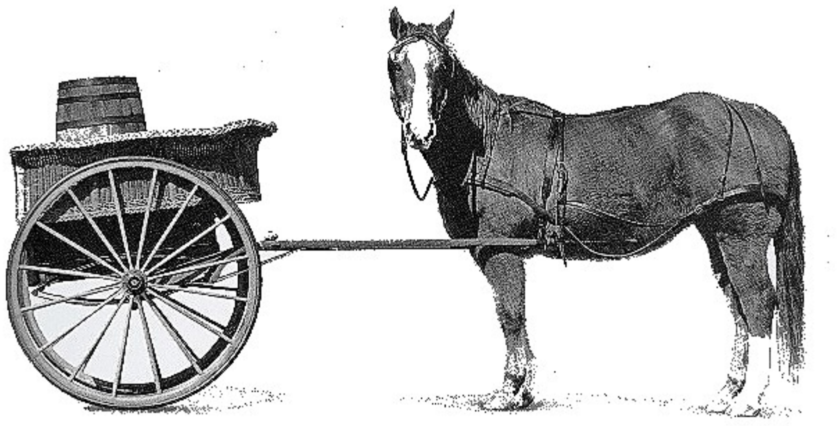


or

A **H**orse **B**efore the **C**art Model

"Radiative Convective Equilibrium" *(RCE)* has been a long-time, personal anathema. Early calculations of tropospheric infra-red spectra showed a necessity for first assuming a temperature profile and a simple linear one proved useful,

$$T(x) = T_0 + a\,x \tag{1}$$

The gradient or lapse rate, *a*, was chosen to be the adiabatic value, -g/c_p. Subsequently, this equation was adopted for calculating greenhouse gas perturbations of T_0 , but always with the lapse rate determined by an equilibrium parameter, $c_p(x)$. This approximation has persisted over half a century as the *RCE* model and remains the cornerstone of contemporary climate science modeling. From elementary thermodynamics, equilibrium systems in a gravitational field are necessarily free of thermal gradients.[1] From elementary fluid dynamics, the adiabatic lapse rate defines a zero-flux threshold for steady-state convection.[2] While this lapse rate asserts initial and final states of a convective translation are of equal internal energy, it does not imply that transitions at finite rates are free of dissipation.

The notion that perhaps it might be better to put the horse before the cart is absent in climate science methodology. Why don't we first find the thermal profile for a dissipative system? And that shall be the theme for this essay! A formal definition for temperature is entwined in concepts of path-independence and exact differentials. Should we apply a set of thermal contacts to an object, we require that energy fluxes at these contacts be reproducible after a steady-state is reached, independent of history. The internal thermal profile will then be that which minimizes the work or dissipation needed to maintain this state.[3] In a one-dimensional case, this is given by Carnot's equation,

$$Dissipation = J\left(1 - T_2/T_1\right) \tag{2}$$

with *J* the energy flux entering at T_1 and exiting at T_2. The variational solution for *T(x)* minimizes internal deviations of total flux when modeled as local functions of *T(x)*.

1 Theory of Heat, J. Clerk Maxwell, Longmans, Green and Co., 1872, p.300.
2 Fluid Dynamics, "§4. The condition that convection is absent", L.D. Landau and E.M. Lifshitz, Addison-Wesley (1959)
3 https://pdq2021.000webhostapp.com/Thermal_Dissipation_V.pdf

We shall be interested in two fundamentally different energy fluxes, nominally radiation and convection. At any given point, convection has a specific value and direction. Radiation, however, is a superposition of fluxes flowing in multiple directions at any point. Laser beams passing through the same point in vacua are oblivious to each other. In a one-dimensional model, we must presume three distinct fluxes, J_r^+, J_r^-, and J_c , which add to a spatially constant value.

To describe radiative flux, we adopt the gray gas model,

$$\begin{aligned} \lambda(x)\frac{dJ_r^+}{dx} &= -J_r^+ + \sigma T(x)^4 \\ \lambda(x)\frac{dJ_r^-}{dx} &= J_r^- - \sigma T(x)^4 \end{aligned} \tag{3}$$

with σ the Stefan-Boltzmann constant and λ a distance characterizing radiative absorption.[4]

Equations 3 may be written more succinctly in the form

$$\begin{aligned} [\lambda D + 1]J_r^+(x) &= \varphi(x) \\ [\lambda D - 1]J_r^-(x) &= -\varphi(x) \\ \varphi(x) &= \sigma T^4(x) \end{aligned} \tag{4}$$

We represent convection by a function proportional to the gradient of temperature,

$$J_c(x) = -\kappa(x)\frac{dT}{dx} = -\kappa(x)\,\psi'(x) = -\kappa(x)\,D\psi(x) \tag{5}$$

To describe the steady state, we then seek solutions for which the total flux is independent of x,

$$D[J_r^+(x) - J_r^-(x) + J_c(x)] = D\,J_{tot} = 0 \tag{6}$$

Thus,

$$\begin{aligned} -J_{tot} &= [\lambda D - 1]\varphi(x) + [\lambda D + 1]\varphi(x) - [\lambda D + 1][\lambda D - 1]\kappa D\psi(x) \\ &= 2\lambda D\varphi - [\lambda D + 1][\lambda D - 1]\kappa D\psi \\ &= 2\lambda D\varphi - [\lambda D\,\lambda D - 1]\kappa D\psi \end{aligned} \tag{7}$$

or simply,

$$-J_{tot} = 2\lambda D\varphi + \kappa D\psi - \lambda D\,\lambda D\,\kappa D\psi \tag{8}$$

As φ and ψ are functions only of $T(x)$, the problem to be solved, given explicit functions for κ and λ, is finding that function, $T(x)$, rendering J_{tot} independent of x. This 3rd order inhomogeneous differential equation is exact for the defined φ and ψ functions. The two leading terms linear in κ and λ and may be labeled as $J_r(x)$ and $J_c(x)$, 'radiative' and 'convective' components of the total flux.

The third term, J_{rc}, is a hybrid, neither fish nor fowl, and long a puzzlement. In many calculations, it contributes but a few percent to the total. As λ becomes commensurate with the physical size of the system, however, it becomes a major factor. In this limit, radiation is passing between boundaries without significant dissipation. Nominal solutions will presume black-body boundaries defining T_l and

4 It should be noted that J_r^- is a positive number for the magnitude of the flux flowing towards the warmer interface.

T_2. Physically, however, no black-body exists at the tropopause to contribute to J_r^- . When *κ(x)* vanishes at T_2, all flux becomes radiative . This is resolved by supposing J_{rc} is the reduction of J_r by an absent, hypothetical black-body at T_2, and that the appropriate value for J_r is given by J_r plus J_{rc} . And so we presume, although a cleaner mathematical argument is welcome!

Atmospheric functions are slowly varying functions of altitude and ought be amenable to description by polynomials. The Python programming language includes compiled functions for multiplying and differentiating polynomials and for finding coefficients to minimize differences between nonlinear trial and reference functions. The remainder of this essay will explore how *Equation 8* describes atmospheric fluxes of energy.

The variational function adopted to describe temperature is an *8th* order polynomial representing deviations from a default linear profile,

$$T(x) \;=\; T_1+(T_2-T_1)*[x + x(1-x)(a + bx + cx^2 + dx^3 + ex^4 + fx^5 + gx^6)] \qquad (9)$$

Our basic model is a troposphere *10km* deep (*x=1*) with boundary temperatures *285K* and *220K* (mean lapse rate *-6.5K/km*). Trial polynomials for *κ* and *λ* are

$$\begin{aligned} \lambda &= 1500 * (1 + 2.0\,x) \\ \kappa &= 25000 * (1 - 1.0\,x) \end{aligned} \qquad (10)$$

The *λ* function presumes an absorption depth of *1500 meters* at the surface, tripling to *4500 meters* at an altitude of *10 km*. This follows from the 3-fold decrease in air density at this altitude. Its magnitude was chosen to set a total flux density *ca. 240 W/m²* when convection is absent (*x=1*) . Note that these distances are commensurate with the thickness of our troposphere! The *κ* function describes a surface convective flux of *160 W/m²* , slightly above that for radiation, in accord with MODTRAN calculations. Given these parameters, the following plots appear:

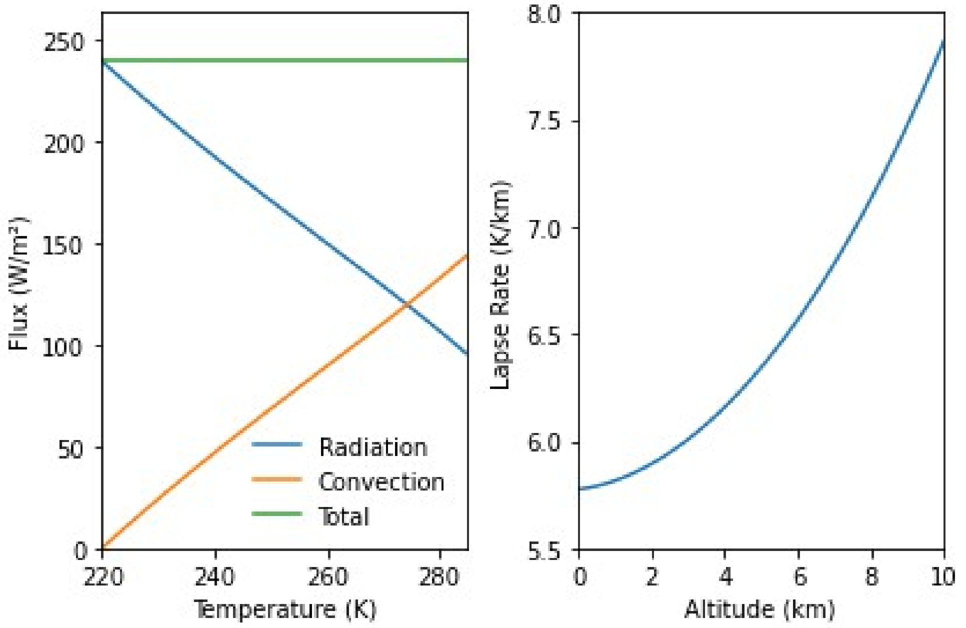


Fig 1: HBC parameters: λ(0)=1500m, κ(0)=25000 W/m/K, T_1=285K, T_2=220K

The mean value for J_{tot} is *239.542772*, with a standard deviation of *0.000083*. The apparent linearity of fluxes seems not overly informative, but lapse rate variation contradicts the fundamental premise of CRE modeling. For the profile in *Eq. 9,* it is of some interest to examine the consequences of varying the number of variational coefficients,

Coefficients	Mean Flux	Std. Dev.
1	240.679669	9.529233
2	239.430446	0.460400
3	239.688475	0.106538
4	239.582352	0.012007
5	239.556492	0.001852
6	239.547983	0.000480
7	239.542772	0.000083
8	239.540624	0.000017
9	239.539490	0.000004
10	239.539703	0.000005

Table 1: Variational Minima

The calculated mean flux is nearly constant whilst deviations decrease by six orders of magnitude as the number of variational parameters increases, illustrating the efficacy of variational solutions. (The non-monotonic decrease towards convergence is due to lack of a 1st Law constraint equating boundary fluxes.)

Similar plots for CRE models can be drawn from online MODTRAN calculators.[5] The following results were obtained for the U.S. Standard Atmosphere. These programs return J_r^+ and J_r^- values showing a doubling of radiative flux on transit from surface to tropopause. As a steady-state flux needs be non-divergent, a 'convective' flux is inferred which vanishes at the tropopause.

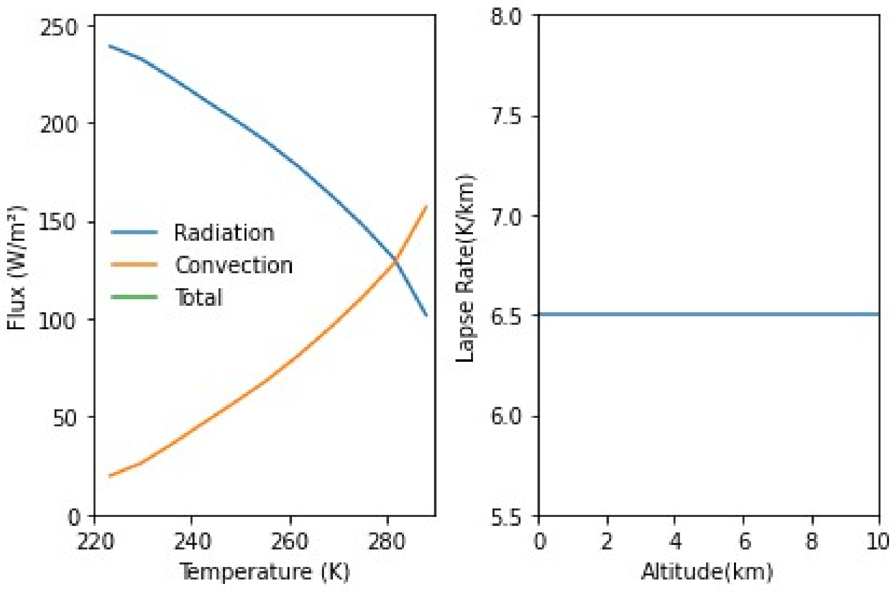


Fig 2: MODTRAN CRE results for U.S. Standard Atmosphere.

5 http://climatemodels.uchicago.edu/modtran/

While lapse rate variations with altitude might seem testable, actual plots appear visually linear and suggest otherwise.

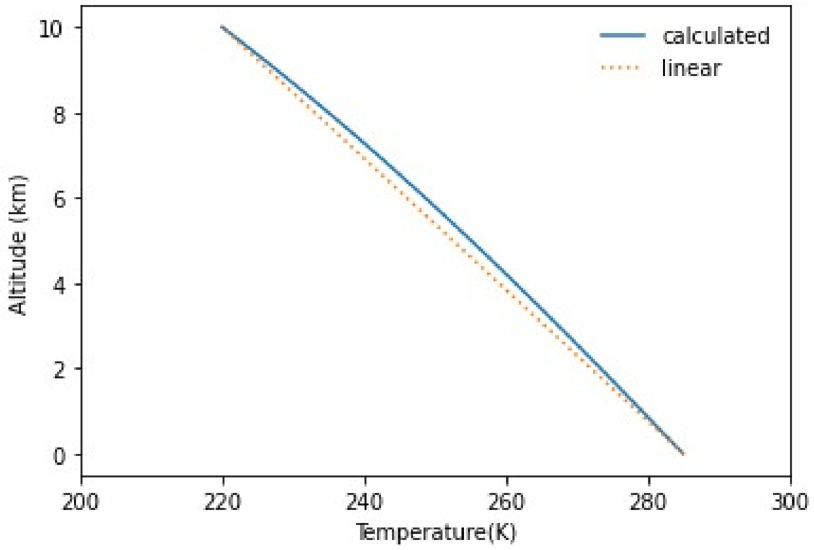


Fig 3: Model parameters as in Fig. 1.

To visualize the interdependence of radiation and convection, the following table shows relative changes of the several fluxes <u>at the surface</u> for 1% changes of temperature, radiation (λ) and convection (κ). Also listed is the equivalent change in surface temperature required to restore the unperturbed flux value.

1% Perturbation in	Total Flux	Surface Radiation	Surface Convection	Equivalent ΔT_1(K)
T_1	5.417%	7.162%	4.266%	-2.850
λ(0)	0.705%	1.332%	0.292%	-0.371
κ(0)	0.451%	-0.114%	0.823%	-0.237

Table 2: Parameters as in Fig. 1, T_2 = 220K.

A *7%* radiation change for a *1%* temperature change is twice that which might be expected from back-of-the-envelope estimates. This is of some importance as it implies that surface temperature is significantly less dependent on radiation than is conventionally supposed. A Stefan-Boltzmann function, $J \propto T^4$, would give *4%*. From *Eq. 6,*

$-J_r = 2\lambda\phi' = 8\lambda\sigma T^3 T'$ and logarithmically, $\dfrac{\delta J_r}{J_r} = 3\,\dfrac{\delta T}{T} + \dfrac{\delta T'}{T'}$.

The explanation appears to be a varying lapse rate. The λ parameter is not an explicit function of temperature and it's *1%* effect on J_r is quasi-linear. The outgoing radiation, *i.e.* total flux, increases *5%* while surface convection is, perhaps surprisingly, up *4%* and indicative of lapse rate changes.

Greenhouse gas (*GHG)* effects may be modeled as λ perturbations. Current models suggest that CO_2 doubling reduces outgoing fluxes by *3.7* W/m^2 or about *1.54%,* equivalent to a *λ(0)* decrease of *2.18% (1.54/.705)* and *0.81K (3.7*(285/239.5)*/5.417)* warming. This value is basically half that conventionally calculated. The difference lies in the *RCE* presumption that thermal gradients are set by convection to the adiabatic lapse rate and thus a function only of the heat capacity, an equilibrium parameter. As this parameter is not significantly *GHG* dependent*,* then neither are thermal gradients.

A minor tweak in the HBC model lets us keep the difference between T_1 and T_2 constant to emulate an *ALR* model. With this constraint,

1% Perturbation in	Total Flux	Surface Radiation	Surface Convection	Equivalent ΔT_1(K)
T_1	1.866%	3.602%	0.721%	-2.850
λ(0)	0.705%	1.332%	0.292%	-1.077
κ(0)	0.451%	-0.114%	0.823%	-0.688

Table 3: $T_2 = T_1 - 65K$

The responses to surface temperature changes are markedly less and larger changes are required, with the compensating increase in surface temperature now 2.38K, a value in closer agreement with transient climate response estimates. The *3.6%* dependence of surface radiation for T_1 changes is also closer to that expected for a T^4 dependence.

The obvious message is that *GHG* perturbations are dependent on how the temperature change at T_2 is modeled. For these calculations, it is the temperature at an altitude of *10,000* unit distances. Mathematically, it is the temperature at which convection vanishes (*Eq. 10*). If we postulate that this altitude defines a critical density or temperature independent of *GHG* radiation absorption at lower altitudes, then the first model (*Table 1*) is appropriate. Should we postulate that this temperature varies as does the surface temperature, *i.e.* the adiabatic lapse rate, *GHG* effects are nearly tripled *(Table 3*).

Consider now the following experiment. Reduce λ by *1%.* J_{tot} drops *1.69* W/m^2. Now increase T_1 by *0.377K*, restoring J_{tot} to its original value. What are the net changes in J_r and J_c? In *Fig. 4* we see that radiation has dropped *0.4* W/m^2 in the lower troposphere and convection has increased an equivalent amount. This exercise illustrates that greenhouse gas perturbations are compensated for by a combination of radiative and convective changes. It is the total energy flux, not just its radiative component, which requires restoration and convection provides a significant '*negative feedback*' contribution.

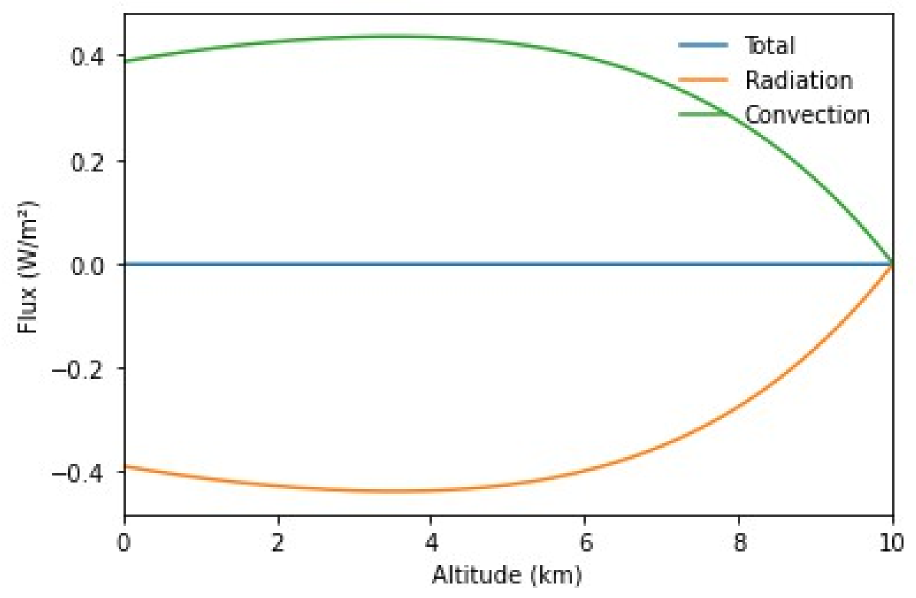


Fig 4: Net flux changes when λ is reduced 1% and T_1 is increased 0.377K.

The origins of *Convective Equilibrium* and the *Adiabatic Lapse Rate may* be traced back to Wm. Thomson, later to become Lord Kelvin.[6]

> *"When all the parts of a fluid are freely interchanged and not subject to the influence of radiation and conduction, the temperature of the fluid is said by the Author to be in a state of convective equilibrium."*

In the same paper, Kelvin calculates atmospheric gradients of *329ft/K (10.0K/km)* for dry air and values for wet air ranging 6.6 to 3.5K/km for temperatures 0°C to 35°C.

Shortly thereafter, J. C. Maxwell observed that a perpetual motion machine could be constructed should temperature differences exist between two vertical columns in thermodynamic equilibrium sharing a common thermal base.[1]

> *"In this condition of what Sir William Thomson has called convective equilibrium of heat, it is not the temperature which is constant, but the quantity ϕ, which determines the adiabatic curves."*

Indeed, if equilibrium is identified as a configuration of maximum entropy, it is not the derivative of local entropy per unit mass, *s(z)*, which vanishes, but the derivative of its volume integral. The thermodynamic condition that convection be absent is that entropy increase with altitude $ds/dz > 0$.[2] The adiabatic lapse rate, $ds/dz = 0$, corresponds to a threshold for convection. Finite convective fluxes entail negative *ds/dz* values.

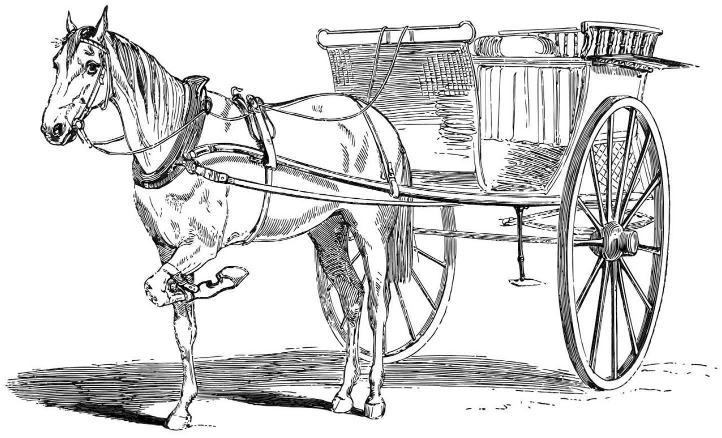
Textbook rationalizations presume a parcel embedded in a fluid from which it is adiabatically insulated, but in neutral buoyancy with no favored direction of motion. As it rises or falls its volume and density

6 "On the Convective Equilibrium of Temperature in the Atmosphere" by Professor Wm. Thomson, LL.D., P.K.S., &c, Manchester Phil. Soc. Proc. Vol. ii. 1860-1862, pp. 170-176 [Jan. 21, 1862]; https://www.biodiversitylibrary.org/item/39422#page/178

change, but it remains in neutral buoyancy, exchanging no thermal energy with its surroundings. Its rate of rise or fall will be determined by a viscous impedance and energy fluxes will depend on both this resistance and the internal energy of the parcel. While this might seen a plausible model for a weather balloon, it is not obvious why it should describe a parcel embedded in an ensemble of parcels, some rising some descending. In serious descriptions of free convection, fluxes depend on both kinematic viscosities and heat capacities (Prandtl numbers).[7]

The intent of this essay has been to suggest an alternative to the adiabatic lapse rate. We start with calculation of the temperature profile from non-equilibrium parameters rather than presuming an equilibrium parameter, *i.e.* $c_p(x)$. While the adiabatic lapse rate offers computational simplifications, it precludes incorporating the effects of greenhouse gases on thermal gradients. Thermodynamic gradients are sustained by free energy dissipation and require dissipative parameters for both radiation and convection. Calculations are based on a previously derived variational theorem to the effect that, with boundary values fixed, total energy flux is a minimum with respect to variations of the internal temperature profile. Thus calculation of this profile returns flux values of still greater accuracy because of its minimal nature *(Table 1).*

It is the minimization of global free energy dissipation, not local entropy gradients, that determines the structure and properties of the troposphere.



August 24, 2022

7 "Fluid Dynamics, §56 Free convection", L.D. Landau and E.M. Lifshitz, Addison-Wesley (1959)

Computations

The purpose of computation is insight, not numbers − R. Hamming

All computations, apart from MODTRAN results, used a Windows 7 system on a Dell 3867 desktop. Python software was downloaded from https://sourceforge.net/projects/winpython/ . A listing for the full program follows. Typical execution times are under a second and interactive exploration and experimentation, editing parameters in the editor/console on the go, works for me.

For those interested in pasting and modifying this code into a text file *run_me.py*, a brief summary may be useful. All spatially varying functions, *T, J,* λ and κ, are expressed as polynomials. Additions, multiplications and differentiations of polynomials yield new polynomials and the functions of interest are slowly varying. Three functions are defined corresponding to the three terms in *Eq. 8*. As inputs, each receives a *1-D* array of physical coordinates and an array of coefficients for a trial temperature profile. As output, they return corresponding functional values. For convenience, the J_r term also returns the thermal profile and its gradient. These three functions are called by a fourth which sums them and returns J_{tot} as defined in *Eq. 8*. The heavy lifting for finding the temperature profile for which J_{tot} at all points equals its mean value is done in the single line

data = leastsq(residuals, a, args=(J, x)) ,

which returns the polynomial coefficients, *a,* for *T(x)* minimizing *residuals*. The initial trial function for *a* is a set of seven null coefficients corresponding to a linear thermal profile. Probably overkill, but …

```
# -*- coding: utf-8 -*-
""" For variable names
α β γ δ ε ζ η θ ι κ λ μ ν ξ ο π ρ σ τ υ ϕ χ ψ ω
"""
import numpy as np
import pylab as plt
from scipy.optimize import leastsq
poly = np.polynomial.Polynomial


def Jrf(x, a):
    '''Return  λ(x)*Dϕ(x), T(x) and dT/dx '''
    p = poly([0, 1, -1]) * poly(a)   # (x - x^2)*(a + bx + cx^2 +...)
    p = poly([0, 1]) + p             # x + (x-x^2)*(a+bx+..)
    T = T1 + (T2-T1) * p
    dT = T.deriv(1)
    f = T**4
    f = f.deriv(1)
    f = poly(λ) * f
    return -2*σ * f(x) / height, T(x), dT(x) / height


def Jcf(x, a):
    '''Return κ(x)*Dψ(x)  '''
    p = poly([0, 1, -1]) * poly(a)   # (x - x^2)*(a + bx + cx^2 +...)
    p = poly([0, 1]) + p             # x + (x-x^2)*(a+bx+..)
    T = T1 + (T2-T1) * p
    f = T.deriv(1)
    f = f * poly(κ)
    return -f(x) / height
```

```
def Jrcf(x, a):
    '''Return λ(x)*D λ(x)*D κ(x)*Dψ(x)'''
    p = poly([0, 1, -1]) * poly(a)   # (x - x^2)*(a + bx + cx^2 +...)
    p = poly([0, 1]) + p             # x + (x-x^2)*(a+bx+..)
    T = T1 + (T2-T1) * p
    f = poly(κ) * T.deriv(1)
    f = poly(λ) * f.deriv(1)
    f = poly(λ) * f.deriv(1)
    return f(x) / height**3


def J(x, a):
    '''Returns the total flux as a function of  parameters in a[]'''
    Jr, _, _ = Jrf(x, a)
    Jc = Jcf(x, a)
    Jrc = Jrcf(x, a)
    return Jr + Jc + Jrc


def Js(x, a):
    '''The variational solution is for a constant value'''
    return np.mean(J(x, a)) * np.ones(len(x))


def residuals(a, J, x):
    return J(x, a) - Js(x, a)


x = np.linspace(0, 1, 1001)      # Normalized altitude

# Radiation parameters
σ = 5.67e-8                     # W/m^2/K
λ = 1500. * np.array([1, 2])     # meters ;default 1500, [1, 2]


# Convection parameters
κ = 25000. * np.array([1, -1.])   # W/m/K  ;default 25000,[1, -1]

# Troposphere
height = 10000                  # meters 10000
altitude = height * x
T1 = 1.0 * 285
T2 = T1 - 65             # Kelvin  285, 220
#T1 = 285 - 0.0           # For perturbation, T2 constant

# Polynomial coefficients for T(x)
a = np.array([0, 0, 0, 0, 0, 0, 0])     # default initial value

# Find coefficients making J(x) independent of x
data = leastsq(residuals, a, args=(J, x))
a1 = data[0]                           # coefficients of minimization
y = J(x, a1)
Jmean, Jstd = np.mean(y),  np.std(y)
print('Mean: {:10.6f}, Std: {:10.6f}'.format(Jmean, Jstd))
print('Coefficients: {} '.format(a1))

# Find fluxes for calculated temperature function
Jr, T, dT = Jrf(x, a1)
Jc = Jcf(x, a1)
Jrc = Jrcf(x, a1)
Jtot = Jr + Jc + Jrc
Jmax = np.max(Jtot)
i = 0
print(altitude[i], T[i], Jtot[i], Jr[i] + Jrc[i], Jc[i])
```

```
# Plot fluxes vs T and lapse rate vs altitude
fig, axes = plt.subplots(nrows=1, ncols=2)

# plot 1
ax = axes[0]
ax.plot(T, Jr + Jrc, label='Radiation')
ax.plot(T, Jc, label='Convection')
ax.plot(T, Jtot, label='Total')
ax.set_xlabel('Temperature (K)')
ax.set_ylabel('Flux (W/m²)')
ax.set_xlim(T2, T1)               # (220, 290)
ax.legend(frameon=False)
ax.set_ylim(0, 1.1*Jmax)

# plot 2
ax = axes[1]
ax.plot(altitude/1000, -1000*dT)
ax.set_xlabel('Altitude (km)')
ax.set_ylabel('Lapse Rate (K/km)')
ax.set_xlim(0, height/1000)
ax.set_ylim(5.5, 8)
fig.tight_layout()
plt.show()

# Plot altitude vs. temperature
plt.plot(T, altitude/1000, label="calculated")
Y = [altitude[0]/1000, altitude[-1]/1000]
X = [T1, T2]
plt.plot(X, Y, ls=':', label='linear')
plt.xlabel('Temperature(K)')
plt.ylabel('Altitude (km)')
plt.xlim(200, 300)
plt.legend(frameon=False)
plt.show()

# Plot differences between runs
''' To save a profile for perturbation changes, after the first profile has
been generated and in memory, copy the following line to the console window
and execute:
     AA=np.copy(Jtot); BB=np.copy(Jr + Jrc); CC=np.copy(Jc)
These arrays will persist for later reference until variables are cleared.
'''
# aa, bb, cc = Jtot[0]/AA[0] - 1, (Jr[0]+Jrc[0])/BB[0] - 1, Jc[0]/CC[0] - 1
# print(aa, bb, cc)
# plt.plot(altitude/1000, (Jtot - AA), label='Total')
# plt.plot(altitude/1000,(Jr + Jrc - BB), label='Radiation')
# plt.plot(altitude/1000,(Jc - CC), label='Convection')
# plt.xlabel('Altitude (km)')
# plt.ylabel('Flux (W/m²)')
# plt.xlim(0, height/1000)
# plt.legend(frameon=False)
# plt.show
```