Jacobians in LIDORT

February 18, 2011

As part of the Level 2 Full Physics calculation, we need to calculate the Jacobian of the radiance calculated by the LIDORT raditiave transfer code, giving the deriviatives with respect to the state vector:

$$\frac{\partial I}{\partial S_i} \tag{1}$$

Here i is the state vector index, i.e it typically goes from 0 to something like 111 (depending on the exact make up of the state vector).

However, LIDORT doesn't work directly with out state vector. Instead, it takes a weighting function and returns two results:

$$J_{lj}^{\text{atmosphere}} = \frac{\partial I}{\partial \tau_l} W_{lj}^{\tau} + \frac{\partial I}{\partial \omega_l} W_{lj}^{\omega} + \sum_k \frac{\partial I}{\partial \gamma_{lk}} W_{lkj}^{\omega} \tag{2}$$

$$J_i^{\text{surface}} = \frac{\partial I}{\partial a_i} \tag{3}$$

In this equation, l is the layer number, j is the number of weighting values passed in, k goes over the moments of the phase function, and a_i is the surface parameters (e.g., the surface albedo for the lambertian case). The quantities τ, ω, and γ are the optical depth, single scattering albedo, and phase function moments.

If speed weren't an issue, we could easily calculate the desired results of equation (1) by just passing in the partial derivatives of each of τ, ω, and γ with respect to the state vector, so we would have:

$$\frac{\partial I}{\partial S_i} = \sum_l J_{li}^{\text{atmosphere}} + \sum_j J_j^{\text{surface}} \frac{\partial a_j}{\partial S_i} = \sum_l \left(\frac{\partial I}{\partial \tau_l} \frac{\partial \tau_l}{\partial S_i} + \frac{\partial I}{\partial \omega_l} \frac{\partial \omega_l}{\partial S_i} + \sum_k \frac{\partial I}{\partial \gamma_{lk}} \frac{\partial \gamma_{lk}}{\partial S_i} \right) + \sum_j \frac{\partial I}{\partial a_j} \frac{\partial a_j}{\partial S_i} \tag{4}$$

However, testing has shown that the speed of LIDORT is directly proportional to the number of columns passed in the weighting function (see DFM-0007, "LIDORT Performance"). This is a somewhat surprising results, one would expect the Jacobian to have some fixed cost in calculating, with perhaps a small cost as the weighting function grows in size. But that is not the case. Since our state vector is typically 100 elements or so, this means that directly

passing in the weighting function of the Jacobians with respect to S_i would take about 100 times as long to calculate LIDORT with the Jacobian vs. without Jacobians.

We can improve the performance by reducing the time taken in LIDORT by changing the weighting function passed in. The surface portion is a fixed cost, there is no surface weighting function supplied, we just have $\frac{\partial I}{\partial a_j}$ returned by LIDORT. We then multiple this by $\frac{\partial a_j}{\partial S_i}$ and sum over j to give us the surface portion of the jacobian.

Because τ and γ are 1 dimensional, we can get the deritivative of I for these two variables by passing two columns of the weighting function that are all 1's. So with:

$$W^{\tau} = \begin{bmatrix} 1 & 0 & \dots & 0 \\ 1 & 0 & \dots & 0 \\ & \ddots & & \\ 1 & 0 & 0 & 0 \end{bmatrix} \tag{5}$$

$$W^{\omega} = \begin{bmatrix} 0 & 1 & \dots & 0 \\ 0 & 1 & \dots & 0 \\ & \ddots & & \\ 0 & 1 & 0 & 0 \end{bmatrix} \tag{6}$$

$$W^{\gamma}_{lkj} = 0 \text{ where } k = 1 \text{ or } 2 \tag{7}$$

Then when we pass these weighting functions to LIDORT, we get back the results:

$$\begin{bmatrix} \frac{\partial I}{\partial \tau_i} & \frac{\partial I}{\partial \omega_i} & \dots \end{bmatrix} \tag{8}$$

So we can get the derivatives of τ and γ by reading the first two columns of the results returned by LIDORT in the driver code.

In the most general case, to get the derivative with respect to γ would require us to pass in a column for every state variable S_i, putting us back at the roughly 100 times the cost of LIDORT without the Jacobians.

However, the calculation of the phase moments often has a much simpler form. We can often choose a set of intermediate variables such that:

$$\frac{\partial \gamma_{lk}}{\partial S_i} = \sum_m \frac{\partial \gamma_{lk}}{\partial V_m} \frac{\partial V_m}{\partial S_i} \tag{9}$$

If we can write this with number of m less than number of i, then we can use just these extra m columns (plus the 2 we have for τ and γ) to give us the full jacobian in equation (1). We then have:

$$W^{\gamma}_{lkj} = \begin{cases} 0 & k = 1 \text{ or } 2 \\ \frac{\partial \gamma_{lk}}{\partial V_{k-2}} & k > 2 \end{cases} \tag{10}$$

The software is purposely vague on exactly what V_m might be. We don't want the LIDORT driver code to make any assumption about how our Atmosphere code works. Instead, we will simply supply both Jacobians $\frac{\partial \gamma_{lk}}{\partial V_m}$ and $\frac{\partial V_m}{\partial S_i}$ to the driver code. The driver code will then get $\sum_k \frac{\partial I}{\partial \gamma_{lk}} \frac{\partial \gamma_{lk}}{\partial V_m}$ from LIDORT (as the last m columns returned). It will then multiple this result with the supplied matrix $\frac{\partial V_m}{\partial S_i}$ to get the results $\sum_k \frac{\partial I}{\partial \gamma_{lk}} \frac{\partial \gamma_{lk}}{\partial S_i}$ which can then be used in equation (4) to calculate the final desired Jacobian.

In the most general case, V_m would just be $\equiv S_i$, the second matrix would be the identify matrix, and the whole division would be pointless (but harmless).

However with our current Atmosphere model, we calculate the phase moment as the Rayleigh component plus a phase moment for each particle multiplied by a weighting. The phase moment for the particle is not dependent on the state vector, just the weighting is. So we can use this weighting as our set of V_m, so the total number of columns passed to LIDORT will be 2 + number particles.

It is possible that our Atmosphere model might get more complicated in the future, perhaps the phase moment might be a function of the state vector (e.g., we pass in the particle radius from the state vector to a simple Mie calculation). In that case we may need to come with another formulation for V_m, but as long as we can have a set with the number of m less than i, we are ahead by making this division.

One last point to avoid confusion. The LIDORT code doesn't directly take W^τ, W^γ, and W^ω. Rather it takes what it calls "normalized derivative inputs". This just means that we divide by τ, ω, and γ before passing these values to LIDORT.