1 Problem set 2: Hints & Tipps

1.1 Formal solution of the radiative transfer equation

- re-use the class (or code) you wrote for the previous problem, in particular the tau grid and the infrastructure
- start by generating the points μ_j and weights w_j for the Gauss quadrature that replaces the J integration $J(\tau) = 1/2 \sum w_j I(\mu_j, \tau)$. There's a numpy method for this:
  `self.mu, self.w = np.polynomial.legendre.leggauss(nmu)`
- You will need to write a routine that computes $I(j, i)$ for all μ_j and τ_i
- This is done by compute $I(j, :)$ for each j in succession by applying the scheme discussed in the lectures and the problem sheet.
- For this you need the coefficients $\alpha_i, \beta_i, \gamma_i$. The method `computeFScoeffs` in the template does that for you, you need top provide S_{i-1}, S_i, S_{i+1} in `Sm, S0, Sp` and $\tau_{\text{prev}\to\text{curr}}$ and $\tau_{\text{curr}\to\text{next}}$ in `dtau, dtau1`. Set `self.taulin=1e-2` and `self.use_Bezier = False`.
- It is easier to first (for a given direction μ_j) compute the $\alpha_i, \beta_i, \gamma_i$ for all i and store them in numpy arrays and then in a second step to compute the $I(j, i)$ for all i and given j.
- Keep in mind that the direction of solving changes depending on the sign of μ_j. For $\mu > 0$ you need to got from large τ towards zero, for $\mu < 0$ it is from zero to τ_{max}. Similar for γ. Remember that the solution follows the propagation of the photons. α always considers the previous point, that can be either $i-$ or $i + 1$, depending on sign(μ_j).
- With the routine computing $I(i, j)$, the formal solution is now easy, just call it for the different j and compute $J(i) = \sum_j w_j I(j, i)$. Note: you do NOT need to store $I(j, i)$, just call your routine and the contributions for J on the fly. For this exercise it's easy to just store the full $I(j, i)$, but imagine you have millions of τ points (3D models!) and thousands of j ...
- In general, it is best to scetch the solution on a (big) sheet of paper. Think first, code later. In a variation of a quote from Stanislav Lem: Think more, code less.