave to grid operations in the adiabatic part of the time step, calling LTI for meridional (spectral to latitude-Fourier) and FFT991 for the zonal (latiude-Fourier to grid) part of the transformation. Returns grid point model variables.

- G2WA: Called from the adiabatic part of the timestep. Collects together the grid to wave operations in the adiabatic part of the time step, calling FFT991 for the zonal (grid to latiude-Fourier) and LTD for meridional (latitude-Fourier to spectral) part of the transformation. Returns spectral fields that will be further manipulated to find the spectral tendencies.

- W2GD: Called from the diabatic part of the timestep. As W2GA but for grid point variables needed in diabatic calculations, including instantaneous and reference moisture flux convergence.

- G2WD: Called from the diabatic part of the timestep. As G2WA but for grid point tendencies derived from diabatic calculations.

- ADVECTION: Called from the main program after wave-to grid transforms in the adiabatic part of the timestep. Calculates all the advective tendencies with variable names marked in red on fig. A1.

- SPDEL2: Called from LTD to find the EKE term in the divergence equation. Calculates the Laplacian (or inverse Laplacian) of a spectral variable.

- TSTEP: Called just after the first set of spectral transforms that evaluate the advective tendencies in the main program. This subroutine basically implements the equations in HS75 in spectral space. It has nested loops for hemisphere, zonal wavenumber m and meridional wavenumber n.

Briefly:

After setting up a few local variables, such as RCN=1/n(n+1) there are several calls to the matrix multiplier SGEMM. These evaluate variables like RMPA=TTR*G(the first part of the term in HS75 eq 17) and TMPG=TMI*G (the geopotential in HS75 eq 11).

Then a vertical integral to find :

```
D1 = 1/Cn DMI + DELT*[ PHI + (250/CT)*SPMI + 1/Cn DT(from advection) ]
              + DELT^2*[ TT(from advection)*G + (250/CT)*curlyP ]
```

where `curlyP = V.grad log p*` and is equal to -VP.

Then another matrix multiplication to get the left hand side of HS75 eq (17):

```
DT = D1 * BM1(IBM1+1)
```

Note that DT is used here to store the mean divergence, not the tendency !

Then a vertical sum to get the tendency of `log p*` (HS75 eq 4): VP(I1)=VP(I1)+DT(K)*DSIGMA(L)

Then another matrix multiplication to get TMPA=DT*TAU: to build the semi-implicit temperature tendency from HS75 eq (14).

This is added to TT, and then the rest of the subroutine basically adds in the advection, integrating with the time filter in triplets of lines like :

```
ZPAV=ZMI(I)
ZMI(I)=PNU21*Z(I)+PNU*ZPAV
Z(I)=ZPAV+DELT2*ZT(I)
```

And thus the model variables have been updated, and so have the pervious timestep versions, with just a time filter to close, which is done in the subroutine TFILT.

- WRSPS: Legacy routine to print out spectral coefficients

- SCALEDOWN: Called from the main program after the timestep has been completed. Activates the modefinder, which compares a quadratic norm in vorticity with its reference value. If this ratio exceeds 10, the difference between the model state and the reference state is scaled down by this ratio so the anomaly returns to its standard magnitude. This scaling is applied to both current and previous model state. If the ratio is less than 0.1 the anomaly is scaled up.

- LTIUVTPQ: Called during wave to grid transformations in the diabatic part of the time step. See LTI.

- LTDUVTPQ: Called during grid to wave transformations in the diabatic part of the time step. See LTD.

- DIABATIC: Called from the main program after wave-to grid transforms in the diabatic part of the timestep. This is the master subroutine for calculating diabatic tendencies in grid space. Each call to DIABATIC is for a vertical slice of grid data at north and south latitude pair JH (see Appendix A section 2). First all the grid point tendencies are initialised to zero. Then in sequence:
  - The nudging tendencies are evaluated directly.
  - The moisture flux convergence is calculated in PWATER and the deep convective tendencies are calculated in DEEPTEND.
  - The influence of SST is calculated with first another call to PWATER and then SSTTEND.
  - The vertical diffusion is evaluated in VDIFFTEND.
  - Land-sea drag is added directly.
  - Large scale rain tendencies are calculated in LSRTEND.

For each process, the gridpoint values of the model variables are updated as we go, so tendencies calculated for each process are influenced by the previous processes.

Finally the radiative cooling is applied directly and all the tendencies are added up to be passed back through to spectral space.

- SHUFFLE: Called from the main program for grid variables just after the diabatic tendencies have been evaluated in DIABATIC. Places the two-latitude slice data in a global gridpoint array.

- SHUFFLE1: As shuffle but for single level data.

- LTIUVTPQREF: As LTDUVTPQ for reference data.

- LTIUVTPQNDG: As LTIUVTPQ for nudging data.

- LTDQCONREF: As LTD for Q flux convergence reference data.

- LTDQTFCE: As LTD for forcing data.

- LTIQCONREF: As LTI for Q flux convergence reference data.

- LTIQTFC: As LTI for temperature and humidity forcing data.

- TRUNCATE(XTRUNC): Called from READFCE and SPECPREP an applied to specific humidity Q and its convergence. Truncates spectral variable XRUNC from NN to NNTRUNC by packing with zeros.

- TRUNCATE1(XTRUNC): As TRUNCATE for single level data - by default not used.

- ZMEAN(ZWK,DWK,TWK,SPWK,QWK): Called from INIIC and READFCE. Takes the zonal mean of the entire model state passed to the work arrays. Can also be used for forcing data. Global namelist options control symmetry: ISYM=1 for the full zonal mean and ISYM=0 for a zonal mean that is symmetric about the equator. IWAVE adds wavenumbers 1, 2 and 3 (see Appendix C).

- PWATER: Called from DIABATIC. Calculates column total moisture flux convergence and water content. Works on one latitude pair at a time. Evaluates: VIMCON: the vertically integrated moisture flux convergence, by summing the flux convergence QCONG over sigma layers and dimensionalising to mm/day. VIMCONTR: as VIMCON but with truncated moisture flux convergence. VIMCONREF: as VIMCON for the reference state. COLWATER: the total column water, by summing over sigma layers and dimensionalising to mm. COLSAT: for diagnostic purposes only, this is the value COLWATER would take if the column was at saturation at every level.

Some values are accumulated into 2-d grid arrays for diagnostic output: VIMC=VIMCON, the vertically integrated moisture convergence (mm/day) VICW=COLWATER, the column total water (mm) VISF=COLWATER/COLSAT, the saturated fraction. (note that 2-d values in mm could also be expressed in kg/m2).

- DEEPTEND(J, TGTDEEP, QGTDEEP): Called from DIABATIC. Returns deep convective tendencies for temperature and specific humidity. Works on one latitude pair at a time. See Chapter 4, section 4 for a description of the convection scheme.

This subroutine first sets the decision to trigger convection LTRIG as true, and then looks for reasons to set it to false. The first criterion depends on the difference VIMCONA between the smoothed vertically integrated moisture convergence VIMCONTR minus its reference value VIMCONREF. This difference must be positive and exceed a threshold for convection to be triggered.

The other optional criterion depends on vertical temperature differences in the lowest four layers of the model: specifically the mean of the bottom two layers minus the mean of the next two layers up. The "instability anomaly" BLSIA is the departure of this difference from its reference value. It must exceed a threshold for convection to be triggered.

If convection is triggered the column total rain falling in next TAUCOND period PPTTAU is calculated in mm as the convergence in mm/day times the period TACOND. PPTTAU is constrained not to exceed the total column water COLWATER and to be positive. It is also subject to a maximum value PPTTAUCAP, which is is constrained to approach asymptotically.

The value of PPTTAU is passed to PROFILEHEAT to evaluate the tendencies of specific humidity and temperature. The convective precipitation rate in mm/day associated with PPTTAU is stored for the current latitude pair for diagnostic 2-d output in PPTRATEDEEP.

- PROFILEHEAT(J, FR, PPTTAU, QGT, TGT): Called from DEEPTEND and SSTTEND. Uses the precipitation PPTTAU(mm) that is passed to it and the associated rate FR, and returns tendencies for temperature and specific humidity for deep convective heating. Works on one latitude pair at a time.

For a given column total rain PPTTAU(mm) falling in next tau period associated with rate FR, this subroutine calculates the tendency QGT of gridpoint specific humidity based on pro-rata reduction of specific humidity at rate FR at each level. This operation is carried out on smoothed humidity if the option LTRUNCQ is enabled.

It also sets the tendency TGT of temperature by spreading the total latent heating over the predetermined profile PRHEAT(L). Along the way it accumulates the 3-d diabatic heating rate CONDHEAT (deg/day) for diagnostic output.

These calculations are done level by level and the grid point values of temperature TG and specific humidity QG are updated as we go. This subroutine also updates the total precipitation rate for diagnostics PPTRATE (mm/day).

- SSTTEND: Called from DIABATIC. Returns tendencies for temperature and specific humidity associated with the response to an SST anomaly. Works on one latitude pair at a time. See Chapter 4, section 4c for a description of the way SST anomalies are implemented.

The purpose of this subroutine is to read SST data and, depending on the options set for ISSTRAN, calculate a precipitation rate anomaly associated with an anomaly in SST.

The subroutine starts by calculating the SSTA based on the nature of the variable SST. In the case ISSTREAD=1 the SSTA is calculated by subtracting SSTC from SST. Otherwise (ISSTREAD=0) SST is just used directly as the SSTA. At this point the SSTA can also be scaled by the factor SCALESSTA.

There different methods of for deducing a precipitation rate anomaly from the SSTA are set by ISSTRAN. The first three options translate the SST anomaly directly into a precipitation rates in mm/day (ISSTTRAN=1, 2, 3). The fourth passes the problem to the vertical diffusion scheme (ISSTTRAN=4). This is described in Chapter 4, section 4c.

For the first three cases the code actually uses the variable PPTTAU for rainfall in mm. And then it assumes that this amount of rain falls in one day by setting the rate FRSST appropriately. This is quite artificial and it is only done like this so that the subroutine PROFILEHEAT can be used to get the tendencies. For the fourth case (ISSTTRAN=4), this step is bypassed by setting PPTTAU=0. and the effect of the SSTA is calculated in the subroutine VDIFFTEND.

For the nonlinear transfer function, SSTC and the model divergence at σ=0.2, together with its reference value are used, along with a number of tuneable parameters. These parameters have not yet made their way into a namelist and this part of the code is work in progress. Once PPTTAU has been calculated, it is constrained by the land-sea mask to be only over the sea, and by the selective mask to either pick an ocean basin or cover the whole of the tropics. A constraint is added in the case that the convection scheme is being used, that the daily rainfall cannot exceed the column water total.

Finally PROFILEHEAT is called to set the tendencies of specific humidity and temperature associated with the SSTA: QGTSST and TGTSST as it does for the convection scheme, by spreading the heating into the deep profile and removing humidity pro-rata. But unlike the convection scheme call to PROFILEHEAT, the rate FRSST is now one day.

- VDIFFTEND(J,UGTVDIFF,VGTVDIFF,TGTVDIFF,QGTVDIFF): Called by DIABATIC. Returns tendencies for zonal and meridional wind, temperature and specific humidity due to vertical diffusion. Works on one latitude pair at a time. For a description of the vertical diffusion see Appendix A section 5 and fig. A4.

This subroutine sets up damping rates due to vertical diffusion according to the damping rate profile PRDAMP, which is defined on sigma layer boundaries. DM and DP refer to the layer boundaries above and blow a given sigma level. Variables such as UGM and UGP are assigned to represent the values in fictitious layers above and below the vertical domain so that different boundary conditions can be imposed. We choose damped boundary conditions with these variables fixed at the reference values at the adjacent (top and bottom) levels.

It is at this point that the anomalous boundary flux due to SSTAs is added in the case that this is selected by setting ISSTTRAN=4. The SSTA is defined again as a local variable like in SSTTEND. Anomalies in temperature and humidity (the difference in saturation value) are calculated and added to the bottom boundary condition. The optional boost in land-surface humidity flux by the factor QGPFAC is also implemented here.

A simple centred difference is applied to find diffusive flux convergence and this is assigned to the tendencies of momentum, temperature and specific humidity. Only temperature and specific humidity grid variables are updated as there is no need to keep the momentum up to date.

- LSRTEND(J,TGTLSR,QGTLSR): Called by DIABATIC. Returns height dependent tendencies for temperature and specific humidity associated with resolved condensation (large scale rain). Works on one latitude pair at a time. See Chapter 4, section 4c for a description of the large scale rain scheme.

This subroutine calculates saturated specific humidity QSAT at every level at a given grid point and compares it with the local specific humidity (or its smoothed value QGTR if LTRUNCQ is enabled). The supersaturation QDIFF is the difference between the two, and it is used directly to calculate the local specific humidity tendency QGTLSR using the rate FRCOND derived from the timescale TAUCOND. The associated latent heating rate is applied to the local temperature tendency TGTLSR. A limit is set to the maximum value of supersaturation that can be rained out at any given level. It is current;y set in the code as QDIFFCAP=50./16. * 1.e-4. For a standard value of TAUCOND=0.0625 (i.e. 1.5 hours) this means that the maximum precipitation rate possible (provided the limit is hit at every level) is 50 mm/day.

For diagnostic output, the rain rate PPTRATE (mm/day) is updated level by level. Note that this 2-d grid variable is the sum of all precipitation in the model. It is updated directly from this LSR scheme and also from the deep convection and SSTA schemes in PROFILEHEAT. Likewise the 3-d grid diagnostic variable for condensation heating CONDHEAT (deg/day) is updated level by level.

Finally the grid values of temperature TG and specific humidity QG are updated with the LSR tendencies.

- CUSTOMMASK(JN,JS,IW,IE,WRKMASK): Called by INIVAR. Returns a 2-d mask for a region defined by latitude and longitude boundaries JN,JS,IW,IE.

A box is bounded by JN to the north, JS to the south, IW to the west and IE to the east. These are Gaussian latitudes and equally spaced longitudes with the first point on Greenwich. The box is filled with the value 1, so information is retained inside the box when the mask is applied. Outside this box the values are zero so this area is masked out. On the boundaries the value is 0.5 for a smooth transition from unmasked to masked areas.

This subroutine is called to define the SST mask RMASKSST: currently four options, the full tropics, and the Pacific, Atlantic and Indian tropical oceans (these three masks are contiguous). The boundaries of these masks for T42 and T31 are given in the include file setup.f.

This subroutine is also called to define the nudging mask RMASKNDG: the default values included in `setup.f` are for the full tropics.

10 Appendix E: DREAM Warriors - Publications with DREAM and its Earlier Variants.

- Hai Lin, Bin Yu, and Nicholas M.J. Hall, 2022: Origin of the Warm Arctic–Cold North American Pattern on the Intraseasonal Time Scale. J. Atmos. Sci., 79, https://doi.com/10.1175/JAS-D-22-0013.1
- Braga, Hugo, Tercio Ambrizzi and Nicholas M.J.Hall, 2022: Relationship Between Interhemispheric Rossby Wave Propagation and South Atlantic Convergence Zone during La Niña Years. Int. J. Climatol. DOI: 10.1002/joc.7755
- Torres, Victor M., C. D. Thorncroft and N. M. J. Hall, 2021: Genesis of Easterly Waves over the Tropical Eastern Pacific and the Intra-Americas Sea. J. Atmos. Sci., 78, 3263–3279.
- Hall, N.M.J., H-H Le and S. Leroux 2020: The extratropical response to a developing MJO: Forecast and climate simulations with the DREAM model. Climate Dyn., https://10.1007/s00382-020-05299-y
- Hall, N.M.J., S. Leroux, and T. Ambrizzi, 2019: Transient contributions to the forcing of the atmospheric annual cycle: A diagnostic study with the DREAM model. Climate Dyn. https://doi.org/10.1007/s00382-018-4539-y
- Lin, H., and G. Brunet 2018: Exratropical response to the MJO: nonlinearity and sensitivity to initial state. J. Atmos. Sci., 75, 219-234. https://doi.com/10.1175/JAS-D-17-0189.1.
- Lin, H., R. Mo, F. Vitart, and C. Stan, 2018: Eastern Canada flooding 2017 and its subseasonal predictions, Atmosphere-Ocean, https://doi.com/10.1080/07055900.2018.1547679.
- Wu, Z., P. Zhang, H. Chen, and Y. Li, 2016: Can the Tibetan Plateau Snow Cover influence the interannual variations of Eurasian Heat Wave Frequency? Climate Dyn., 46, 3405-3417.
- Wu, Z., and P. Zhang, 2015: Interdecadal Variability of the mega-ENSO-NAO Synchronization in Winter. Climate Dyn., 45, 1117-1128.
- Yu, B., and H. Lin, 2015: Tropical atmospheric forcing of the wintertime North Atlantic Oscillation. J Climate. 29, 1755-1772.

- Wu, Z., J. Dou, and H. Lin, 2014: Potential Influence of the November-December Southern Hemisphere Annular Mode on the East Asian Winter Precipitation: A New Mechanism. Climate Dyn., 44, 1215-1226.
- Hall, N.M.J., H. Douville, and L. Li, 2013: Extratropical summertime response to tropical interannual variability in an idealized GCM. J. Climate, 26, 7060-7079.
- Yu, B., and H. Lin, 2013: Tropical American-Atlantic forcing of austral summertime variability in the southern annular mode. Geophy. Res. Lett. , 40, 5, 943-947.
- Lin, H., and Z. Wu, 2012: Indian Summer Monsoon Influence on the Weather in the North Atlantic-European Region. Clim. Dyn., 39, DOI:10.1007/s00382-011-1286-8, 303-311.
- Wu, Z., and H. Lin, 2012: Interdecadal Variability of the ENSO-North Atlantic Oscillation Connection in boreal summer. Quart. J. Roy. Meteor. Soc., 138, 1668-1675.
- Wu, Z., J. P. Li, Z. Jiang, J. He, and X. Zhu, 2012: Possible effects of the North Atlantic Oscillation on the strengthening relationship between the East Asian summer monsoon and ENSO. Int. J. Climatol., 32, 794-800.
- Wu, Z., J. P. Li, Z. Jiang, and T. Ma, 2012: Modulation of the Tibetan Plateau Snow Cover on the ENSO Teleconnections: From the East Asian Summer Monsoon Perspective. J. Climate, 25, 2481-2489.
- Wu, Z., Z. Jiang, J. P. Li, S. Zhong, and L. Wang, 2012: Possible Association of the western Tibetan Plateau Snow Cover with the Decadal to Interdecadal Variations of Northern China Heatwave Frequency. Climate Dyn., 39, 2393-2402.
- Yu, B., and H. Lin, 2012: Tropical/Extratropical forcing on wintertime variability of the extratropical temperature and circulation. Clim. Dyn., https://doi.com/10.1007/s00382-012-1367-3. Zhang, X., H. Lin, and J. Jiang, 2012: Global response to tropical diabatic heating variability in boreal winter, Adv. Atmos. Sci., 29, 369-380.
- He, J., H. Lin, and Z. Wu, 2011: Another look at influences of the Madden-Julian Oscillation on the wintertime East Asian weather. J. Geophys. Res., 116, https://doi.com/10.1029/2010JD014787.
- Leroux, S., N.M.J. Hall, and G.N. Kiladis, 2011: Intermittent African Easterly Wave activity in a dry atmospheric model: influence of the extratropics. J. Climate, 24, 5378-5396.
- Wu, Z., J. P. Li, Z. Jiang, and J. He, 2011: Predictable climate dynamics of abnormal East Asian winter monsoon: once-in-a-century snowstorms in 2007/2008 winter. Climate Dyn., 37, 1661-1669. Lin, H., G. Brunet, and R. Mo, 2010: Impact of the Madden-Julian Oscillation on wintertime precipitation in Canada. Mon. Wea. Rev., 138, 3822—3839.
- Janicot, S., F. Mounier, N.M.J. Hall, S. Leroux, B. Sultan, and G. N. Kiladis, 2009: The dynamics of the West African monsoon, part IV: Analysis of 25-90-day variability of convection and the role of the Indian monsoon. J. Climate, 22, 1541-1565.
- Leroux, S., and N.M.J. Hall, 2009: On the relationship between African Easterly Waves and the African Easterly Jet, J. Atmos. Sci., 66, 2303-2316.
- Lin, H., 2009: Global extratropical response to diabatic heating variability of the Asian summer monsoon. J. Atmos. Sci. 66, 2693-2713.
- Lin, H., J. Derome, and G. Brunet 2009 The Nonlinear Transient Atmospheric Response to Tropical Forcing. J. Climate, 20, 5642-5665.
- Wu, Z., B. Wang, J. P. Li, and F.-F. Jin, 2009: An empirical seasonal prediction model of the East Asian summer monsoon using ENSO and NAO. J. Geophys. Res., 114, D18120, https://doi.com/10.1029/2009JD011733.
- Tang, Y., H. Lin, and A. M. Moore, 2008, Measuring the potential predictability of ensemble climate predictions. J. Geophys. Res., 113, D04108, https://doi.com/10.1029/2007JD008804.
- Thorncroft, C.T., N.M.J. Hall, and G.N. Kiladis, , 2008: Three dimensional structure and dynamics of African Easterly Waves, part III: Genesis. J. Atmos. Sci., 65, 3596-3607.
- Lin, H., G. Brunet, and J. Derome 2007: Intraseasonal variability in a dry atmospheric model, J. Atmos. Sci., 64, 2442-2441.
- Tang, Y., H .Lin, J. Derome, and M.K. Tippett, 2007, Measuring the reliability of seasonal predictions for the Arctic Oscillation. J. Climate, 20, 4733-4750.

- Hall, N.M.J., G.N. Kiladis, and C.T. Thorncroft, 2006: Three dimensional structure and dynamics of African Easterly Waves, part II: Dynamical modes. J. Atmos. Sci., 63, 2231-2245.
- Derome, J., H. Lin, and G. Brunet, 2005: Seasonal forecasting with a simple general circulation model: predictive skill in the AO and PNA. J. of Climate, 18, 597-609.
- Lin, H., and J. Derome, 2004: Nonlinearity of extratropical response to tropical forcing. J. Climate, 17, 2597-2608.
- Lin, H., J. Derome, R. J. Greatbatch, K. A. Peterson, and J. Lu, 2002: Tropical links of the Arctic Oscillation. Geophys. Res. Lett., 29(20), 1943-1947
- Peterson, A. R. Greatbatch, J. Lu, H. Lin, and J. Derome, 2002: Hindcasting the NAO using diabatic forcing of a simple AGCM. Geophys. Res. Lett., 29, https://doi.com/10.1029/2001GL014502.
- Hall, N.M.J., J. Derome, and H. Lin, 2001: The extratropical signal generated by a midlatitude SST anomaly. Part 1: Sensitivity at equilibrium. J. Climate, 14, 2035-2053.
- Hall, N.M.J., H. Lin, and J. Derome, 2001: The extratropical signal generated by a midlatitude SST anomaly. Part 2: Influence on seasonal forecasts. J. Climate, 14, 2696-2709.
- Hall, N.M.J., 2000: A simple GCM based on dry dynamics and constant forcing. J. Atmos. Sci., 57, 1557-1572.
- Hall, N.M.J. and J. Derome, 2000: Transience, nonlinearity and eddy feedback in the remote response to El Niño. J. Atmos. Sci., 57, 3992-4007.
- Hall, N.M.J., and P.D. Sardeshmukh, 1998: Is the time mean Northern Hemisphere flow baroclinically unstable ? J. Atmos. Sci., 55, 41-56.

11 How to cite this manual

Nick MJ Hall, & Stephanie Leroux. (2023). Dream user manual v8.4 (v8.4.0). Zenodo. https://doi.org/10.5281/zenodo.8414525

You can browse the latest version of this manual on-line (https://dreamusermanual.readthedocs.io/en/latest/), or download it in PDF format by clicking on the "Read the Docs" menu at the bottom left of this page.