9.1 *Dynamics in the zero-D Energy-balance climate model*

We'll now make the zero-D energy-balance climate model (very) slightly more interesting, or at least, (very) slightly more realistic. The time-dependent behavior of the initial version of the energy balance model is trivial. In fact: there isn't any. The system is always in equilibrium as constructed. Why? No thermal inertia – i.e. nothing in the system defined so far has been given any heat capacity and the outgoing (longwave) energy flux is always assumed to be in exact equilibrium with the incoming (shortwave) flux. So we need to add an ocean, or rather: a box (a *variable* in the **MATLAB** code) to store the heat content, or temperature, of the ocean, and update this (temperature) in the event of there being any imbalance between gain and loss of energy at the surface of the Earth.

The science behind the new model is based directly on the basic energy balance equations you had before, except this time you are not going to assume the 2 equations equal (and solve for T) but employ them directly. Instead, you are going to calculate the net energy gain (or loss) over a given interval of time and use the specific heat capacity of a substance (assuming water here)[1] to link the energy change to a temperature change (see Box). This will be the basis of the 'dynamics' of the climate model and will dictate how quickly the mean surface temperature responds to any imbalance in loss vs. gain of energy. You can also assume the following:

- The average mixed layer depth of the ocean is 70 m.
- The average fraction of the Earths surface that is ocean is 0.7.

(both from *Henderson-Sellers* [2014]). You'll also need to know:

- The specific heat capacity of water.

but you can find this out for yourself ...[2] Note that you do not need to know e.g. the radius of the Earth as we are constructing the model on a global average per m^{-2} basis as before.

The form of the program is shown schematically in Figure 9.1 and you'll need to create yourself a new script (`scr_1`) to make this. Much of this and the main sections of code should look familiar. Break the code down into logical sections. Start by defining any constants you need, as well as parameter values. For the time loop, we are going to start off with a fixed total duration and a fixed time step (a little later we'll relax these constraints). And to make things really simple to start – assume a 100 year duration (starting at $T = 1.0$) and a time increment $\Delta T = 1.0$. So you are not even going to need to initialize and update a loop counter in the code! In the loop itself, you firstly need to calculate the energy imbalance (assuming there is

Specific Heat Capacity
According to wikipedia: "*An object's* [or here: ocean] *heat capacity (symbol C) is defined as the ratio of the amount of heat energy transferred to an object and the resulting increase in temperature of the object:*"

$$C = \frac{Q}{\Delta T}$$

where Q is the (change in) energy (so could equally be written ΔQ if you prefer) and ΔT the associated change in temperature. Units are:

- C — JK^{-1}
- ΔT — K
- Q — J

[1] Once again – be very careful with the units. Or all will be lost ...

[2] Be careful to end up with CONSISTENT units!

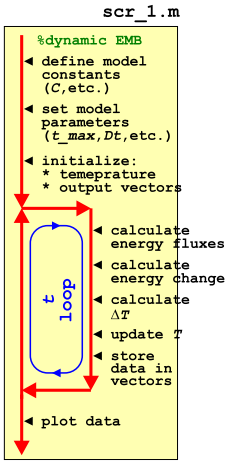

Figure 9.1: Schematic of the script for the basic dynamic EBM

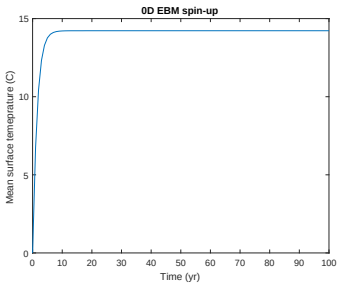

Figure 9.2: 100 yr spin-up of the basic EBM.

one) – remembering that the energy fluxes are in units of Wm^{-2}, i.e. $Js^{-1}m^{-2}$, so you'll need to take the time-step duration into account and find the number of J of heat gain/loss during that time (in s)– then use this to update the temperature of the mixed layer ocean.[3] Then after the loop, plot something helpful at the end.

If successful, you should see something similar to (actually, identical to) Figure 9.2 (assuming a 1 yr time-step).

Next, you are going to play a little with the time-step in the model. So rather than a simple loop from 1 to 100 (years) with an increment of 1, you are going to generalize the increment as Δt. If `dt` is your parameter representing the increment in time (presumably, conveniently defined hear the start of the code)[4], and `max_t` the maximum time (here: 100 years) (also conveniently defined near the start of the code?), then:

```
% start of time-stepping loop
for t = 1:dt:max_t,
    % SOME CODE GOES HERE
end
```

Now however, you will need to crete yourself a loop counter in order to store the results (for subsequent plotting), as because `dt` will not necessarily be an integer, you will not be able to use `a` to index your data storage vector (/array). The modification needed is only minor however – see Figure 9.3. The only slight complication is in knowing the size of the output vectors, assuming that you have created them (using `zeros`) up-front in the code (and as per the Figure 9.1 schematic), rather than growing the vectors as the loop progresses (see earlier). Initially, you would have been able to simply write e.g.

```
data_time  = zeros(100);
data_T = zeros(100);
```

One strategy is simply to pick a number larger than you think the number of times the loop will execute. The downside being that you might create a vast array with only a small portion of it ever being used. Better in this example would be to append to the vectors as the loop progresses and not attempt to define them beforehand (i.e. Figure 9.1 rather than Figure 9.3).

By playing around with different parameter values for Δt, you should discover that some care has to be taken with the choice of time-step duration, e.g. Figure 9.4 has a time-step of 3.5 years, which clearly is on the verge of going doolally. [5]

So far, so far from exciting – you have been simply time-stepping the model to equilibrium, for which there was an analytical solution anyway (with ocean heat capacity irrelevant to this). However, it should be apparent that it takes some years (how many) for the system to reach equilibrium. This would have important implications for

[3] It is much easier and less prone to bug, if you do this in two stages. You could even split things into four:

1. Incoming energy flux.
2. Outgoing energy flux.
3. Net energy change (per m^2) at the Earths surface.
4. Update surface temperature.

[4] Don't forget to convert `dt` into units of s when you use it in the energy calculation.

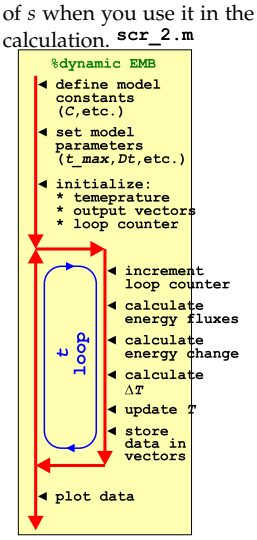

Figure 9.3: Schematic of the script for the basic dynamic EBM – now with added loop count(!)

[5] For practice (fun!?), you could turn the script into a function. Make two parameters as inputs: (1) the total simulation duration, and (2) the time-step, both in units of yr.

Doolally
Mad, insane, eccentric.

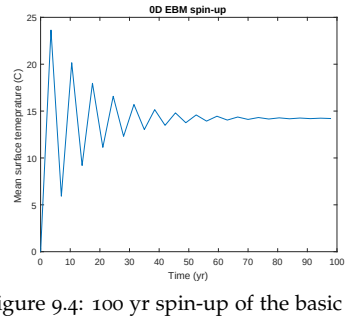

Figure 9.4: 100 yr spin-up of the basic EBM, but with a poor choice of time-step ...

a (real world) system in which the one of the terms in the radiative balance equation changes relatively rapidly (or on a time-scale comparable to the adjustment time of the system). The concentration of CO_2, and radiative forcing due to the 'greenhouse effect', is just such an example.

A follow-on Example to this, takes the time-stepping (dynamic) zero-D EBM (`scr_1`) and drives it with a time history of atmospheric CO_2 concentration (technically: mixing ratio) data.

First off: check out the CO_2 radiative forcing (Greenhouse Effect) Box. This will guide you as to how you are going to modify your energy budget (within the time-stepping loop) – basically, you are simply adding a 3rd term (and a second incoming term) to the heat budget. Test the model first with a fixed, assumed CO_2 concentration and check that the mean surface temperature responds in a reasonable way.[6,7]

The first thing you are going to do, is to take your previous script (`scr_1` or `scr_2`, it does not really matter) and turn it into a function, with a single input (`co2`) and no output. The passed parameter `co2` (or call it something different) will be the concentration of CO_2 in the atmosphere in μatm (equivalent to units of ppm for 1 atmosphere total pressure). You'll then need to edit the calculation of the energy loss/gain by incorporating the greenhouse effect term. The code looks not much different from before – Figure 9.5).

From your previous experiments, you should have determined what value the equilibrium temperature ended up as. You should make this your new initial condition for the planetary temperature and set the appropriate parameter. (If you don't, the results of all your subsequent experiments will be dominated by the climate system adjusting from your initial condition rather than necessarily responding to whatever perturbation you have applied (/experiment carried out).) Having done this, explore the effect of calling your function and passing values for CO_2 different from $278ppm$ ($278\ \mu atm$). For reference:

- Peak of last glacial — $\sim 190ppm$
- Pre-industrial — $278ppm$
- Current — $\sim 400ppm$
- End of century — $\sim 900ppm$
- Cretaceous — $\sim 834 - 1112ppm$(?)

or try other values.

The Greenhouse Effect

The effect of changing CO_2 concentrations on the global energy budget is typically written in terms of a virtual (long-wave) radiation flux applied at the top of the atmosphere. The flux anomaly, ΔF, as a function of CO_2 concentration (technically: mixing ratio) (CO_2) relative to a reference (pre-industrial) concentration (typically: $CO_{2(0)} = 278ppm$) can be approximated:

$$\Delta F = 5.35 \cdot ln(\frac{CO_2}{CO_{2(0)}})$$

The complete basic EBM energy budget now looks like:

$$F_{in} = \frac{\alpha \cdot S_0}{4} + 5.35 \cdot ln(\frac{CO_2}{CO_{2(0)}})$$

$$F_{out} = 0.62 \cdot \sigma \cdot T^4$$

[6] What is 'reasonable'? Well, you could conduct a pair of experiments – one in which you do not modify CO_2, and one in which your double it. The IPCC and there (now) five Assessment reports have much to say about the climate system response to a doubling of CO_2. So you can conduct a reality check on your model based on existing and widely available climate sensitivity information.

[7] By way of reference: assume that the pre-industrial concentration (mixing ratio) of CO_2 in the atmosphere ($CO_{2(0)}$) is 278 ppm.

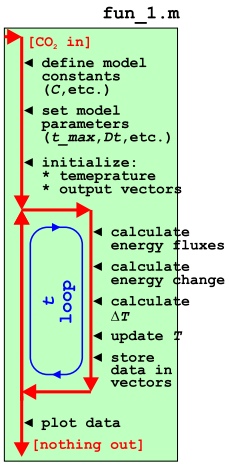

Figure 9.5: Schematic of the dynamic EBM as a function and with the CO2 concentration passed in.

The final Example involves loading in a CO_2 data-set and driving the dynamic zero-D EBM with a changing concentration of CO_2 in the atmosphere.

Go back again to your first dynamic EBM program (`scr_1`). The new version (`scr_3`) will be similar (Figure 9.6). You need to:

1. Add in code to load in the CO_2 dataset. You are going to use the ice-core derived record from week #1 (`etheridge_etal_1996.txt`).
2. From the resulting data array – determine the minimum and maximum years and the total length (number of rows) of the data. All these values might usefully be stored in variables in your code.
3. Create results vectors of the same length. Create one vector for each of: year, CO_2 value, temperature. (Create a single array instead if you prefer.)
4. Edit the time loop such that it runs from the minimum to maximum year (with a time-step of 1 year).
5. In the loop – take the CO_2 value from that year and use it in the calculation of the radiation balance.
6. Also in the loop – save the current year, CO_2 value, and associated calculated temperature. Be careful that indexing of arrays in **MATLAB** always starts at a value of 1. You will either need to derive an index from the current year, or add a loop counter (it is simple to do the former and it takes less lines of code).

When you have this working you should get something like Figure 9.7 (but note that this was done with not quite the same CO_2 dataset ...). If you want to be fancy you can add a horizontal line indicating the pre-industrial equilibrium solution (using `line`).

Finally, the lagged behavior of the climate system (as encapsulated in your EBM) is maybe not obvious as the forcing (CO_2) is varying. Common in model experiments and characterization, is to create artificial and deliberately simplified forcings and perturbations, so as to more readily diagnose the response time and characteristics of a system. Crete an artificial CO_2 data-set, spanning the same time interval as the real data, and at the same frequency, but substitute an idealized CO_2 forcing in which CO_2 stays constant (at 278 ppm) up until year 1999, then at year 2000, increases to 400 ppm, and stays there. The result of such an experiment should look like Figure 9.8.

Other common model scenarios are linear ramps (up, and/or down) and compound increases, such as a 1% per year increase in the concentration of CO_2 (each and every year) starting ca. 1960.

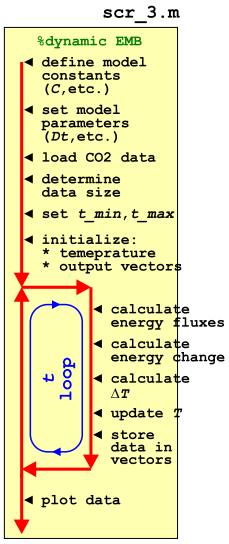

Figure 9.6: Schematic of the dynamic EBM driven by a history of CO2 (read in from a file).

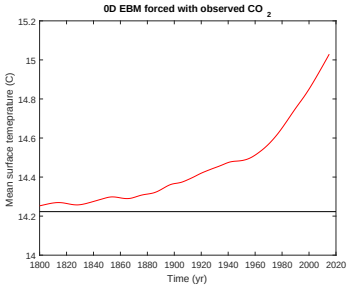

Figure 9.7: Transient EBM response to observed changes in atmospheric CO2. For reference, the pre-industrial equilibrium global temperature is shown as a horizontal black line.

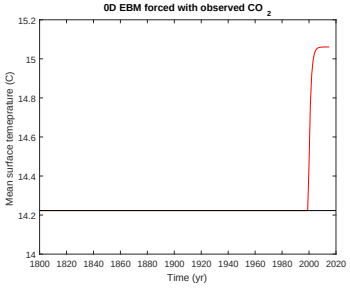

Figure 9.8: Transient EBM response to (fake) changes in atmospheric CO2.