Automatic Differentiation for Inverse Modeling Notes

Toby Harvey

December 8, 2025

1 Introduction

Some parameters in ice sheet models are hard if not impossible to observe. One strategy for estimating these parameters is to systematically adjust them so that outputs of the model best fit data that is observable. A cost function $\mathcal{J}(\alpha, u^{obs}, \hat{u}(\alpha))$, where α is the model parameter, is used to quantify the error between the model outputs $\hat{u}$ and the observable data u^{obs}. Most methods of minimizing $\mathcal{J}$ (gradient decent, etc...) require taking the derivative of the cost function with respect to the parameter, $\frac{d\mathcal{J}}{d\alpha}$. By the chain rule this is computed as:

$$\frac{d\mathcal{J}}{d\alpha} = \frac{\partial \mathcal{J}}{\partial \alpha} + \frac{\partial \mathcal{J}}{\partial \hat{u}} \frac{\partial \hat{u}}{\partial \alpha}.$$

The first two derivatives are easy to compute analytically, but the 3rd one is much harder. One possible solution is to try to compute it via perturbing the parameter of interest slightly and taking a finite difference:

$$\frac{\partial \hat{u}}{\partial \alpha} \approx \frac{\hat{u}^{pert} - \hat{u}}{\alpha^{pert} - \alpha}.$$

This requires running your model at least twice to compute a gradient for a single parameter, and in the case of a spatially variable parameter at least n times where n is the number of spatial degrees of freedom. The adjoint method is a method that requires more mathematical machineary, but requires solving only one additional model/system. The problem here is that it can be very difficult, if not impossible to derive this system. A third technique, automatic differentiation (AD), is summerized here. It is automatic in the sense that nothing coming from the specfic model system has to be derived, although it is more expensive than the adjoint method. First I describe automatic differentiation usinng a toy example, then explain how it is applied to a full forward model code.

2 Automatic Differentiation of a Simple Code/Function

Lets say we write a code that computes a very simple function:

$$y = f(x_1, x_2, x_3) = x_1 x_2 + \sin(x_3) - \log(x_1).$$

at its essence, regardless of language or computer architecture, we need to follow the dependency graph in figure 1. If we take this as our primary (think forward model) code, we can have a secondary code that, at runtime builds, the dependancy graph in the background. This is the AD code.

3 Automatic Differentiation of a Forward Model

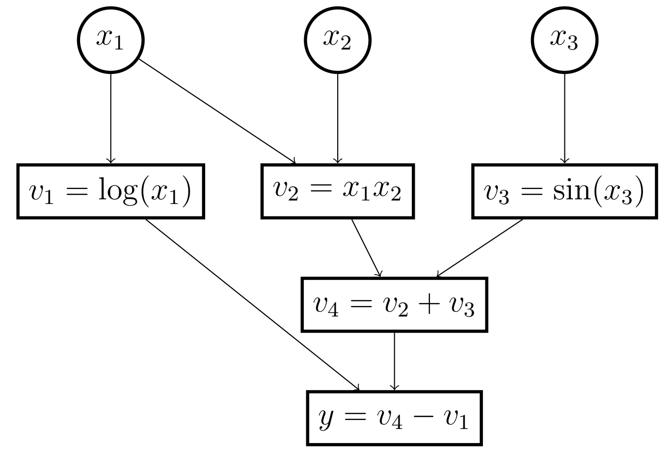

Figure 1: computational dependency graph of $y = x_1x_2 + \sin(x_3) - \log(x_1)$.