HW3: 1d energy balance model

November 28, 2022

Let T_0 be an initial (meridionally-averaged) temperature distribution, set to be $T_0(x) = 10°C$ for all $x \in [0,1]$, the meridional coordinate. We assume that the temperature $T(t,x)$ of the system evolves according to

$$\partial_t T = QS(x)a(x) - (A + BT) + \mathbb{D}[T] \tag{1}$$

$$\text{subject to } T(t,x)\Big|_{t=0} = T_0 \tag{2}$$

$$(1-x^2)\frac{\partial T}{\partial x}(t,x)\Big|_{x=0,1} = 0. \tag{3}$$

where Q is the solar constant, $S(x)$ the solar radiation distribution, and $a(x)$ the co-albedo; together they comprise a term modelling the incoming energy radiated from the sun. The term which representing the outgoing radiation from earth, modeled by the Stefan-Boltzmann law, is linearized as $\sigma T^4 \sim (A + BT)$ for some parameters A and B. In atmospheric temperature ranges like $[-10, 40]°C$, this linearization is a good approximation. Furthermore, the diffusion operator $\mathbb{D}$ takes the form

$$\mathbb{D}[f](x) \doteq D * \frac{\partial}{\partial x}\left((1-x^2)\frac{\partial f}{dx}\right). \tag{4}$$

To this second-order differential operator, we might consider the associated eigenvalue problem

$$\mathbb{D}[f](x) = -\lambda f(x). \tag{5}$$

Note that we have taken the following special forms for the solar radiation and albedo terms:

$$S(x) = S_0 + S_2 P_2(x) \tag{6}$$

$$a(x) = a_0 + a_2 P_2(x) \tag{7}$$

subject to the constraint

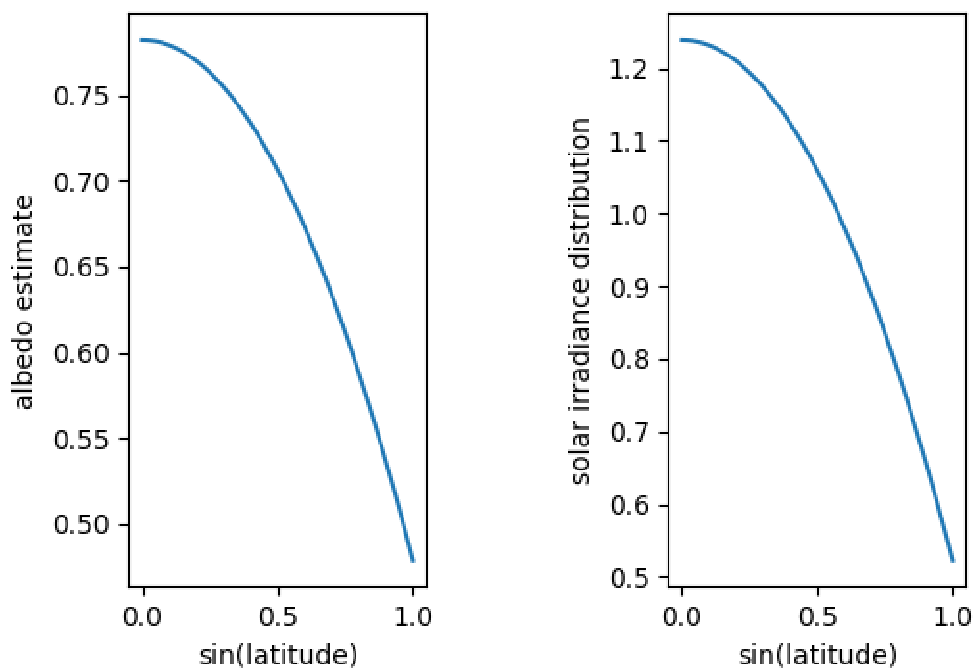
$$\int_0^1 S(x) = 1. \tag{8}$$

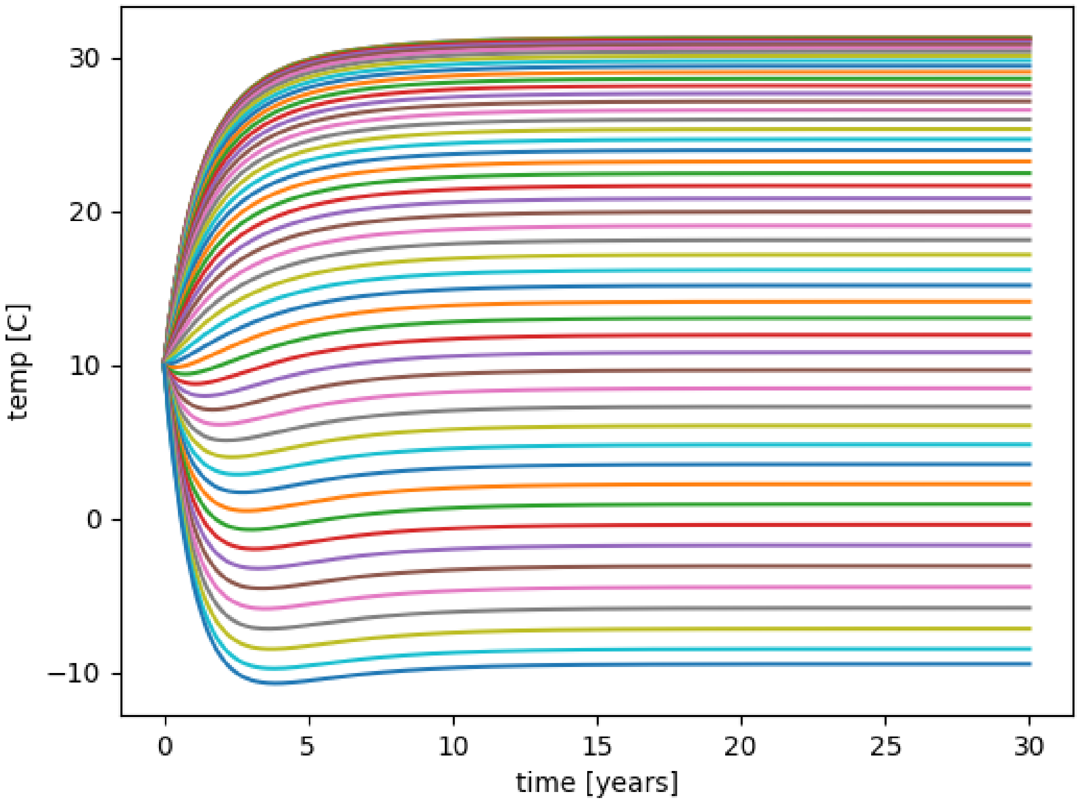
We think of S as a distribution of solar irradiance over the meridional coordinate. The co-albedo coefficient $a(x)$ likewise has some spatial dependence. In this way, with $QS(x)a(x)$ the average power per meter squared, the power density at x is modulated by the distributions of solar irradiance as well as the co-albedo. The net difference between these effects forces the heat equation above. Plots of both as a function of $\sin(\theta)$ with $\theta \in [0, \frac{\pi}{2}]$ are below.

0.1 1d heat equation with ODEINT

Here we describe some of the code `in energy_bal1d.py`. In particular, we focus on the diffusion operator, in which the boundary conditions of the model are encoded. At the equator grid-point, $i = 0$, we use the boundary condition (**??**) to rewrite diffusion operator. We invoke a GHOST POINT at $i = -1$, where the no-flux condition, implemented via a leapfrog difference, is enforced:

$$\frac{\partial T}{\partial x}\Big|_{x=0} \sim D_h^{leap}[T][0] \doteq 1/(2h)(T[1] - T[-1]) = 0. \tag{9}$$


albedo estimate
0.75
0.70
0.65
0.60
0.55
0.50
0.0
0.5
1.0
sin(latitude)
solar irradiance distribution
1.2
1.1
1.0
0.9
0.8
0.7
0.6
0.5
0.0
0.5
1.0
sin(latitude)



Therefore, this boundary condition introduces the constraint $T[0] = T[2]$. We may then use to re-write the only non-degenerate portion of the discretized diffusion operator $\mathbb{D}_h[T]$ at $i = 0$, i.e., the second-order term of the central-difference operator approximating the term $\frac{\partial^2 T}{\partial x^2}$:

$$\mathbb{D}_h[T][0] = D(T[1] - 2T[0] + T[-1])/(h^2) = 2D(T[1] - T[0])/(h^2) \tag{10}$$

At the pole, since $x[N] = 1$ here, the only non-degenerate part of the diffusion operator is the first-order term:

$$\frac{\partial T}{\partial x}(t,1) = (1 - 1^2)\frac{\partial^2 T(t,1)}{\partial x^2} - 2\frac{\partial T(t,1)}{\partial x} = -2\frac{\partial T(1)}{\partial x} \tag{11}$$

which is implemented via a one-sided difference of $O(h)$ accuracy. The interior updates are discretized by a combination of a central-difference scheme for the second-order term and a leap-frog difference scheme for the first-order term.

A plot of the solution found with ODEINT is included below, using a grid resolution of $n = 100$, over a time period of 30 years with monthly time steps:

We want to benchmark this solution against the analytic solution. The code for this is in `analytic_benchmarking.py`. This is described in the third section.

0.2 Implicit euler scheme

Discretizing first in space, we consider the first-order (in time) ODE

$$\frac{d}{dt}T(t, x[i]) = QS(x[i])a(x[i]) - (A + BT)(x[i]) + \mathbb{D}_h[T]_i(x). \tag{12}$$

We now have, for each grid point $x[i]$, $i = 0, \dots N$ an ODE to solve given initial condition $T_0(x[i]) = 10$. We discretize this according to the implicit Euler, for the stability properties of its solutions. With k denoting a time grid-point and i denoting a spatial grid-point, (12) becomes

$$(T[k+1,:] - T[k,:]) = \frac{dt}{c_w}\left(QSa - (A + BT[k+1,:]) + \mathbb{D}_h[T[k+1,:]]\right) \tag{13}$$

To construct the matrix $\mathbb{D}_h[\cdot]$, recall that the i-th column of a matrix A in standard basis coordinates is given by $A[e_i]$. So given the operator defined in the function `diffusion`, we define another function which outputs the matrix $(A[e_i])_i$, with A here standing for the right-hand-side of (12). Then:

$$\left(I + \frac{dt}{c_w}\left(B - \mathbb{D}_h\right)\right) T[k+1,:] = T[k,:] + \frac{dt}{c_w}\left(QSa - A\right) \tag{14}$$

For $\frac{dt}{c_w}\left\|B - \mathbb{D}_h\right\|_{op} \leq \frac{1}{2}$ we may invert the matrix on the right-hand side yielding:

$$T[k+1,:] = \left(I + \frac{dt}{c_w}\left(B - \mathbb{D}_h\right)\right)^{-1}\left(T[k,:] + \frac{dt}{c_w}\left(QSa - A\right)\right). \tag{15}$$

Thus, we may write an advance mapping $T[k+1,:]$ given $T[k,:]$ and explicit forms for the matrices in the equality above; this is done in the function `advance` in the file `energy_bal1dIE.py`. For the first matrix on the right-hand side of (15) to be invertible, we require that

$$\left\|\frac{dt}{c_w}\left(B - \mathbb{D}_h\right)\right\|_{\ell^\infty} \leq \frac{1}{2} \tag{16}$$

$$\Leftrightarrow dt \leq \frac{c_w}{2}\left\|B - \mathbb{D}_h\right\|_{\ell^\infty}^{-1}. \tag{17}$$

Therefore, given a grid resolution h, we choose $dt \doteq \frac{c_w}{10}\left\|B - \mathbb{D}_h\right\|_{\ell^\infty}^{-1}$ with an additional loss by a factor of $\frac{1}{5}$ for safety.

0.3 Error benchmarking

The two solution methods above are compared against an analytically-derived steady-state solution in the file `analytic_benchmark.py`. Given the nonlinearities are represented by Legendre basis functions which are defined such that they solve the eigenvalue problem (5), we may try to construct solutions to the steady-state problem out of these basis functions. The analytic function is defined from the canvas-page code in `analytic1d.py`. From this process, the output of solutions using implicit Euler, as well as the benchmark itself seem to be outputting non-sensical values. I have tried to de-bug each of these but unsuccessfully. A next step is to try and solve for the analytic solution via Legendre basis elements. Note that repeating this task for arbitrary N, initially set to 2 (with the odd component stemming from $\phi_1(x)$ removed), would allow us to solve for analytic steady states for more general $S(x)$ and $a(x)$ distributions.

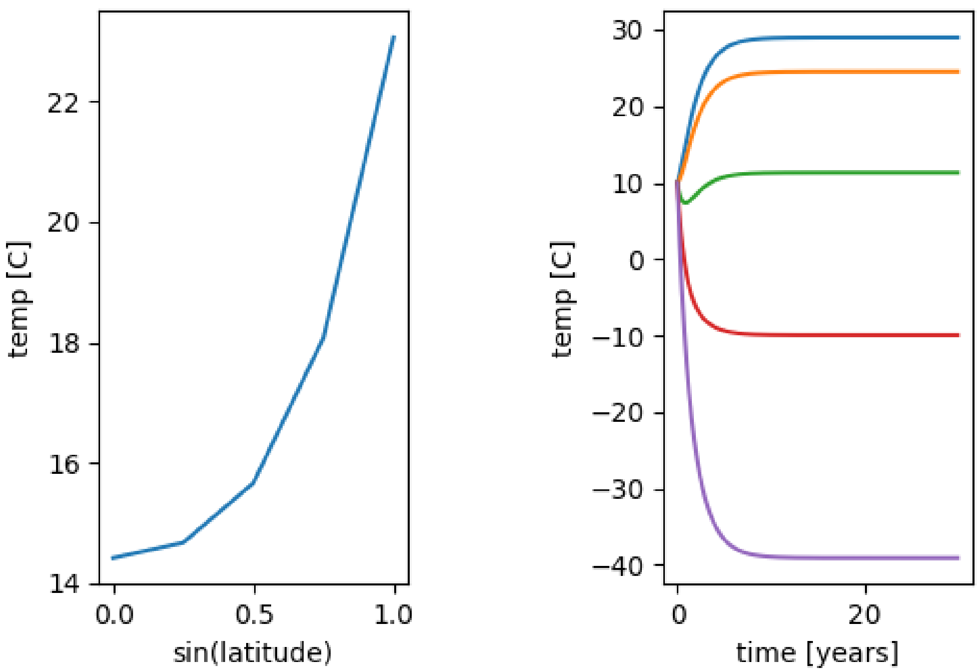
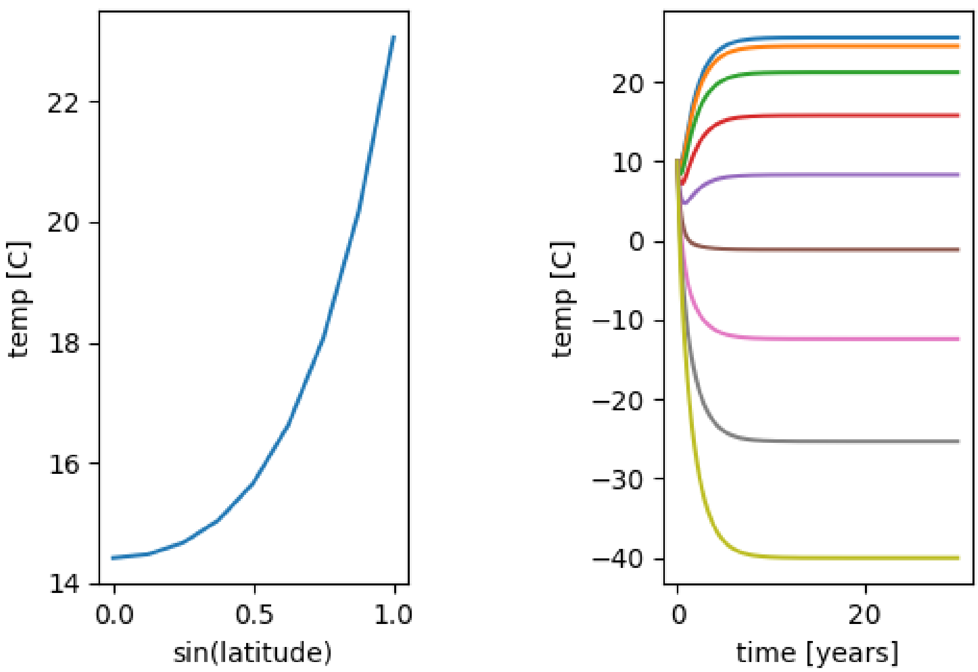
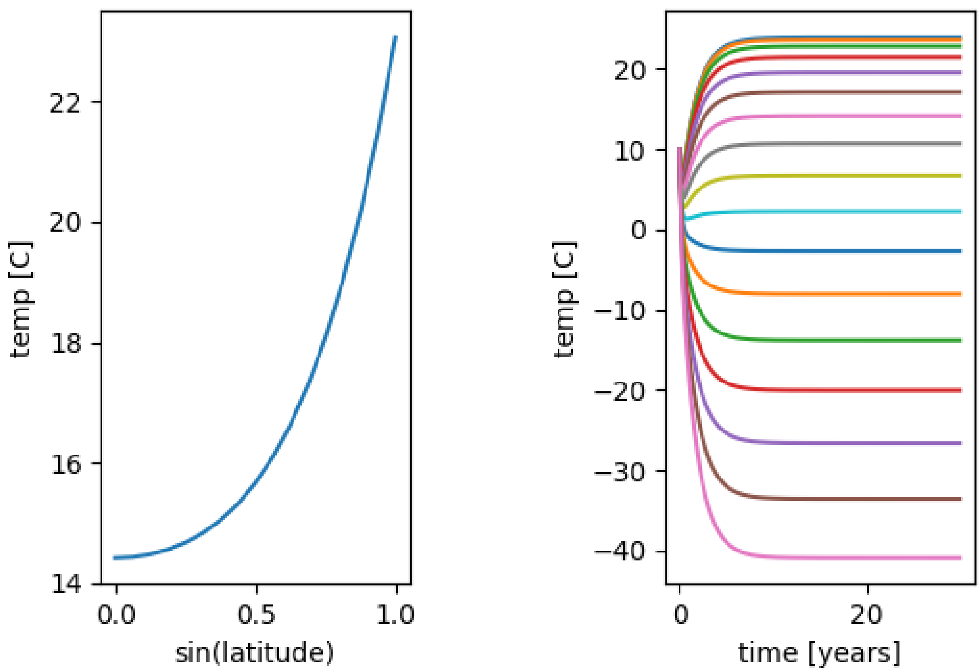

22
20
18
16
14
temp [C]
0.0
0.5
1.0
sin(latitude)
30
20
10
0
−10
−20
−30
−40
temp [C]
0
20
time [years]

Figure 1: $n = 2^2$



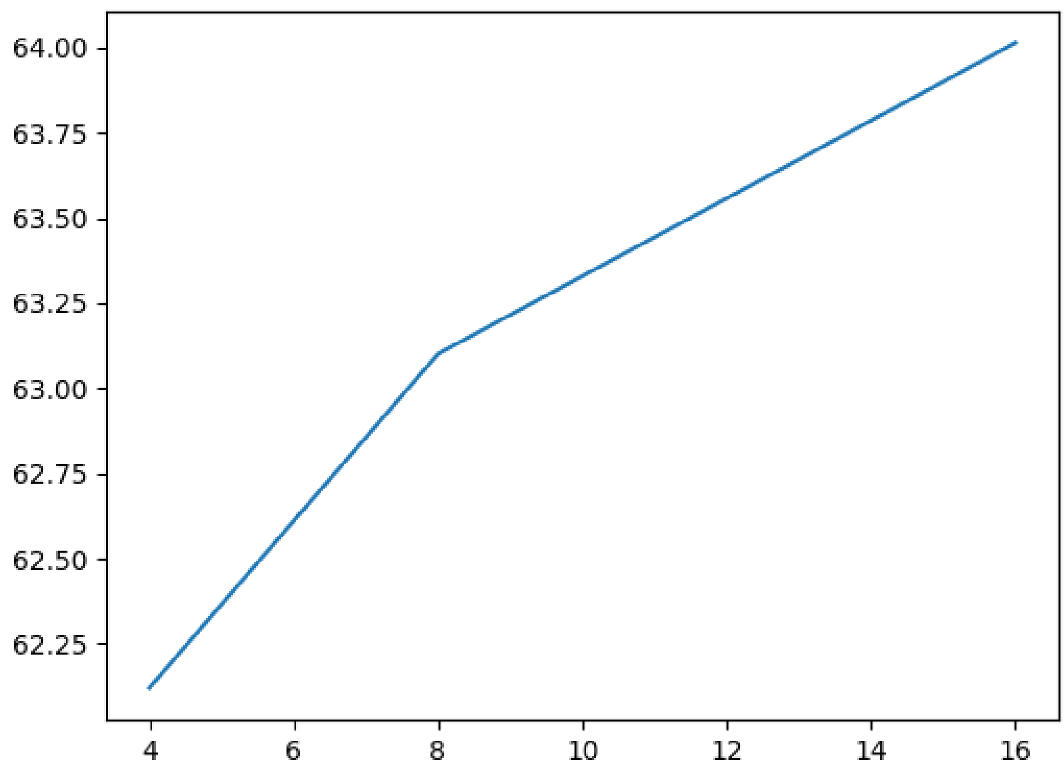
temp [C]
22
20
18
16
14
0.0
0.5
1.0
sin(latitude)
temp [C]
20
10
0
−10
−20
−30
−40
0
20
time [years]

Figure 2: $n = 2^3$



22
20
18
16
14
0.0
0.5
1.0
sin(latitude)
temp [C]
20
10
0
−10
−20
−30
−40
0
20
time [years]
temp [C]

Figure 3: $n = 2^4$


64.00
63.75
63.50
63.25
63.00
62.75
62.50
62.25
4
6
8
10
12
14
16