Discrete Adjoint for TITAN2D

H. AGHAKHAI

The aim of this report is to briefly show how the discrete adjoint for the system of Savage_Hutter partial differential equations is computed.

1 Mathematical Definition

1.1 Savage_Hutter Equation

To simulate granular pyroelastic flow, resulted from a volcanic avalanche TITAN2D solves Savage_Hutter Equation which is a modified version of Shallow Water (SW) equation presented in 1989 [1].

$$U_t + F(U)_x + G(U)_y = S(U) \tag{1}$$

Where:

$$U = (h, hv_x, hv_y)^T$$
$$F = (hv_x, hv_x^2 + 0.5k_{ap}g_z h^2, hv_x hv_y)^T$$
$$G = (hv_y, hv_x v_y, hv_y^2 + 0.5k_{ap}g_z h^2)^T$$
$$S = (0, S_x, S_y)^T$$
$$S_x = g_x h - \frac{V_x}{\sqrt{V_x^2 + V_y^2}}\left(g_z h + \frac{hV_x^2}{r_x}\right)\tan(\phi_{bed}) - hk_{ap}\mathrm{sgn}\left(\frac{\partial V_x}{\partial y}\right)\frac{\partial(g_z h)}{\partial y}\sin(\phi_{int})$$
$$S_y = g_y h - \frac{V_y}{\sqrt{V_x^2 + V_y^2}}\left(g_z h + \frac{hV_y^2}{r_y}\right)\tan(\phi_{bed}) - hk_{ap}\mathrm{sgn}\left(\frac{\partial V_y}{\partial x}\right)\frac{\partial(g_z h)}{\partial x}\sin(\phi_{int})$$

1.2 Adjoint definition and formulation

Let U and V be two vector spaces, and L be a linear operator that maps any $u \in U$ into $v \in V$. And $< \cdot, \cdot >$ be a bilinear map that maps any two vectors like u, v two a real number, $U \times V \to \mathbb{R}$. Then the adjoint operator, L^*, of L is defined:$< Lu, v >=< u, L^* v >$.
Given $R(U, \alpha)$ as a system of governing equations, where U is the solution vector, and α is the vector of design parameters.
The object is to minimize $\quad J(U, \alpha) \quad$ subject to $\quad R(U, \alpha) = 0$
So in optimization context we can say that we want to optimize the goal functional under the restriction of the governing equations. If we write the first variation of the functional and the governing equations for a discrete set of points, we will have:

$$\frac{dJ}{d\alpha} = \frac{\partial J}{\partial U}\frac{dU}{d\alpha} + \frac{\partial J}{\partial \alpha} \tag{2}$$

and:

$$\frac{\partial R}{\partial U}\frac{dU}{d\alpha} + \frac{\partial R}{\partial \alpha} = 0 \tag{3}$$

replacing $\frac{dU}{d\alpha}$ from the second equation into the first equation leads to:

$$\frac{dJ}{d\alpha} = -\frac{\partial J}{\partial U}(\frac{\partial R}{\partial U})^{-1}\frac{\partial R}{\partial \alpha} + \frac{\partial J}{\partial \alpha} \tag{4}$$

Previous equation shows that we can compute the sensitivity in two different ways:

1. Forward mode: first computes $(\frac{\partial R}{\partial U})^{-1}\frac{\partial R}{\partial \alpha}$
2. Adjoint mode: first computes $\frac{\partial J}{\partial U}(\frac{\partial R}{\partial U})^{-1}$

In the adjoint mode the gradient of functional is obtained by a forward solution of U, and one backward solution for the adjoint, and it is independent from α. Thus if the number of design parameters be greater than the number of objective functions, then the computational cost of the adjoint method is much lower than the forward method. To connect the above formulation with the adjoint concept, we can write:

$$\begin{aligned} u = \frac{dU}{d\alpha}, \qquad A &= \frac{\partial R}{\partial U} \\ g^T = \frac{\partial J}{\partial U}, \qquad f &= -\frac{\partial R}{\partial \alpha} \end{aligned} \tag{5}$$

Forward method:

$$\begin{aligned} \frac{dJ}{d\alpha} &= g^T u + \frac{\partial J}{\partial \alpha} \\ \text{Subject to} \quad Au &= f \end{aligned} \tag{6}$$

Adjoint Method:

$$\begin{aligned} \frac{dJ}{d\alpha} &= v^T f + \frac{\partial J}{\partial \alpha} \\ \text{Subject to} \quad A^T v &= g \end{aligned} \tag{7}$$

From the above we can see $< Au, v >=< u, A^T v >$,where v is the adjoint vector and is the solution of the following system of equations:

$$(\frac{\partial R}{\partial U})^T v = (\frac{\partial J}{\partial U})^T \tag{8}$$

2 Adjoint Computation

As shown in equation 8 the adjoint equation is:

$$(\frac{\partial R}{\partial U})^T v = (\frac{\partial J}{\partial U})^T$$

TITAN2D solves hyperbolic system of equations of Savege_Hutter, using Godunov scheme finite volume with HLL solver to compute the flux terms. The discritized form of the equations can be written:

$$U_i^{n+1} = U_i^n - \frac{\triangle t}{\triangle x}\{F^n_{i+\frac{1}{2}} - F^n_{i-\frac{1}{2}}\} - \frac{\triangle t}{\triangle y}\{G^n_{i+\frac{1}{2}} - G^n_{i-\frac{1}{2}}\} \tag{9}$$

$$(\frac{\partial R}{\partial U})^T_{m\times m} = K_{ij} \quad \text{which m is the number of time steps}$$

every K_{ij} is also a $n \times n$ matrix, which n is the number of elements. For the Godunov method:

$$K_{i,i} = I \quad \text{and} \quad K_{i,i+1} = (\frac{\partial R_p^{i+1}}{\partial U_q^i})^T \quad \& \quad \text{rest the of elements} = 0$$

The 2nd term is the sensitivity of the residual vector in time step (i+1) with respect to the state variables in time step i, that has to be evaluated at element p with respect to element q, consequently:

$$(\frac{\partial R}{\partial U})^T_{m\times m} = \begin{pmatrix} I & K_{1,2} & & \\ & I & K_{2,3} & 0 \\ & & \ddots & \ddots \\ & 0 & & I \quad K_{m-1,m} \\ & & & I \end{pmatrix} \tag{10}$$

Taking to account that in numerical methods the solution at each point only depends on the neighbor points so every $K_{i,i+1}$ is also a block banded matrix. We used first order derivative so the state variables in each elements only depends on its neighbors.

$$\begin{aligned} v_1 + K_{1,2}v_2 &= \left(\frac{\partial J}{\partial U}\right)^T_1 \\ &\vdots \\ v_{m-1} + K_{m-1,m}v_m &= \left(\frac{\partial J}{\partial U}\right)^T_{m-1} \\ v_m &= \left(\frac{\partial J}{\partial U}\right)^T_m \end{aligned} \tag{11}$$

The previous equation means that to compute the Jacobian matrices, the solution vectors for all of the elements and for all of time steps have to be stored. Then we can compute the adjoint in a reverse time order. Since it is impossible to store all of these matrices in the memory, people use dynamic check pointing schemes and appropriate parallel I/O to overcome these difficulties.

3 Error Estimation

Suppose that $J(Q)$, $J(Q_h)$ and $J(Q_H)$ are respectively exact solution, numerical solution on fine mesh, and numerical solution on a coarse solution of objective

functional. We want to minimize the numerical error of objective functional $|J(Q) - J(Q_H)|$. Following the steps described in [**?**] we can write:

$$J(Q_h) \approx J(Q_h^H) - \underbrace{(\psi_h^H)^T R(Q_h^H)}_{\text{Adjoint correction term}} - \underbrace{(\psi_h - \psi_h^H)^T R(Q_h^H)}_{\text{Remaining term}}$$

where Q_h^H and ψ_h^H are reconstruction of the flow and adjoint solution from coarse mesh to embedded mesh, we use linear reconstruction to approximate ψ_h and Q_h, and use constant reconstruction to approximate ψ_h^H and show them respectively with ψ_L and ψ_C, so the above equation changes to

$$J(Q_h) \approx J(Q_L) - (\psi_L)^T R(Q_L) - (\psi_L - \psi_C)^T R(Q_L) \tag{12}$$

So the total error is going to be

$$e_k = \sum |(\psi_L - \psi_C)^T R(Q_L)| \tag{13}$$

Given a threshold for the functional, TOL, so the local error parameter is going to be

$$\frac{TOL}{N} \tag{14}$$

then we can normalize the error in the element by

$$r_k = \frac{e_k}{t} \tag{15}$$

4 Computer Programming

For sake of simplicity and for the first phase We implemented the above computations without Adaptive Mesh Refinement (AMR). To store the solutions we created Jacobian class that stores the all required information including the vector of solutions. We also created a vector in Element class that keeps the address of Jacobian class. So Jacobian class is created at each time step for each element, stores the solution vector and its address is maintained in the corresponding element. The reason that we did not used the same element class for storing the data is that in forward run we do not need the solutions and if we do that we have to pay a huge not necessary communication cost. With this method at the end of forward run we have the address of all the data that we need to compute the adjoint, and we just have to retrieve them from the memory to compute the jacobian. In the reverse mode, after using the required data of the jacobian class we will not need them anymore and we can free memory from them.

References

[1] S.B. Savage and K. Hutter. The motion of a finite mass of granular material down a rough incline. *Journal of Fluid Mechanics*, 199:177–215, 1989.