AOS 801 - Fall '22

Till Wagner

Diffusive Energy Balance Models

Lectures 8-10

3-10 Oct 2022

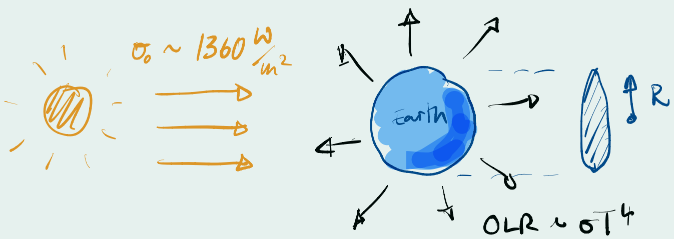
Energy Balance Models (EBMs)

Budyko - Sellers climate models

1969

Reading: North et al (1981) Rev. Geophys, also Intro Videos on YouTube / Canvas

1. "0-D" picture



"gray rock" model (no atmosphere)

$$(1) \quad \underset{\text{incoming}}{\pi R^2 \sigma_0 a} = \underset{\text{outgoing}}{4\pi R^2 \sigma T^4}$$

↳ coalbedo, $a = (1-\alpha) \approx 0.7$ for Earth

Solve (1) for T: $T = \left(\frac{\sigma_0 a}{4\sigma}\right)^{1/4} \approx -21°C$ → too cold since we're missing an atmosphere

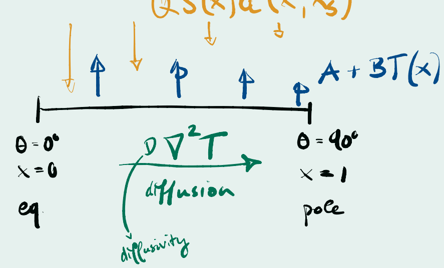
↳ define $Q = \frac{\sigma_0}{4}$

Include Atmosphere (Budyko 69): $\sigma T^4 \approx A + BT$ (A,B are empirical constants) ⟶ $T = \left(\frac{Qa - A}{B}\right) \approx 15°C$

2. "1-D" picture

write $x = \sin\theta$

θ - latitude



distribution of solar radiation: $S = S_0 + S_2 x^2$

$\left[\int_0^1 S(x)\,dx = 1\right]$

Here: consider only the annual mean $(S = \bar{S})$

2.1. Governing energy balance equation

$$\boxed{\underbrace{c_w}_{\text{heat capacity of ocean surface mixed layer } (\sim 50\text{m})} \frac{\partial T}{\partial t} = QSa - (A+BT) + D\nabla^2 T}$$

where
$T = T(x,t)$
$S = S(x)$
$a = a(x, x_s)$

2.2. Albedo

→ we're considering planetary (co-)albedo (top-of-atmosphere)

→ increased cloud cover at high latitudes & zenith angle dependence of albedo lead to x-dependence of a

→ ice-free planet : $\boxed{a(x) \simeq a_0 + a_2 x^2 \quad (2)}$

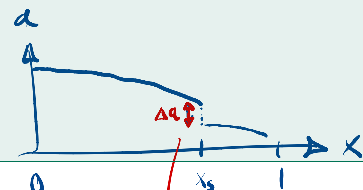
heaviside

→ planet with sea ice cover for $x > x_s$: $\boxed{a(x) \simeq a_0 + a_2 x^2 - \Delta a\, H(x - x_s)}$

→ ice-edge: $x_s = x_s(T(x,t))$

∴ albedo feedback

current climate :



↳ jump introduces nonlinearity → gives rise to

A) small ice cap instability ("Tipping Point")

B) Snowball earth instability (in a cooling climate)

2.3. Diffusion

$$(3) \quad \nabla^2 T = \frac{d}{dx}\left[D (1-x^2) \frac{dT}{dx} \right]$$

→ spherical laplacian, no radial, no azimuthal dependence

diffusivity $\cancel{D = D(x)}$

for us: D is constant

↳ in reality D is higher in the tropics, due to Hadley cell

↳ Flux $\propto \nabla T = (1-x^2)^{1/2} \frac{\partial T}{\partial x}$

2.4. Boundary Conditions :

↳ no flux through equator $(x=0)$: $\boxed{\left.\frac{\partial T}{\partial x}\right|_{x=0} = 0}$

↳ no flux through pole $(x=1)$ $\boxed{\left.(1-x^2)^{1/2}\frac{\partial T}{\partial x}\right|_{x=1} = 0}$ (trivially satisfied / not really a 2nd B.C.)

3. Numerical Model

3.1. Background:

mostly used for ODEs

Forward Euler : say $\boxed{\frac{dT}{dt} = f(T)}$

discretize, and at timestep 'i' you have :

$$\frac{T_{i+1} - T_i}{\Delta t} = f(T_i) \quad \therefore \quad T_{i+1} = T_i + \Delta t\, f(T_i) = T_i + \Delta T_i$$

Backward Euler : $\frac{T_{i+1} - T_i}{\Delta t} = f(T_{i+1})$

$\boxed{T_{i+1} = T_i + \Delta t\, f(T_{i+1})}$ → solve for T_{i+1} (if you can)

timestepping our EBM

more general than the Euler methods (also used in PDEs)

Central difference :

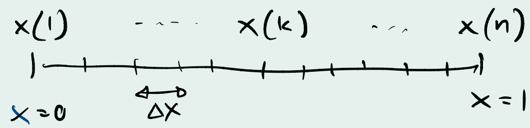
$$\frac{dT}{dx} \rightarrow \frac{T^{k+1} - T^{k-1}}{2\Delta x}$$

$$\frac{d^2T}{dx^2} \rightarrow \frac{T^{k+1} - 2T^k + T^{k-1}}{\Delta x^2}$$

→ need this for diffusion operator "$\nabla^2 x$"

3.2. No-Transport Case $D=0$

total of n gridboxes



n independent ODEs

$$C_w \frac{dT^{(k)}}{dt} = Q S^{(k)} a^{(k)} - \left(A + B T^{(k)}\right) = C^{(k)} - B T^{(k)}$$

where $\left[C^{(k)} \equiv Q S^{(k)} a^{(k)} - A\right]$

solve using Forward Euler to give:

$$\boxed{T_{i+1}^{(k)} = T_i^{(k)} + \frac{\Delta t}{C_w}\left(C_i^{(k)} - B T_i^{(k)}\right)}$$

time-dep. due to albedo jump

3.3. Transport Case, $D \neq 0$

B Spatial Differencing

from earlier:
$$D\nabla^2 T = D\left[(1-x^2)\frac{d^2T}{dx^2} - 2x\frac{dT}{dx}\right]$$

Boundary Conditions:

equator: $x(1) = 0 \quad \therefore \quad D\nabla^2 T_i \Big|_{x=0} = D \frac{d^2 T_i}{dx^2}\Big|_{x=0} = D\,\frac{T_i^{(2)} - 2T_i^{(1)} + T_i^{(0)}}{\Delta x^2}$

timestep

plug in here

BUT! T_0 is outside the domain → invoke "Ghost Point":
$$\frac{dT_i}{dx}\Big|_{x=0} = \underbrace{\frac{T_i^{(2)} - T_i^{(0)}}{2\Delta x} = 0}_{\text{no flux through eq.}} \quad \therefore \quad T_i^{(0)} = T_i^{(2)}$$

$$\boxed{D\nabla^2 T_i\Big|_{x=0} = D\,\frac{2\left(T_i^{(2)} - T_i^{(1)}\right)}{\Delta x^2}}$$

accuracy $\mathcal{O}(\Delta x^2)$

pole: $x(n) = 1 \quad \therefore \quad \nabla^2 T_i^{(n)}\Big|_{x=1} = -2\frac{d}{dx}T_i\Big|_{x=1} = \boxed{-2\,\frac{T_i^{(n)} - T_i^{(n-1)}}{\Delta x}}$ (*)

accuracy only $\mathcal{O}(\Delta x)$

[we can't do central diff. here since that gives $\frac{T^{(n+1)} - T^{(n-1)}}{\Delta x}$ and we don't have $T^{(n+1)}$ (and no indep. BC)]

(*) increase accuracy of backward diff. by including higher order terms:
$$\frac{dT_i}{dx}\Big|_{x=1} \approx \frac{1}{2\Delta x}\left(3T_i^{(n)} - 4T_i^{(n-1)} + T_i^{(n-2)}\right)$$

B Time Differencing

Implicit Euler :

$$\boxed{T_{i+1} = T_i + \underbrace{\Delta T_{i+1}}_{\text{get this from governing equation}}} \quad (1)$$

here, T is a vector of length n (n being # of spatial gridboxes)

$$C_w \frac{\Delta T_{i+1}^{(k)}}{\Delta t} = C_i^{(k)} - B T_{i+1}^{(k)} + \underbrace{D T_{i+1}^{(k)}}$$

from $D\left[(1-x^2)\frac{d^2T}{dx^2} - 2x\frac{dT}{dx}\right]$

$$\underbrace{D\left[(1-x_k^2)\left(\frac{T_{i+1}^{(k+1)} - 2T_{i+1}^{(k)} + T_{i+1}^{(k-1)}}{\Delta x^2}\right) - 2x_k\left(\frac{T_{i+1}^{(k+1)} - T_i^{(k-1)}}{2\Delta x}\right)\right]}$$

diffusion operator $\mathbb{D}$: Tri-Diagonal matrix (HW3)

$$\Delta T_{i+1} = \frac{\Delta t}{C_w}\left[C_i - (\mathbb{1}B - \mathbb{D})T_{i+1}\right] \quad (2)$$

$\mathbb{1}$: identity matrix

plug (2) into (1)

$$T_{i+1} = T_i + \frac{\Delta t}{C_w}\left[C_i - (\mathbb{1}B - \mathbb{D})T_{i+1}\right]$$

Solve for T_{i+1}

$$\underbrace{\left[\mathbb{1} + \frac{\Delta t}{C_w}(\mathbb{1}B - \mathbb{D})\right]}_{M} T_{i+1} = T_i + \frac{\Delta t}{C_w} C_i$$

M is computed just once, outside the integration

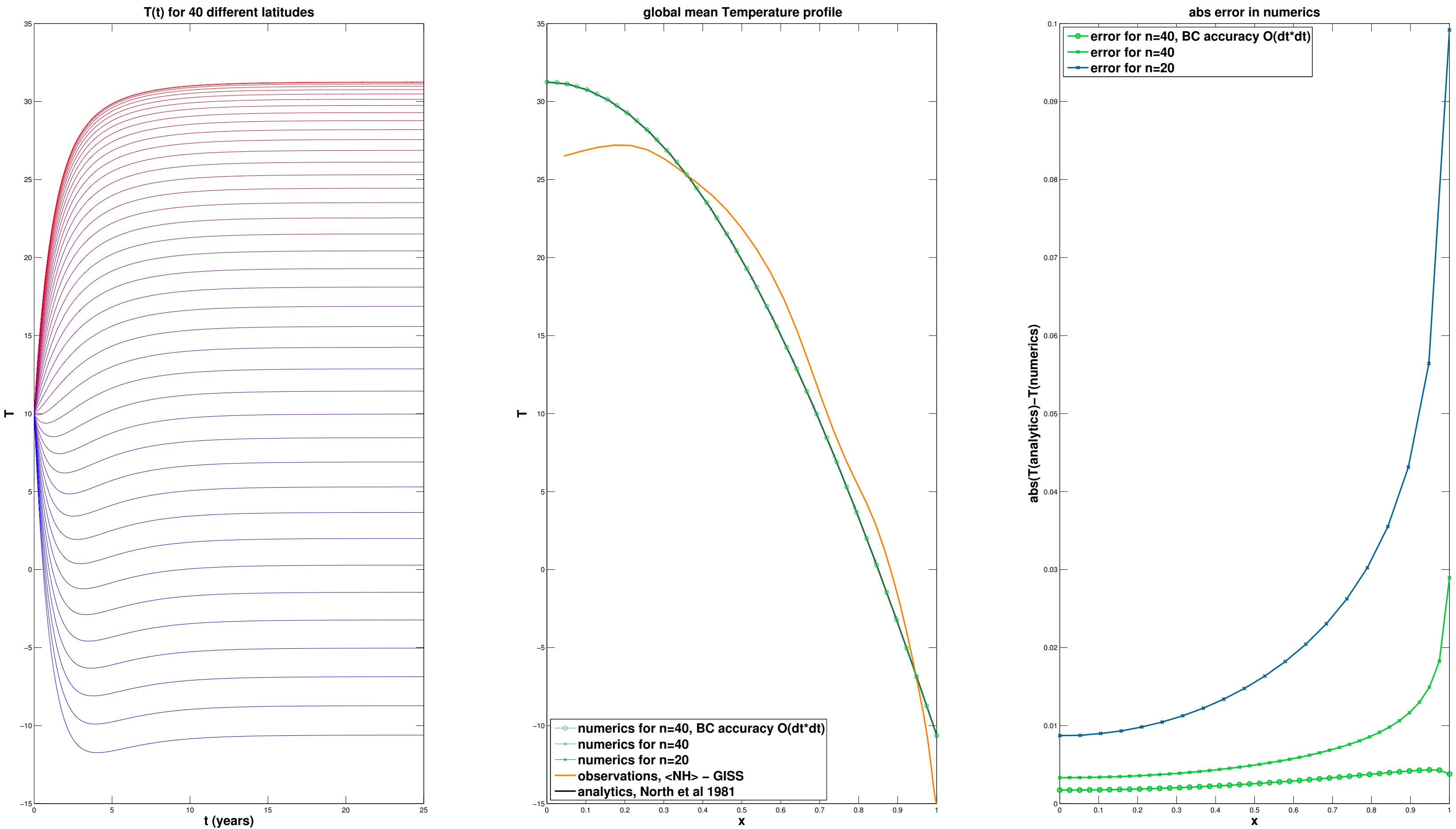
$$M\,T_{i+1} = T_i + \frac{\Delta t}{C_w} C_i$$

multiply both sides by M^{-1}

$$\boxed{T_{i+1} = M^{-1}\left(T_i + \frac{\Delta t}{C_w} C_i\right)}$$

to get the inverse use numpy.linalg.inv

this operation is a matrix multiplication → use numpy.dot$\left(M^{-1}, T + \frac{\Delta t}{C_w}C\right)$



T(t) for 40 different latitudes
T
t (years)
global mean Temperature profile
T
x
numerics for n=40, BC accuracy O(dt*dt)
numerics for n=40
numerics for n=20
observations, <NH> – GISS
analytics, North et al 1981
abs error in numerics
abs(T(analytics)−T(numerics))
x
error for n=40, BC accuracy O(dt*dt)
error for n=40
error for n=20