1 Problem set 2: Modern radiation transport

1.1 Formal solution of the radiative transfer equation

Algorithms for solving radiation transport problems with scattering also discretize the angular (μ) space and approximate $\int f(\mu)\,d\mu$ with a quadrature sum $\sum w_i f(\mu_i)$.

Extend your python class from problem set 1 to

1. compute angles μ_i and w_i for a Gauss-Legendre quadrature with n_μ (4–32) points. Hint: numpy has a method for this purpose.

2. write a method to compute $I(\mu,\tau)$ for a given $S(\tau)$ for μ_i and on the $\tau_{\rm std}$ grid from Problem 1 (see part05.pdf):

$$\begin{aligned} I(\tau_i) &= I(\tau_{i-1})\exp(\tau_{i-1}-\tau_i) + \int_{\tau_{i-1}}^{\tau_i} S(\tau)\exp(\tau-\tau_i)\,d\tau \\ I(\tau_i) &\equiv I_{i-1}\exp(-\Delta\tau_{i-1}) + \Delta I_i \end{aligned}$$

   with linear or parabolic polynomials so that

$$\Delta I_i = \alpha_i S_{i-1} + \beta_i S_i + \gamma_i S_{i+1}$$

   with

$$\begin{aligned} \alpha_i &= e_{0i} + [e_{2i} - (\Delta\tau_i + 2\Delta\tau_{i-1})e_{1i}]/[\Delta\tau_{i-1}(\Delta\tau_i + \Delta\tau_{i-1})] \\ \beta_i &= [(\Delta\tau_i + \Delta\tau_{i-1})e_{1i} - e_{2i}]/[\Delta\tau_{i-1}\Delta\tau_i] \\ \gamma_i &= [e_{2i} - \Delta\tau_{i-1}e_{1i}]/[\Delta\tau_i(\Delta\tau_i + \Delta\tau_{i-1})] \end{aligned}$$

   for parabolic interpolation and

$$\begin{aligned} \alpha_i &= e_{0i} - e_{1i}/\Delta\tau_{i-1} \\ \beta_i &= e_{1i}/\Delta\tau_{i-1} \\ \gamma_i &= 0 \end{aligned}$$

   for linear interpolation. The auxiliary functions are:

$$\begin{aligned} e_{0i} &= 1 - \exp(-\Delta\tau_{i-1}) \\ e_{1i} &= \Delta\tau_{i-1} - e_{0i} \\ e_{2i} &= (\Delta\tau_{i-1})^2 - 2e_{1i} \end{aligned}$$

   Hint: I've added a method computeFScoeffs to the example class for convenience.

3. Use your method (routine) to compute $J(\tau_{\rm std}) = \sum w_i I(\mu_i, \tau_{\rm std})$ in (yet another) method to complete the formal solution. Hint: it may be useful to do this in 2 steps: first compute and store α, β, γ and then compute I in a second phase (this will be useful for the next problem!). Be aware of the boundary conditions!

4. Compute H, K, and the Eddington factor.

5. Compare the results for $S = 1$ with the results of the classic formal solution from problem 1.

1.2 Λ iterations

Use your new code to implement a Λ iteration and calculate the results for $S = (1-\epsilon)J + \epsilon B$ and $\epsilon = 1,\ 0.5,\ 0.1,\ 10^{-2},\ 10^{-4}$ (independent of τ for simplicity). Plot the convergence behaviors (corrections $\max(|\Delta J|/J)$ as functions of iteration number). Limit the number of iterations to 100 ... and profile the runs to see where the time is burnt (module 'cProfile' works well enough). Compare the results to the classic Λ iteration.