ODE-Constrained Optimization with Automatic Differentiation

Tianju Xue

December 11, 2022

1 Background

This tutorial discusses the formulation of solving ODE-constrained optimization (ODE-CO) problems with the adjoint method. The tutorial comes with an example about the famous Lorenz system, implemented in this Jupyter Notebook. The code is based on JAX with its handy Automatic Differentiation feature that makes life much easier.

2 ODE-constrained optimization

Assume $\boldsymbol{u}(t, \mathbf{a}, \boldsymbol{b}) \in \mathbb{R}^k$ is the variable we want to solve with $\mathbf{a} \in \mathbb{R}^m$ being the ODE parameters and $\boldsymbol{b} \in \mathbb{R}^k$ being the initial condition. Note that t, $\mathbf{a}$, and $\boldsymbol{b}$ are independent variables, and determine the value of $\boldsymbol{u}$. The forward problem is defined as

$$\frac{\mathrm{d}}{\mathrm{d}t}\boldsymbol{u}(t, \mathbf{a}, \boldsymbol{b}) = \boldsymbol{r}(\boldsymbol{u}, t, \mathbf{a}), \tag{1a}$$

$$\boldsymbol{u}(t_0, \mathbf{a}, \boldsymbol{b}) = \boldsymbol{b}, \tag{1b}$$

which is an ODE system. In some cases, this ODE system is just the semi-discretized system of the Finite Element/Finite difference method with spatial discretization performed but time discretization not yet performed (e.g., method of lines).

We are interested in solving the inverse problem or the design problem. In plain explanation, an inverse problem is when you have certain desired requirement for the solution $\boldsymbol{u}$ and you want to figure out what values of $\mathbf{a}$ and $\boldsymbol{b}$ fulfill the requirement. Mathematically, we have an optimization problem of the following form:

$$\min_{\mathbf{a} \in \mathbb{R}^m, \boldsymbol{b} \in \mathbb{R}^k} \mathcal{J}(\boldsymbol{u}) = h(\boldsymbol{u}(t_f, \mathbf{a}, \boldsymbol{b})) + \int_{t_0}^{t_f} g(\boldsymbol{u}(t, \mathbf{a}, \boldsymbol{b}))\mathrm{d}t, \tag{2a}$$

$$\text{s.t.} \quad \frac{\mathrm{d}}{\mathrm{d}t}\boldsymbol{u}(t, \mathbf{a}, \boldsymbol{b}) = \boldsymbol{r}(\boldsymbol{u}, t, \mathbf{a}), \tag{2b}$$

$$\boldsymbol{u}(t_0, \mathbf{a}, \boldsymbol{b}) = \boldsymbol{b}, \tag{2c}$$

where $\mathcal{J} : \mathcal{U} \to \mathbb{R}$ is the objective functional, $h : \mathbb{R}^k \to \mathbb{R}$, and $g : \mathbb{R}^k \to \mathbb{R}$.

Since $\boldsymbol{u}$ is implicitly determined by $\mathbf{a}$ and $\boldsymbol{b}$ via the ODE system, for a fixed set of $\mathbf{a}$ and $\boldsymbol{b}$ we have a deterministic $\mathcal{J}$. Therefore, it makes sense to consider the total derivative of $\mathcal{J}$ with respect to $\mathbf{a}$ and $\boldsymbol{b}$. In fact, the central goal of this tutorial is to show how to compute

$$\frac{\mathrm{d}\mathcal{J}}{\mathrm{d}\mathbf{a}} \quad \text{and} \quad \frac{\mathrm{d}\mathcal{J}}{\mathrm{d}\boldsymbol{b}}, \tag{3}$$

so that gradient-based optimization algorithms can be employed to solve the ODE-CO problem. We will introduce two approaches to compute $\frac{\mathrm{d}\mathcal{J}}{\mathrm{d}\mathbf{a}}$ and $\frac{\mathrm{d}\mathcal{J}}{\mathrm{d}\boldsymbol{b}}$. Depending on when the time discretization happens, we refer to these two approaches as "early discretization" and "late discretization", respectively.

3 Early discretization (discrete adjoint method)

This part is based on [1]. Let us immediately discretize the ODE system (1) in time so that

$$\boldsymbol{u}^n = \boldsymbol{f}(\boldsymbol{u}^{n-1}, n, \mathbf{a}) := \boldsymbol{f}^n, \quad \boldsymbol{u}^0 = \boldsymbol{b}, \tag{4}$$

where n is the time step number, and $\boldsymbol{f}$ is any explicit ODE integrator, e.g., explicit Euler method or Runge-Kutta method.

The objective function is

$$J = h(\boldsymbol{u}^N), \tag{5}$$

where we have omitted the g term for simplicity.

With chain rule, we have

$$\frac{\mathrm{d}J}{\mathrm{d}\mathbf{a}} = \frac{\mathrm{d}h}{\mathrm{d}\boldsymbol{u}^N}\left\{\frac{\partial \boldsymbol{f}^N}{\partial \mathbf{a}} + \frac{\partial \boldsymbol{f}^N}{\partial \boldsymbol{u}^{N-1}}\left[\frac{\partial \boldsymbol{f}^{N-1}}{\partial \mathbf{a}} + \frac{\partial \boldsymbol{f}^{N-1}}{\partial \boldsymbol{u}^{N-2}}\left(\frac{\partial \boldsymbol{f}^{N-2}}{\partial \mathbf{a}} + \dots\right)\right]\right\}, \tag{6}$$

and

$$\begin{aligned}\frac{\mathrm{d}J}{\mathrm{d}\boldsymbol{b}} &= \frac{\mathrm{d}h}{\mathrm{d}\boldsymbol{u}^N}\left\{\frac{\partial \boldsymbol{f}^N}{\partial \boldsymbol{b}} + \frac{\partial \boldsymbol{f}^N}{\partial \boldsymbol{u}^{N-1}}\left[\frac{\partial \boldsymbol{f}^{N-1}}{\partial \boldsymbol{b}} + \frac{\partial \boldsymbol{f}^{N-1}}{\partial \boldsymbol{u}^{N-2}}\left(\frac{\partial \boldsymbol{f}^{N-2}}{\partial \boldsymbol{b}} + \dots\right)\right]\right\} &(7)\\ &= \frac{\mathrm{d}h}{\mathrm{d}\boldsymbol{u}^N}\frac{\partial \boldsymbol{f}^N}{\partial \boldsymbol{u}^{N-1}}\frac{\partial \boldsymbol{f}^{N-1}}{\partial \boldsymbol{u}^{N-2}}\cdots\frac{\partial \boldsymbol{f}^1}{\partial \boldsymbol{u}^0}\frac{\mathrm{d}\boldsymbol{u}^0}{\mathrm{d}\boldsymbol{b}}. &(8)\end{aligned}$$

Let us define adjoint variable $\boldsymbol{\lambda}_{\boldsymbol{b}} \in \mathbb{R}^k$ associated with initial condition $\boldsymbol{b}$ so that

$$\boldsymbol{\lambda}_{\boldsymbol{b}}^{n-1} = \boldsymbol{\lambda}_{\boldsymbol{b}}^n \frac{\partial \boldsymbol{f}^n}{\partial \boldsymbol{u}^{n-1}}, \quad \boldsymbol{\lambda}_{\boldsymbol{b}}^N = \frac{\mathrm{d}h}{\mathrm{d}\boldsymbol{u}^N}, \tag{9}$$

and similarly the adjoint variable $\boldsymbol{\lambda}_{\mathbf{a}} \in \mathbb{R}^m$ associated with ODE parameters $\mathbf{a}$ so that

$$\boldsymbol{\lambda}_{\mathbf{a}}^{n-1} = \boldsymbol{\lambda}_{\boldsymbol{b}}^n \frac{\partial \boldsymbol{f}^n}{\partial \mathbf{a}} + \boldsymbol{\lambda}_{\mathbf{a}}^n, \quad \boldsymbol{\lambda}_{\mathbf{a}}^N = \mathbf{0}. \tag{10}$$

With some tedious algebraic operations, we have

$$\frac{\mathrm{d}J}{\mathrm{d}\mathbf{a}} = \boldsymbol{\lambda}_{\mathbf{a}}^0 \quad \text{and} \quad \frac{\mathrm{d}J}{\mathrm{d}\boldsymbol{b}} = \boldsymbol{\lambda}_{\boldsymbol{b}}^0. \tag{11}$$

It is important to note that since the forward problem, i.e., Eq. (4) is not reversible (at least not so explicit to compute $\boldsymbol{u}^{n-1}$ from $\boldsymbol{u}^n$), we have to store the entire trajectories of $\boldsymbol{u}_0, \boldsymbol{u}_1, \dots, \boldsymbol{u}^N$ in memory because they are required in the adjoint updates of Eq. (9) and Eq. (10). Therefore this is a linear memory approach, and does not scale well if the ODE system is large.

4 Late discretization (continuous adjoint method)

This part is based on the Neural-ODE paper [2], the associated tutorial, and the nice Youtube video by Chris H. Rycroft at Harvard.

We will delay the discretization of the ODE system. In stead, first notice that by Eq. 2b we have

$$\frac{\mathrm{d}}{\mathrm{d}t}\left(\frac{\mathrm{d}\boldsymbol{u}}{\mathrm{d}\mathbf{a}}\right) = \frac{\mathrm{d}\boldsymbol{r}}{\mathrm{d}\mathbf{a}}, \quad \frac{\mathrm{d}}{\mathrm{d}t}\left(\frac{\mathrm{d}\boldsymbol{u}}{\mathrm{d}\boldsymbol{b}}\right) = \frac{\mathrm{d}\boldsymbol{r}}{\mathrm{d}\boldsymbol{b}} = \mathbf{0}. \tag{12}$$

Then consider the adjoint variable $\boldsymbol{\lambda}_{\mathbf{a}}(t) \in \mathbb{R}^m$ and $\boldsymbol{\lambda}_{\boldsymbol{b}}(t) \in \mathbb{R}^k$ that satisfy the adjoint ODE system:

$$\frac{\mathrm{d}}{\mathrm{d}t}\begin{bmatrix}\boldsymbol{\lambda}_{\mathbf{a}}\\ \boldsymbol{\lambda}_{\boldsymbol{b}}\end{bmatrix} = \begin{bmatrix} -\boldsymbol{\lambda}_{\boldsymbol{b}}\frac{\partial \boldsymbol{r}}{\partial \mathbf{a}} \\ -\boldsymbol{\lambda}_{\boldsymbol{b}}\frac{\partial \boldsymbol{r}}{\partial \boldsymbol{u}} - \dfrac{\partial g}{\partial \boldsymbol{u}}\end{bmatrix}, \tag{13}$$

$$\begin{bmatrix}\boldsymbol{\lambda}_{\mathbf{a}}(t_f)\\ \boldsymbol{\lambda}_{\boldsymbol{b}}(t_f)\end{bmatrix} = \begin{bmatrix} \mathbf{0} \\ \left.\dfrac{\partial h}{\partial \boldsymbol{u}}\right|_{t=t_f}\end{bmatrix}, \tag{14}$$

where $\boldsymbol{\lambda}_{\mathbf{a}}$ and $\boldsymbol{\lambda}_{\boldsymbol{b}}$ are solved backwards in time.

We have

$$\begin{aligned}
\frac{\mathrm{d}\mathcal{J}}{\mathrm{d}\mathbf{a}} &= \left.\left(\frac{\partial h}{\partial \boldsymbol{u}}\frac{\mathrm{d}\boldsymbol{u}}{\mathrm{d}\mathbf{a}}\right)\right|_{t=t_f} + \int_{t_0}^{t_f}\frac{\partial g}{\partial \boldsymbol{u}}\frac{\mathrm{d}\boldsymbol{u}}{\mathrm{d}\mathbf{a}}\mathrm{d}t \qquad (15)\\
&= \boldsymbol{\lambda}_{\boldsymbol{b}}(t_f)\left.\frac{\mathrm{d}\boldsymbol{u}}{\mathrm{d}\mathbf{a}}\right|_{t=t_f} + \int_{t_0}^{t_f}\frac{\partial g}{\partial \boldsymbol{u}}\frac{\mathrm{d}\boldsymbol{u}}{\mathrm{d}\mathbf{a}}\mathrm{d}t\\
&= \boldsymbol{\lambda}_{\boldsymbol{b}}(t_0)\left.\frac{\mathrm{d}\boldsymbol{u}}{\mathrm{d}\mathbf{a}}\right|_{t=t_0} + \int_{t_0}^{t_f}\left(\frac{\mathrm{d}\boldsymbol{\lambda}_{\boldsymbol{b}}}{\mathrm{d}t}\frac{\mathrm{d}\boldsymbol{u}}{\mathrm{d}\mathbf{a}} + \boldsymbol{\lambda}_{\boldsymbol{b}}\frac{\mathrm{d}}{\mathrm{d}t}\left(\frac{\mathrm{d}\boldsymbol{u}}{\mathrm{d}\mathbf{a}}\right) + \frac{\partial g}{\partial \boldsymbol{u}}\frac{\mathrm{d}\boldsymbol{u}}{\mathrm{d}\mathbf{a}}\right)\mathrm{d}t\\
&= \boldsymbol{\lambda}_{\boldsymbol{b}}(t_0)\frac{\mathrm{d}\boldsymbol{b}}{\mathrm{d}\mathbf{a}} + \int_{t_0}^{t_f}\boldsymbol{\lambda}_{\boldsymbol{b}}\frac{\partial \boldsymbol{r}}{\partial \mathbf{a}}\mathrm{d}t \qquad (16)\\
&= \boldsymbol{\lambda}_{\mathbf{a}}(t_0), \qquad (17)
\end{aligned}$$

and similarly,

$$\begin{aligned}
\frac{\mathrm{d}\mathcal{J}}{\mathrm{d}\boldsymbol{b}} &= \left.\left(\frac{\partial h}{\partial \boldsymbol{u}}\frac{\mathrm{d}\boldsymbol{u}}{\mathrm{d}\boldsymbol{b}}\right)\right|_{t=t_f} + \int_{t_0}^{t_f}\frac{\partial g}{\partial \boldsymbol{u}}\frac{\mathrm{d}\boldsymbol{u}}{\mathrm{d}\boldsymbol{b}}\mathrm{d}t \qquad (18)\\
&= \boldsymbol{\lambda}_{\boldsymbol{b}}(t_f)\left.\frac{\mathrm{d}\boldsymbol{u}}{\mathrm{d}\boldsymbol{b}}\right|_{t=t_f} + \int_{t_0}^{t_f}\frac{\partial g}{\partial \boldsymbol{u}}\frac{\mathrm{d}\boldsymbol{u}}{\mathrm{d}\boldsymbol{b}}\mathrm{d}t\\
&= \boldsymbol{\lambda}_{\boldsymbol{b}}(t_0)\left.\frac{\mathrm{d}\boldsymbol{u}}{\mathrm{d}\boldsymbol{b}}\right|_{t=t_0} + \int_{t_0}^{t_f}\left(\frac{\mathrm{d}\boldsymbol{\lambda}_{\boldsymbol{b}}}{\mathrm{d}t}\frac{\mathrm{d}\boldsymbol{u}}{\mathrm{d}\boldsymbol{b}} + \boldsymbol{\lambda}_{\boldsymbol{b}}\frac{\mathrm{d}}{\mathrm{d}t}\left(\frac{\mathrm{d}\boldsymbol{u}}{\mathrm{d}\boldsymbol{b}}\right) + \frac{\partial g}{\partial \boldsymbol{u}}\frac{\mathrm{d}\boldsymbol{u}}{\mathrm{d}\boldsymbol{b}}\right)\mathrm{d}t\\
&= \boldsymbol{\lambda}_{\boldsymbol{b}}(t_0)\frac{\mathrm{d}\boldsymbol{b}}{\mathrm{d}\boldsymbol{b}} + \int_{t_0}^{t_f}\boldsymbol{\lambda}_{\boldsymbol{b}}\frac{\partial \boldsymbol{r}}{\partial \boldsymbol{b}}\mathrm{d}t \qquad (19)\\
&= \boldsymbol{\lambda}_{\boldsymbol{b}}(t_0), \qquad (20)
\end{aligned}$$

where integration by part has been used.

Because the ODE system is reversible, we do not need to store the forward solutions $\boldsymbol{u}$ in memory. Instead, when solving the adjoint system (13), we can simultaneously solve $\boldsymbol{u}$ backward in time starting from $\boldsymbol{u}(t_f)$. Therefore the continuous adjoint method can be trivially implemented with constant memory, which is important for large scale problems.

5 The role of automatic differentiation

In both discrete and continuous adjoint approaches, we need to compute something like $\boldsymbol{\lambda}\frac{\partial \boldsymbol{f}}{\partial \boldsymbol{u}}$ which is essentially vector-Jacobian product (VJP). We use JAX to compute these VJPs automatically for us.

5.1 Implementing JVP and VJP

The ODE-constrained optimization problem:

$$\min_{\mathbf{a}\in\mathbb{R}^l, \boldsymbol{b}\in\mathbb{R}^k} J(\mathbf{a}, \boldsymbol{b}) = h(\boldsymbol{u}(t_f, \mathbf{a}, \boldsymbol{b})), \tag{21a}$$

$$\text{s.t.} \quad \frac{\mathrm{d}}{\mathrm{d}t}\boldsymbol{u}(t, \mathbf{a}, \boldsymbol{b}) = \boldsymbol{r}(\boldsymbol{u}, \mathbf{a}), \tag{21b}$$

$$\boldsymbol{u}(t_0, \mathbf{a}, \boldsymbol{b}) = \boldsymbol{b}, \tag{21c}$$

The goal is to compute the gradient $\frac{\partial J}{\partial \mathbf{a}}$ and $\frac{\partial J}{\partial \boldsymbol{b}}$.

5.1.1 JVP

Given perturbations $\Delta\mathbf{a} \in \mathbb{R}^l$ and $\Delta\boldsymbol{b} \in \mathbb{R}^k$, we define a new state variable $\boldsymbol{z} = \frac{\partial \boldsymbol{u}}{\partial \mathbf{a}}\Delta\mathbf{a} + \frac{\partial \boldsymbol{u}}{\partial \boldsymbol{b}}\Delta\boldsymbol{b}$. We have $\boldsymbol{z} \in \mathbb{R}^k$. The goal is to compute $\boldsymbol{z}(t_f, \mathbf{a}, \boldsymbol{b})$. The ODE for $\boldsymbol{z}$ is

$$\frac{\mathrm{d}}{\mathrm{d}t}\boldsymbol{z}(t, \mathbf{a}, \boldsymbol{b}) = \frac{\partial \boldsymbol{r}}{\partial \boldsymbol{u}}\boldsymbol{z} + \frac{\partial \boldsymbol{r}}{\partial \mathbf{a}}\Delta\mathbf{a}, \tag{22}$$

$$\boldsymbol{z}(t_0, \mathbf{a}, \boldsymbol{b}) = \Delta\boldsymbol{b}. \tag{23}$$

5.1.2 VJP

Given perturbation $\boldsymbol{g} \in \mathbb{R}^k$, we define the adjoint variables $\boldsymbol{\lambda}_{\mathbf{a}} \in \mathbb{R}^l$ and $\boldsymbol{\lambda}_{\boldsymbol{b}} \in \mathbb{R}^k$. The goal is to compute $\boldsymbol{g}\frac{\partial \boldsymbol{u}(t_f, \mathbf{a}, \boldsymbol{b})}{\partial \mathbf{a}}$ and $\boldsymbol{g}\frac{\partial \boldsymbol{u}(t_f, \mathbf{a}, \boldsymbol{b})}{\partial \boldsymbol{b}}$. The ODE for $[\boldsymbol{\lambda}_{\mathbf{a}}, \boldsymbol{\lambda}_{\boldsymbol{b}}]$ is

$$\frac{\mathrm{d}}{\mathrm{d}t}\begin{bmatrix}\boldsymbol{\lambda}_{\mathbf{a}} \\ \boldsymbol{\lambda}_{\boldsymbol{b}}\end{bmatrix} = \begin{bmatrix}-\boldsymbol{\lambda}_{\boldsymbol{b}}\frac{\partial \boldsymbol{r}}{\partial \mathbf{a}} \\ -\boldsymbol{\lambda}_{\boldsymbol{b}}\frac{\partial \boldsymbol{r}}{\partial \boldsymbol{u}}\end{bmatrix}, \tag{24}$$

$$\begin{bmatrix}\boldsymbol{\lambda}_{\mathbf{a}}(t_f) \\ \boldsymbol{\lambda}_{\boldsymbol{b}}(t_f)\end{bmatrix} = \begin{bmatrix}\mathbf{0} \\ \boldsymbol{g}\end{bmatrix}. \tag{25}$$

The adjoint variables do have interpretations [2]. It can be helpful to think about an objective function defined as $L = \boldsymbol{g} \cdot \boldsymbol{u}(t_f, \mathbf{a}, \boldsymbol{b})$.

References

[1] S. G. Johnson, "Adjoint methods and sensitivity analysis for recurrence relations," *Course notes for MIT' s*, vol. 18, pp. 2007–2011, 2007.

[2] R. T. Chen, Y. Rubanova, J. Bettencourt, and D. Duvenaud, "Neural ordinary differential equations," *arXiv preprint arXiv:1806.07366*, 2018.