Efficient computation of $\frac{\partial H}{\partial m}$ tensor for a 27-points stencil with mass lumping

Rafael Lago

September 5, 2014

First, we shall introduce some terminology. When we discretize the domain, each point relates itself with its 27 neighbours. In Figure 1 we show a 2D example for the sake of simplicity, but this is easily extended to the 3D case.

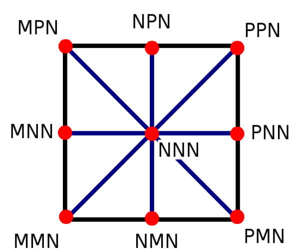

Figure 1: *Illustration of the stencil nomenclature in 2D. The 3D extension is trivial from this scheme.*

Here we denote the coefficient that relates $u_{(i,j,k)}$ with itself by $NNN_{(i,j,k)}$. The coefficient that relates $u_{(i,j,k)}$ with its neighbour to the right (traditionally written as $u_{(i+1,j,k)}$) here is denoted by $PNN_{(i,j,k)}$. Each point has its own stencil which relates the central point to its neighbours; therefore the point $u_{(i+1,j,k)}$ as well has a coefficient $NNN_{(i+1,j,k)}$ associated with itself, and the coefficient that relates $u_{(i+1,j,k)}$ with its left neighbour is $MNN_{(i+1,j,k)}$. Notice that $MNN_{(i+1,j,k)} \neq NNN_{(i,j,k)}$ although they are related to the same point $u_{i,j,k}$. That being said, M stands for a -1 and P stands for $+1$, while N stands for ± 0 in the mnemonics in Figure 1, the first letter representing the x direction, the second representing y and the last representing z (e.g. $MNP_{(i,j,k)}$ is the coefficient that relates the point $u_{(i,j,k)}$ to the point $u_{(i-1,j,k+1)}$). Later, each of these 27-points stencils will form a different line in the matrix.

In the basic 7-points stencil, the media parameter appears only in the NNN coefficients and nowhere else. What the mass lumping does (according to my understanding - which is pretty basic to be honest) is to "spray" the media parameter around. For instance, for the point $u_{(i,j,k)}$,the weight of the mass in the $NNN_{(i,j,k)}$ coefficient is scaled (based on a previously chosen constant

wm_1) and every stencil referring to the point $u_{(i,j,k)}$ will also receive a "part" of the media parameter. That is:

- $NNN_{(i,j,k)}$ receives $wm_1 \times m(i,j,k)$
- $MNN_{(i+1,j,k)}$ receives $wm_2 \times m(i,j,k)$
- $PNN_{(i-1,j,k)}$ receives $wm_2 \times m(i,j,k)$

and so on. From another perspective, the following also holds:

- $NNN_{(i,j,k)}$ receives $wm_1 \times m(i,j,k)$
- $MNN_{(i,j,k)}$ receives $wm_2 \times m(i-1,j,k)$
- $PNN_{(i,j,k)}$ receives $wm_2 \times m(i+1,j,k)$

The weight wm_2 is applied to direct neighbours (e.g. MNN, NPN, etc), the weight wm_3 is used for diagonal neighbours (e.g. MMN, NMP, etc) and the weight wm_4 is used for the corners in the 3D stencil (e.g. MMP, PMP, etc).

Let us first define the tensor $\dfrac{\partial H}{\partial m} \in \mathbb{C}^{n \times n} n$ using MATLAB notation as

$$\frac{\partial H}{\partial m}(:,:,i) = \frac{\partial H}{\partial m_i} \tag{1}$$

where each $\dfrac{\partial H}{\partial m_i} \in \mathbb{C}^{n \times n}$ is a matrix. Then

$$\frac{\partial H}{\partial m} u = \begin{bmatrix} \frac{\partial H}{\partial m_1} u & \frac{\partial H}{\partial m_2} u & \dots & \frac{\partial H}{\partial m_n} u \end{bmatrix} \tag{2}$$

and each $\dfrac{\partial H}{\partial m_i} u \in \mathbb{C}^n$ is a vector. In the 7-points stencil, each $\dfrac{\partial H}{\partial m_i}$ is a matrix whose the only non-zero lies in the $i-th$ row and the $i-th$ column. In the 27-points stencil with mass lumping, however, $\dfrac{\partial H}{\partial m_i}$ is a matrix with 27 non-zeros. Luckily, all of these non-zeros are located in the $i-th$ column. If e_i is the $i-th$ vector of the canonical base, then the non-zero pattern of the vector He_i and that of $\dfrac{\partial H}{\partial m_i} e_i$ is identical (although the values are not the same). With that being said, we infer that it should possible to store the tensor $\dfrac{\partial H}{\partial m}$ using exactly the storage requirement as H.

Knowing that the matrix $\dfrac{\partial H}{\partial m_i}$ contains exactly one non-zero column, we write

$$\frac{\partial H}{\partial m_i} u = \begin{bmatrix} 0 & \dots & a_{(1,i)} & \dots & 0 \\ 0 & \dots & a_{(2,i)} & \dots & 0 \\ \vdots & & \vdots & & \vdots \\ 0 & \dots & a_{(n,i)} & \dots & 0 \end{bmatrix} \begin{bmatrix} u_1 \\ u_2 \\ \vdots \\ u_n \end{bmatrix} = \begin{bmatrix} a_{(1,i)} \\ a_{(2,i)} \\ \vdots \\ a_{(n,i)} \end{bmatrix} u_i$$

Letting a_i denote the non-zero column of $\frac{\partial H}{\partial m_i}$, then

$$\frac{\partial H}{\partial m} u = \begin{bmatrix} a_1 u_1 & a_2 u_2 & \dots & a_n u_n \end{bmatrix}. \tag{3}$$

It might as well be useful to write this as

$$\frac{\partial H}{\partial m} u = \begin{bmatrix} a_1 & a_2 & \dots & a_n \end{bmatrix} * diag(u) \tag{4}$$

This matrix has exactly the same non-zero pattern as H, and thus, requires exactly the same storage. We can further compute

$$u^H \frac{\partial H}{\partial m}^H w = \begin{bmatrix} \bar{u}_1 a_1^H \\ \bar{u}_2 a_2^H \\ \dots \\ \bar{u}_n a_n^H \end{bmatrix} \begin{bmatrix} w_1 \\ w_2 \\ \vdots \\ w_n \end{bmatrix}$$

Since the non-zero pattern of $\frac{\partial H}{\partial m} u$ is the same as H, we can compute the above product by simply performing

$$\begin{aligned} dHt &= conj(Htransp(dH, idx)) \\ udHw &= conj(u). * Hmvp(dHt, idx, w); \end{aligned}$$

or equivalently:

$$\begin{aligned} ud &= spdiags(u, 0, n, n); \\ dH &= H2sparse(dH, idx); \\ udHw &= (dH * ud)' * w; \end{aligned}$$

which looks more straightforward but should take some few extra flops.

References

[1] S. Operto, J. Virieux, P. Amestoy, J.-Y. L'Excellent, L. Giraud, and H. B. H. Ali. 3d finite-difference frequency-domain modeling of visco-acoustic wave propagation using a massively parallel direct solver: A feasibility study. *Geophysics*, 72(5):SM195–SM211, 2007.