The Discrete Adjoint Method: Efficient Derivatives for Functions of Discrete Sequences

Michael Betancourt, Charles C. Margossian, Vianey Leos-Barajas

Abstract. Gradient-based techniques are becoming increasingly critical in quantitative fields, notably in statistics and computer science. The utility of these techniques, however, ultimately depends on how efficiently we can evaluate the derivatives of the complex mathematical functions that arise in applications. In this paper we introduce a discrete adjoint method that efficiently evaluates derivatives for functions of discrete sequences.

Many popular mathematical models, such as common hidden Markov models, utilize sequences of discrete states implicitly defined through *forward difference equations*,

$$\mathbf{u}_{n+1} - \mathbf{u}_n = \boldsymbol{\Delta}_n(\mathbf{u}_n, \psi, n),$$

to capture the regular evolution of a latent system; here $\mathbf{u}_n$ denotes the nth latent state of the system and ψ the model parameters. Typically these sequences are incorporated into larger models through *discrete functionals* that consume particular sequences and return scalar values,

$$\mathcal{J}(\psi) = \sum_{n=0}^{N-1} j_n(\mathbf{u}_n, \psi, n).$$

We can quantify the impact of the parameters, ψ, on these functionals by evaluating the total derivatives, $\mathrm{d}\mathcal{J}/\mathrm{d}\psi$. The evaluation of these derivatives is complicated by the dependence of the sequences on the parameters enforced by the forward difference equations; the total derivative of a functional has to take into account both the *explicit* dependence of the j_n on ψ and also the *implicit* dependence mediated by the latent states $\mathbf{u}_n$.

Michael Betancourt is the principal research scientist at Symplectomorphic, LLC. (e-mail: betan@symplectomorphic.com). Charles Margossian is a PhD candidate in the Department of Statistics, Columbia University. Vianey Leos-Barajas is a postdoctoral researcher in the Department of Forestry and Environmental Resources and the Department of Statistics at North Carolina State University.

We can always compute each *sensitivity*, $\mathrm{d}\mathbf{u}_n / \mathrm{d}\psi$, by propagating derivatives along the forward difference equations and constructing the corresponding sequence of sensitivities. This quickly becomes expensive, however, when there are many parameters that each require their own sensitivities. In order to better scale we need to bypass the superfluous computation of these intermediate derivatives and only propagate the minimal information needed to construct the total derivatives of the desired functionals.

In this paper we introduce a *discrete adjoint* technique that efficiently computes total derivatives without explicitly calculating intermediate sensitivities. We begin by reviewing the powerful continuous adjoint method for ordinary differential equations before deriving a discrete analog. Finally we demonstrate how the method can be applied to hidden Markov models.

1. CONTINUOUS ADJOINT SYSTEMS

The continuous analog of discrete sequences are *state trajectories*, $\mathbf{u}(t)$, defined implicitly through the ordinary differential equations

$$\frac{\mathrm{d}\mathbf{u}}{\mathrm{d}t} = \mathbf{f}(\mathbf{u}, \psi, t)$$

along with the initial conditions

$$\mathbf{u}(t = 0) = \boldsymbol{v}(\psi).$$

A *functional* consumes the state trajectory and returns a single real number through an integration over time,

$$\mathcal{J}(\psi) = \int_0^T \mathrm{d}t \, j(\mathbf{u}, \psi, t).$$

Our goal is then to compute the *total* derivative of $\mathcal{J}$ with respect to the parameter ψ, taking into account not only the explicit dependence of ψ on j but also the implicit dependence through the influence of ψ on the evolution of the states $\mathbf{u}(t)$. For a thorough review of the possible strategies see Section 2.6 and 2.7 of Hindmarsh and Serban (2020).

1.1 Adjoint Task Force

An immediate way to compute gradients of functionals like this is to explicitly compute the state sensitivities

$$\boldsymbol{\eta} = \mathrm{d}\mathbf{u} / \mathrm{d}\psi$$

by solving the auxiliary ordinary differential equations,

$$\begin{aligned}\frac{\mathrm{d}\boldsymbol{\eta}}{\mathrm{d}t} &= \frac{\mathrm{d}}{\mathrm{d}t}\left(\frac{\mathrm{d}\mathbf{u}}{\mathrm{d}\psi}\right) \\ &= \frac{\mathrm{d}}{\mathrm{d}\psi}\left(\frac{\mathrm{d}\mathbf{u}}{\mathrm{d}t}\right) \\ &= \frac{\mathrm{d}}{\mathrm{d}\psi}\left(\mathbf{f}\right) \\ &= \frac{\partial \mathbf{f}}{\partial \psi} + \left(\frac{\mathrm{d}\mathbf{f}}{\mathrm{d}\mathbf{u}}\right)^{\dagger} \cdot \frac{\mathrm{d}\mathbf{u}}{\mathrm{d}\psi} \\ &= \frac{\partial \mathbf{f}}{\partial \psi} + \left(\frac{\mathrm{d}\mathbf{f}}{\mathrm{d}\mathbf{u}}\right)^{\dagger} \cdot \boldsymbol{\eta},\end{aligned}$$

Here a boldfaced fraction is shorthand for the Jacobian matrix

$$\left(\frac{\mathrm{d}\mathbf{f}}{\mathrm{d}\mathbf{u}}\right)_{ij} = \frac{\mathrm{d}f_i}{\mathrm{d}u_j}.$$

Once we've solved for the state sensitivities we can construct the total derivative of the desired functional through the chain rule,

$$\begin{aligned}\frac{\mathrm{d}\mathcal{J}}{\mathrm{d}\psi}(\psi) &= \frac{\mathrm{d}}{\mathrm{d}\psi}\int_0^T \mathrm{d}t\, j(\mathbf{u}, \psi, t) \\ &= \int_0^T \mathrm{d}t\, \frac{\mathrm{d}}{\mathrm{d}\psi} j(\mathbf{u}, \psi, t) \\ &= \int_0^T \mathrm{d}t\, \left[\frac{\partial j}{\partial \psi} + \left(\frac{\mathrm{d}j}{\mathrm{d}\mathbf{u}}\right)^{\dagger} \cdot \boldsymbol{\eta}\right].\end{aligned}$$

This approach becomes burdensome, however, once we consider multiple parameters and hence multiple total derivatives, each of which requires integrating over its own trajectory of sensitivities.

Another way to work out the total derivative of the functional is to treat the influence of the parameter on the state trajectory as *constraints* (Hannemann-Tamás, Muñoz and Marquardt, 2015),

$$\begin{aligned}0 &= \mathbf{u}(0) - \boldsymbol{v}(\psi) \\ 0 &= \frac{\mathrm{d}\mathbf{u}}{\mathrm{d}t} - \mathbf{f}(\mathbf{u}, \psi, t),\end{aligned}$$

which are explicitly incorporated into the functional with *Lagrange multipliers*, $\boldsymbol{\mu}$ and $\boldsymbol{\lambda}(t)$,

$$\begin{aligned}
\mathcal{J}(\psi) &= \int_0^T \mathrm{d}t\, j(\mathbf{u}, \psi, t) \\
&= 0 + \int_0^T \mathrm{d}t\, j(\mathbf{u}, \psi, t) + 0 \\
&= \boldsymbol{\mu}^\dagger \cdot [\mathbf{u}(0) - \boldsymbol{\upsilon}(\psi)] + \int_0^T \mathrm{d}t\, j(\mathbf{u}, \psi, t) + \boldsymbol{\lambda}^\dagger(t) \cdot \left[\frac{\mathrm{d}\mathbf{u}}{\mathrm{d}t} - \mathbf{f}(\mathbf{u}, \psi, t) \right] \\
&\equiv \mathcal{L}(\psi).
\end{aligned}$$

As long as the constraints are satisfied this modified functional will equal our target functional for *any* values of the Lagrange multipliers.

Under these constraints we can compute the total derivative of the functional by instead differentiating this modified functional. If we assume that everything is smooth then we can exchange the order of integration and differentiation to give

$$\begin{aligned}
\frac{\mathrm{d}\mathcal{J}}{\mathrm{d}\psi} &= \frac{\mathrm{d}\mathcal{L}}{\mathrm{d}\psi} \\
&= \boldsymbol{\mu}^\dagger \cdot \left[\frac{\mathrm{d}\mathbf{u}}{\mathrm{d}\psi}(0) - \frac{\mathrm{d}\boldsymbol{\upsilon}}{\mathrm{d}\psi} \right] + \int_0^T \mathrm{d}t\, \frac{\mathrm{d}j}{\mathrm{d}\psi} + \boldsymbol{\lambda}^\dagger(t) \cdot \left[\frac{\mathrm{d}}{\mathrm{d}\psi} \frac{\mathrm{d}\mathbf{u}}{\mathrm{d}t} - \frac{\mathrm{d}\mathbf{f}}{\mathrm{d}\psi} \right] \\
&= \boldsymbol{\mu}^\dagger \cdot \left[\frac{\mathrm{d}\mathbf{u}}{\mathrm{d}\psi}(0) - \frac{\partial\boldsymbol{\upsilon}}{\partial\psi} \right] + \int_0^T \mathrm{d}t\, \left[\frac{\partial j}{\partial\psi} + \left(\frac{\partial j}{\partial\mathbf{u}} \right)^\dagger \cdot \frac{\mathrm{d}\mathbf{u}}{\mathrm{d}\psi} \right] + \boldsymbol{\lambda}^\dagger(t) \cdot \left[\frac{\mathrm{d}}{\mathrm{d}t} \frac{\mathrm{d}\mathbf{u}}{\mathrm{d}\psi} - \frac{\partial\mathbf{f}}{\partial\psi} - \left(\frac{\partial\mathbf{f}}{\partial\mathbf{u}} \right)^\dagger \cdot \frac{\mathrm{d}\mathbf{u}}{\mathrm{d}\psi} \right].
\end{aligned}$$

Once again a boldfaced fraction is shorthand for a Jacobian matrix. For example,

$$\frac{\partial j}{\partial \mathbf{u}} = \left(\frac{\partial j}{\partial u_1}, \ldots, \frac{\partial j}{\partial u_N} \right)^\dagger.$$

The benefit of this approach is that we can use the freedom in our Lagrange multipliers to eliminate the expensive state sensitivities entirely! First we need to integrate the time derivative of the sensitivities by parts to recover a pure sensitivity,

$$\int_0^T \mathrm{d}t\, \boldsymbol{\lambda}^\dagger(t) \cdot \frac{\mathrm{d}}{\mathrm{d}t} \frac{\mathrm{d}\mathbf{u}}{\mathrm{d}\psi} = \boldsymbol{\lambda}^\dagger(T) \cdot \frac{\mathrm{d}\mathbf{u}}{\mathrm{d}\psi}(T) - \boldsymbol{\lambda}^\dagger(0) \cdot \frac{\mathrm{d}\mathbf{u}}{\mathrm{d}\psi}(0) - \int_0^T \mathrm{d}t\, \left(\frac{\mathrm{d}\boldsymbol{\lambda}}{\mathrm{d}t} \right)^\dagger \cdot \frac{\mathrm{d}\mathbf{u}}{\mathrm{d}\psi}.$$

Then we substitute this result into the total derivative and gather all the sensitivity terms

together,

$$\begin{aligned}\frac{\mathrm{d}\mathcal{J}}{\mathrm{d}\psi} = &\quad \boldsymbol{\mu}^\dagger \cdot \left[\frac{\mathrm{d}\mathbf{u}}{\mathrm{d}\psi}(0) - \frac{\partial \boldsymbol{v}}{\partial \psi}\right] + \boldsymbol{\lambda}^\dagger(T) \cdot \frac{\mathrm{d}\mathbf{u}}{\mathrm{d}\psi}(T) - \boldsymbol{\lambda}^\dagger(0) \cdot \frac{\mathrm{d}\mathbf{u}}{\mathrm{d}\psi}(0) \\ &+ \int_0^T \mathrm{d}t\, \frac{\partial j}{\partial \psi} + \left(\frac{\partial j}{\partial \mathbf{u}}\right)^\dagger \cdot \frac{\mathrm{d}\mathbf{u}}{\mathrm{d}\psi} - \left(\frac{\mathrm{d}\boldsymbol{\lambda}}{\mathrm{d}t}\right)^\dagger \cdot \frac{\mathrm{d}\mathbf{u}}{\mathrm{d}\psi} - \boldsymbol{\lambda}^\dagger(t) \cdot \frac{\partial \mathbf{f}}{\partial \psi} - \boldsymbol{\lambda}^\dagger(t) \cdot \left(\frac{\partial \mathbf{f}}{\partial \mathbf{u}}\right)^\dagger \cdot \frac{\mathrm{d}\mathbf{u}}{\mathrm{d}\psi} \\ = &\quad \left[\boldsymbol{\mu} - \boldsymbol{\lambda}(0)\right]^\dagger \cdot \frac{\mathrm{d}\mathbf{u}}{\mathrm{d}\psi}(0) - \boldsymbol{\mu}^\dagger \cdot \frac{\partial \boldsymbol{v}}{\partial \psi} + \boldsymbol{\lambda}^\dagger(T) \cdot \frac{\mathrm{d}\mathbf{u}}{\mathrm{d}\psi}(T) \\ &+ \int_0^T \mathrm{d}t\, \frac{\partial j}{\partial \psi} - \boldsymbol{\lambda}^\dagger(t) \cdot \frac{\partial \mathbf{f}}{\partial \psi} + \int_0^T \mathrm{d}t\, \left[\frac{\partial j}{\partial \mathbf{u}} - \frac{\mathrm{d}\boldsymbol{\lambda}}{\mathrm{d}t} - \boldsymbol{\lambda}(t) \cdot \frac{\partial \mathbf{f}}{\partial \mathbf{u}}\right]^\dagger \cdot \frac{\mathrm{d}\mathbf{u}}{\mathrm{d}\psi}\end{aligned}$$

Now we can exploit the freedom in our Lagrange multipliers to remove all vestiges of the sensitivities. First let's set $\boldsymbol{\mu} = \boldsymbol{\lambda}(0)$ to remove the initial sensitivities and $\boldsymbol{\lambda}(T) = 0$ to remove the final sensitivities. We can then remove the integral term that depends on the intermediate sensitivities if we set

$$\frac{\partial j}{\partial \mathbf{u}} - \frac{\mathrm{d}\boldsymbol{\lambda}}{\mathrm{d}t} - \boldsymbol{\lambda}(t) \cdot \frac{\partial \mathbf{f}}{\partial \mathbf{u}} = 0,$$

or

$$\frac{\mathrm{d}\boldsymbol{\lambda}}{\mathrm{d}t} = \frac{\partial j}{\partial \mathbf{u}} - \boldsymbol{\lambda}(t) \cdot \frac{\partial \mathbf{f}}{\partial \mathbf{u}}.$$

In other words provided that $\boldsymbol{\lambda}(t)$ satisfies the differential equation

$$\frac{\mathrm{d}\boldsymbol{\lambda}}{\mathrm{d}t} = \frac{\partial j}{\partial \mathbf{u}}(\mathbf{u}, \psi, t) - \boldsymbol{\lambda}(t) \cdot \frac{\partial \mathbf{f}}{\partial \mathbf{u}}(\mathbf{u}, \psi, t)$$

with the initial conditions

$$\boldsymbol{\lambda}(T) = 0$$

then then total derivative of our target functional reduces to

$$\frac{\mathrm{d}\mathcal{J}}{\mathrm{d}\psi}(\psi) = -\boldsymbol{\lambda}^\dagger(0) \cdot \frac{\partial \boldsymbol{v}}{\partial \psi} + \int_0^T \mathrm{d}t\, \frac{\partial j}{\partial \psi}(\mathbf{u}, \psi, t) - \boldsymbol{\lambda}^\dagger(t) \cdot \frac{\partial \mathbf{f}}{\partial \psi}(\mathbf{u}, \psi, t).$$

The system of differential equations for $\boldsymbol{\lambda}(t)$ is known as the *adjoint* system relative to the original system of ordinary differential equations. If we first solve for $\mathbf{u}(t)$ then we can solve for the adjoint $\boldsymbol{\lambda}(t)$ and compute the total derivative $\mathrm{d}\mathcal{J}/\mathrm{d}\psi$ at the same time without having to compute any explicit sensitivities.

1.2 Computational Scalings

For a single parameter the direct approach is slightly more efficient, requiring two N-dimensional integrations for the states and their sensitivities compared to the adjoint approach which requires two N-dimensional integrations, one for the states and one for the

adjoint states, *and* the extra one-dimensional integration to solve for the total derivative. The adjoint method, however, quickly becomes more efficient as we consider multiple parameters because the adjoint states *are the same for all parameters*.

When we have K parameters the forward sensitivity approach requires an N-dimensional integration for *each* sensitivity and the total cost scales as $N + N \cdot K$. The adjoint approach, however, requires only two N-dimensional solves to set up the states and the adjoint states and then K one-dimensional solves for each gradient component, yielding a total cost scaling of $2N + K$.

Comparing these two scalings we see that the adjoint method is better when

$$\frac{N}{N - 1} < K,$$

a condition verified for any N provided that $K \geq 2$. In other words the adjoint method will generally feature the highest performance in any application with at least two parameters. As the number of parameters increases the $\mathcal{O}(NK)$ scaling of the forward sensitivity approach grows much faster than the $\mathcal{O}(N + K)$ scaling of the adjoint method, and the performance gap only becomes more substantial.

1.3 An Application to Automatic Differentiation

A particularly useful application of the continuous adjoint method is for the reverse mode automatic differentiation (Bücker et al., 2006; Griewank and Walther, 2008; Margossian, 2019) of functions incorporating the solutions of ordinary differential equations. In order to propagate the needed differential information through the composite function we need to be able to evaluate the Jacobian of the final state with respect to the parameters,

$$\frac{\mathrm{d}\mathbf{u}}{\mathrm{d}\psi}(T),$$

contracted against a vector, δ,

$$\boldsymbol{\delta}^{\dagger} \cdot \frac{\mathrm{d}\mathbf{u}}{\mathrm{d}\psi}(T),$$

where $\dagger$ denotes transposition. This arises, for example, when computing the gradient of a scalar function, for example a probability density or an objective function, which implicitly depends on ψ through $\mathbf{u}$.

We can recover the above contraction by defining the integrand

$$j(\mathbf{u}, \psi, t) = \boldsymbol{\delta}^{\dagger} \cdot \mathbf{f}(\mathbf{u}, \psi, t)$$

and the corresponding functional

$$\begin{aligned}\mathcal{J}(\psi) &= \int_0^T \mathrm{d}t\, j(\mathbf{u}, \psi, t) \\ &= \boldsymbol{\delta}^\dagger \cdot \int_0^T \mathrm{d}t\, \mathbf{f}(\mathbf{u}, \psi, t) \\ &= \boldsymbol{\delta}^\dagger \cdot \int_0^T \mathrm{d}t\, \frac{\mathrm{d}\mathbf{u}}{\mathrm{d}t}(\mathbf{u}, \psi, t) \\ &= \boldsymbol{\delta}^\dagger \cdot (\mathbf{u}(T) - \mathbf{u}(0))\,.\end{aligned}$$

The total derivative of this functional is given by

$$\begin{aligned}\frac{\mathrm{d}\mathcal{J}}{\mathrm{d}\psi}(\psi) &= \boldsymbol{\delta}^\dagger \cdot \left(\frac{\mathrm{d}\mathbf{u}}{\mathrm{d}\psi}(T) - \frac{\mathrm{d}\mathbf{u}}{\mathrm{d}\psi}(0)\right) \\ &= \boldsymbol{\delta}^\dagger \cdot \left(\frac{\mathrm{d}\mathbf{u}}{\mathrm{d}\psi}(T) - \frac{\partial \boldsymbol{\upsilon}}{\partial \psi}\right)\end{aligned}$$

which we can then manipulate into the desired contraction

$$\boldsymbol{\delta}^\dagger \cdot \frac{\mathrm{d}\mathbf{u}}{\mathrm{d}\psi}(T) = \frac{\mathrm{d}\mathcal{J}}{\mathrm{d}\psi}(\psi) + \boldsymbol{\delta}^\dagger \cdot \frac{\partial \boldsymbol{\upsilon}}{\partial \psi}.$$

We can then use the continuous adjoint method to evaluate the total derivative of the functional and hence the desired Jacobian-adjoint product,

$$\begin{aligned}\boldsymbol{\delta}^\dagger \cdot \frac{\mathrm{d}\mathbf{u}}{\mathrm{d}\psi}(T) &= \boldsymbol{\delta}^\dagger \cdot \frac{\partial \boldsymbol{\upsilon}}{\partial \psi} + \frac{\mathrm{d}\mathcal{J}}{\mathrm{d}\psi}(\psi) \\ &= \boldsymbol{\delta}^\dagger \cdot \frac{\partial \boldsymbol{\upsilon}}{\partial \psi} - \boldsymbol{\lambda}^\dagger(0) \cdot \frac{\partial \boldsymbol{\upsilon}}{\partial \psi} + \int_0^T \mathrm{d}t\, \boldsymbol{\delta}^\dagger \cdot \frac{\partial \mathbf{f}}{\partial \psi}(\mathbf{u}, \psi, t) - \boldsymbol{\lambda}^\dagger(t) \cdot \frac{\partial \mathbf{f}}{\partial \psi}(\mathbf{u}, \psi, t) \\ &= \left[\boldsymbol{\delta} - \boldsymbol{\lambda}(0)\right]^\dagger \cdot \frac{\partial \boldsymbol{\upsilon}}{\partial \psi} + \int_0^T \mathrm{d}t \left[\boldsymbol{\delta} - \boldsymbol{\lambda}(t)\right]^\dagger \cdot \frac{\partial \mathbf{f}}{\partial \psi}(\mathbf{u}, \psi, t).\end{aligned}$$

2. DISCRETE ADJOINT SYSTEMS

By carefully translating the differential operations in the continuous adjoint method to their discrete counterparts we can derive a corresponding discrete adjoint method.

Recall that in the discrete case our target functional is defined as

$$\mathcal{J}(\psi) = \sum_{n=0}^{N-1} j_n(\mathbf{u}_n, \psi, n)$$

with the discrete states satisfying the forward difference equation,

$$\mathbf{u}_{n+1} - \mathbf{u}_n = \boldsymbol{\Delta}_n(\mathbf{u}_n, \psi, n),$$

along with the initial condition

$$\mathbf{u}_0(\psi) = \boldsymbol{\upsilon}(\psi).$$

To construct the adjoint system we first introduce the nominal system as explicit constraints in a modified functional,

$$\mathcal{J}(\psi) = \mathcal{L}(\psi) = \boldsymbol{\mu}^T \cdot [\boldsymbol{\upsilon} - \mathbf{u_0}] + \sum_{n=0}^{N-1} j_n + \boldsymbol{\lambda}_n^T \cdot [\mathbf{u}_{n+1} - \mathbf{u}_n - \boldsymbol{\Delta}_n].$$

Taking a total derivative then gives

$$\begin{aligned}
\frac{\mathrm{d}\mathcal{J}}{\mathrm{d}\psi} &= \frac{\mathrm{d}\mathcal{L}}{\mathrm{d}\psi} \\
&= \boldsymbol{\mu}^\dagger \cdot \left[\frac{\mathrm{d}\mathbf{u}_0}{\mathrm{d}\psi} - \frac{\mathrm{d}\boldsymbol{\upsilon}}{\mathrm{d}\psi}\right] + \sum_{n=0}^{N-1} \frac{\mathrm{d}j_n}{\mathrm{d}\psi} + \boldsymbol{\lambda}_n^\dagger \cdot \left[\frac{\mathrm{d}\mathbf{u}_{n+1}}{\mathrm{d}\psi} - \frac{\mathrm{d}\mathbf{u}_n}{\mathrm{d}\psi} - \frac{\mathrm{d}\boldsymbol{\Delta}_n}{\mathrm{d}\psi}\right] \\
&= \boldsymbol{\mu}^\dagger \cdot \left[\frac{\mathrm{d}\mathbf{u}_0}{\mathrm{d}\psi} - \frac{\mathrm{d}\boldsymbol{\upsilon}}{\mathrm{d}\psi}\right] \\
&\quad + \sum_{n=0}^{N-1} \frac{\partial j_n}{\partial \psi} + \left(\frac{\partial j_n}{\partial \mathbf{u}_n}\right)^\dagger \cdot \frac{\mathrm{d}\mathbf{u}_n}{\mathrm{d}\psi} + \boldsymbol{\lambda}_n^\dagger \cdot \left[\frac{\mathrm{d}\mathbf{u}_{n+1}}{\mathrm{d}\psi} - \frac{\mathrm{d}\mathbf{u}_n}{\mathrm{d}\psi} - \frac{\partial \boldsymbol{\Delta}_n}{\partial \psi} - \left(\frac{\partial \boldsymbol{\Delta}_n}{\partial \mathbf{u}_n}\right)^\dagger \cdot \frac{\mathrm{d}\mathbf{u}_n}{\mathrm{d}\psi}\right] \\
&= \boldsymbol{\mu}^\dagger \cdot \left[\frac{\mathrm{d}\mathbf{u}_0}{\mathrm{d}\psi} - \frac{\mathrm{d}\boldsymbol{\upsilon}}{\mathrm{d}\psi}\right] \\
&\quad + \sum_{n=0}^{N-1} \frac{\partial j_n}{\partial \psi} - \boldsymbol{\lambda}_n^\dagger \cdot \frac{\partial \boldsymbol{\Delta}_n}{\partial \psi} + \sum_{n=0}^{N-1} \boldsymbol{\lambda}_n^\dagger \cdot \left[\frac{\mathrm{d}\mathbf{u}_{n+1}}{\mathrm{d}\psi} - \frac{\mathrm{d}\mathbf{u}_n}{\mathrm{d}\psi}\right] + \sum_{n=0}^{N-1} \left(\frac{\partial j_n}{\partial \mathbf{u}_n}\right)^\dagger \cdot \frac{\mathrm{d}\mathbf{u}_n}{\mathrm{d}\psi} - \left(\boldsymbol{\lambda}_n \cdot \frac{\partial \boldsymbol{\Delta}_n}{\partial \mathbf{u}_n}\right)^\dagger \cdot \frac{\mathrm{d}\mathbf{u}_n}{\mathrm{d}\psi} \\
&= \boldsymbol{\mu}^\dagger \cdot \left[\frac{\mathrm{d}\mathbf{u}_0}{\mathrm{d}\psi} - \frac{\mathrm{d}\boldsymbol{\upsilon}}{\mathrm{d}\psi}\right] + \left(\frac{\partial j_0}{\partial \mathbf{u}_0}\right)^\dagger \cdot \frac{\mathrm{d}\mathbf{u}_0}{\mathrm{d}\psi} - \left(\boldsymbol{\lambda}_0 \cdot \frac{\partial \boldsymbol{\Delta}_0}{\partial \mathbf{u}_0}\right)^\dagger \cdot \frac{\mathrm{d}\mathbf{u}_0}{\mathrm{d}\psi} \\
&\quad + \sum_{n=0}^{N-1} \frac{\partial j_n}{\partial \psi} - \boldsymbol{\lambda}_n^\dagger \cdot \frac{\partial \boldsymbol{\Delta}_n}{\partial \psi} + \sum_{n=0}^{N-1} \boldsymbol{\lambda}_n^\dagger \cdot \left[\frac{\mathrm{d}\mathbf{u}_{n+1}}{\mathrm{d}\psi} - \frac{\mathrm{d}\mathbf{u}_n}{\mathrm{d}\psi}\right] + \sum_{n=1}^{N-1} \left(\frac{\partial j_n}{\partial \mathbf{u}_n}\right)^\dagger \cdot \frac{\mathrm{d}\mathbf{u}_n}{\mathrm{d}\psi} - \left(\boldsymbol{\lambda}_n \cdot \frac{\partial \boldsymbol{\Delta}_n}{\partial \mathbf{u}_n}\right)^\dagger \cdot \frac{\mathrm{d}\mathbf{u}_n}{\mathrm{d}\psi}.
\end{aligned}$$

Now we can apply *summation by parts* to the forwards difference of sensitivities,

$$\sum_{n=0}^{N-1} \boldsymbol{\lambda}_n^\dagger \cdot \left[\frac{\mathrm{d}\mathbf{u}_{n+1}}{\mathrm{d}\psi} - \frac{\mathrm{d}\mathbf{u}_n}{\mathrm{d}\psi}\right] = \boldsymbol{\lambda}_{N-1}^\dagger \cdot \frac{\mathrm{d}\mathbf{u}_N}{\mathrm{d}\psi} - \boldsymbol{\lambda}_0^\dagger \cdot \frac{\mathrm{d}\mathbf{u}_0}{\mathrm{d}\psi} - \sum_{n=1}^{N-1} \left[\boldsymbol{\lambda}_n - \boldsymbol{\lambda}_{n-1}\right]^\dagger \cdot \frac{\mathrm{d}\mathbf{u}_n}{\mathrm{d}\psi}.$$

Plugging this result into our functional derivative then gives

$$\begin{aligned}
\frac{\mathrm{d}\mathcal{J}}{\mathrm{d}\psi} = \quad & \boldsymbol{\mu}^\dagger \cdot \left[\frac{\mathrm{d}\mathbf{u}_0}{\mathrm{d}\psi} - \frac{\mathrm{d}\boldsymbol{\upsilon}}{\mathrm{d}\psi}\right] + \left(\frac{\partial j_0}{\partial \mathbf{u}_0}\right)^\dagger \cdot \frac{\mathrm{d}\mathbf{u}_0}{\mathrm{d}\psi} - \left(\boldsymbol{\lambda}_0 \cdot \frac{\partial \boldsymbol{\Delta}_0}{\partial \mathbf{u}_0}\right)^\dagger \cdot \frac{\mathrm{d}\mathbf{u}_0}{\mathrm{d}\psi} + \boldsymbol{\lambda}_{N-1}^\dagger \cdot \frac{\mathrm{d}\mathbf{u}_N}{\mathrm{d}\psi} - \boldsymbol{\lambda}_0^\dagger \cdot \frac{\mathrm{d}\mathbf{u}_0}{\mathrm{d}\psi} \\
& + \sum_{n=0}^{N-1} \frac{\partial j_n}{\partial \psi} - \boldsymbol{\lambda}_n^\dagger \cdot \frac{\partial \boldsymbol{\Delta}_n}{\partial \psi} \\
& + \sum_{n=1}^{N-1} \left(\frac{\partial j_n}{\partial \mathbf{u}_n}\right)^\dagger \cdot \frac{\mathrm{d}\mathbf{u}_n}{\mathrm{d}\psi} - \left(\boldsymbol{\lambda}_n \cdot \frac{\partial \boldsymbol{\Delta}_n}{\partial \mathbf{u}_n}\right)^\dagger \cdot \frac{\mathrm{d}\mathbf{u}_n}{\mathrm{d}\psi} - \sum_{n=1}^{N-1} \left[\boldsymbol{\lambda}_n - \boldsymbol{\lambda}_{n-1}\right]^\dagger \cdot \frac{\mathrm{d}\mathbf{u}_n}{\mathrm{d}\psi} \\
= \quad & \left[\boldsymbol{\mu} + \frac{\partial j_0}{\partial \boldsymbol{\upsilon}} - \boldsymbol{\lambda}_0 \cdot \frac{\partial \boldsymbol{\Delta}_0}{\partial \boldsymbol{\upsilon}} - \boldsymbol{\lambda}_0\right]^\dagger \cdot \frac{\mathrm{d}\mathbf{u}_0}{\mathrm{d}\psi} - \boldsymbol{\mu}^\dagger \cdot \frac{\partial \boldsymbol{\upsilon}}{\partial \psi} + \boldsymbol{\lambda}_{N-1}^\dagger \cdot \frac{\mathrm{d}\mathbf{u}_N}{\mathrm{d}\psi} \\
& + \sum_{n=0}^{N-1} \frac{\partial j_n}{\partial \psi} - \boldsymbol{\lambda}_n^\dagger \cdot \frac{\partial \boldsymbol{\Delta}_n}{\partial \psi} \\
& + \sum_{n=1}^{N-1} \left[\frac{\partial j_n}{\partial \mathbf{u}_n} - \boldsymbol{\lambda}_n + \boldsymbol{\lambda}_{n-1} - \boldsymbol{\lambda}_n \cdot \frac{\partial \boldsymbol{\Delta}_n}{\partial \mathbf{u}_n}\right]^\dagger \cdot \frac{\mathrm{d}\mathbf{u}_n}{\mathrm{d}\psi}.
\end{aligned}$$

As in the discrete case we can exploit the freedom in our Lagrange multipliers to remove all of the sensitivity terms. We first set

$$\boldsymbol{\mu} + \frac{\partial j_0}{\partial \mathbf{u}_0} - \boldsymbol{\lambda}_0 \cdot \frac{\partial \boldsymbol{\Delta}_0}{\partial \mathbf{u}_0} - \boldsymbol{\lambda}_0 = 0,$$

or

$$\boldsymbol{\mu} = -\frac{\partial j_0}{\partial \mathbf{u}_0} + \boldsymbol{\lambda}_0 \cdot \frac{\partial \boldsymbol{\Delta}_0}{\partial \mathbf{u}_0} + \boldsymbol{\lambda}_0,$$

and then

$$\boldsymbol{\lambda}_{N-1} = 0$$

to remove all the sensitivities outside of the summations. We then eliminate the second summation by choosing the rest of the $\boldsymbol{\lambda}_n$ to satisfy

$$\frac{\partial j_n}{\partial \mathbf{u}_n} - \boldsymbol{\lambda}_n + \boldsymbol{\lambda}_{n-1} - \boldsymbol{\lambda}_n \cdot \frac{\partial \boldsymbol{\Delta}_n}{\partial \mathbf{u}_n} = 0,$$

or equivalently

$$\frac{\partial j_{n+1}}{\partial \mathbf{u}_{n+1}} - \boldsymbol{\lambda}_{n+1} + \boldsymbol{\lambda}_n - \boldsymbol{\lambda}_{n+1} \cdot \frac{\partial \boldsymbol{\Delta}_{n+1}}{\partial \mathbf{u}_{n+1}} = 0.$$

This defines an adjoint system defined by the *backward* difference equations

$$\boldsymbol{\lambda}_n - \boldsymbol{\lambda}_{n+1} = -\frac{\partial j_{n+1}}{\partial \mathbf{u}_{n+1}} + \boldsymbol{\lambda}_{n+1} \cdot \frac{\partial \boldsymbol{\Delta}_{n+1}}{\partial \mathbf{u}_{n+1}}$$

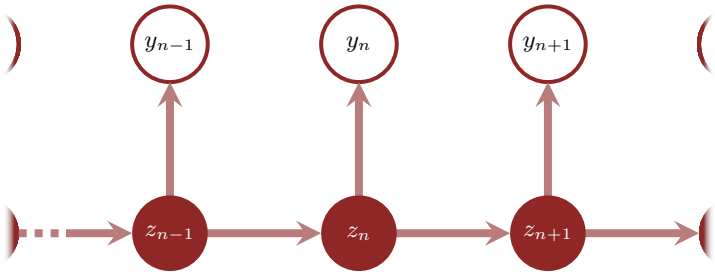

FIG 1. *The conditional dependence structure of a hidden Markov model admits efficient marginalization of the discrete hidden states into state probabilities. Derivatives of the state probabilities with respect to the model parameters also have to navigate this conditional dependence structure.*

along with the terminal condition

$$\boldsymbol{\lambda}_{N-1} = 0.$$

If we solve for the sequence $\boldsymbol{\lambda}_{(N-1):0}$ after first forward solving the original sequence $\mathbf{u}_{0:N}$, we can compute the total derivative of the functional as

$$\frac{\mathrm{d}\mathcal{J}}{\mathrm{d}\psi} = \left[\frac{\partial j_0}{\partial \mathbf{u}_0} - \boldsymbol{\lambda}_0 \cdot \frac{\partial \boldsymbol{\Delta}_0}{\partial \mathbf{u}_0} - \boldsymbol{\lambda}_0\right]^\dagger \cdot \frac{\partial \upsilon}{\partial \psi} + \sum_{n=0}^{N-1} \frac{\partial j_n}{\partial \psi} - \boldsymbol{\lambda}_n^\dagger \cdot \frac{\partial \boldsymbol{\Delta}_n}{\partial \psi}.$$

3. APPLICATION TO HIDDEN MARKOV MODELS

The discrete adjoint method is applicable to any discrete sequence defined by forward difference equations that depend only on the current state. In this section we demonstrate an application of the method to common hidden Markov models.

An elementary hidden Markov model is a probabilistic model over N observations, y_n, and N hidden states, z_n, satisfying the conditional dependence structure shown in Figure 1. The joint density $\pi(y_{1:N}, z_{1:N}, \psi)$ is readily computed, but the derivatives are ill-defined when the hidden states z are discrete. In order to apply gradient-based methods we first need to marginalize out the hidden states to define the *marginal likelihood* $\pi(y_{1:N}, \psi)$ which can be differentiated.

Fortunately exact marginalization is tractable due to the conditional dependencies inherent to a hidden Markov model. Defining the observational density functions

$$\omega_{n,i} \equiv \pi(y_n \mid z_n = i)$$

and the transition matrices

$$\Gamma_{n,ij} \equiv \pi(z_{n+1} = i \mid z_n = j)$$

we can marginalize the hidden states into the forward state probabilities

$$\alpha_{n,i} \equiv \pi(y_{1:N}, z_n = i).$$

Because of the defining conditional structure these state probabilities satisfy the recursion relation

$$\boldsymbol{\alpha}_{n+1}(\psi) = \boldsymbol{\omega}_{n+1}(\psi) \circ (\boldsymbol{\Gamma}_{n+1}(\psi) \cdot \boldsymbol{\alpha}_n(\psi)),$$

where $\circ$ denotes the element-wise Hadamard product, along with the initial condition

$$\boldsymbol{\upsilon}(\psi) = \boldsymbol{\alpha}_0(\psi) = \boldsymbol{\omega}_0(\psi) \circ \boldsymbol{\rho}(\psi).$$

Forward solving the recursion relation efficiently computes each of the state probabilities, the last of which gives the desired marginal likelihood

$$\pi(y_1, \ldots, y_N, \psi) = \sum_{m=1}^{M} \alpha_{N,m}(\psi) = \mathbf{1}^\dagger \cdot \boldsymbol{\alpha}_N(\psi).$$

In order to apply gradient-based learning algorithms to any probabilistic model containing a hidden Markov model we have to compute not only the marginal likelihood but also its gradient with respect to any unknown parameters.

There are many ways to derive the gradient for this problem; in this section we will consider three approaches that tackle the derivation from different directions and different intuitions but arrive at the same result. These different approaches not only serve as cross checks for each other but also suggest that their common result is optimal.

In the statistics literature the gradient of the marginal likelihood is often derived as an indirect and subtle byproduct of the expectation maximization algorithm (Cappé, Moulines and Rydén, 2005).

We can also obtain a more explicit derivation by unrolling the recursion and applying the chain rule iteratively. If we let $\boldsymbol{\Omega}_n$ denote a diagonal matrix of observational densities at the nth iteration,

$$\boldsymbol{\Omega}_n = \operatorname{diag}(\boldsymbol{\omega}_n),$$

then the final state probabilities can be written explicitly as

$$\boldsymbol{\alpha}_N = \left[\prod_{n=1}^{N} \boldsymbol{\Omega}_n(\psi) \cdot \boldsymbol{\Gamma}_{n+1}(\psi) \right] \cdot \boldsymbol{\Omega}_0(\psi) \cdot \boldsymbol{\rho}(\psi)$$

with the marginal likelihood taking the form

$$\pi(y_1, \ldots, y_N, \psi) = \mathbf{1}^\dagger \cdot \boldsymbol{\alpha}_N = \mathbf{1}^\dagger \cdot \left[\prod_{n=1}^{N} \boldsymbol{\Omega}_n \cdot \boldsymbol{\Gamma}_{n+1} \right] \cdot \boldsymbol{\Omega}_0 \cdot \boldsymbol{\rho}.$$

Applying the product rule for derivatives then gives

$$
\begin{aligned}
\frac{\mathrm{d}}{\mathrm{d}\psi}\pi(y_1,\ldots,y_N) &= \frac{\mathrm{d}}{\mathrm{d}\psi}\left(\mathbf{1}^{\dagger}\cdot\left[\prod_{n=1}^{N-1}\boldsymbol{\Omega}_n\cdot\boldsymbol{\Gamma}_{n+1}\right]\cdot\boldsymbol{\Omega}_0\cdot\boldsymbol{\rho}\right)\\
&= \mathbf{1}^{\dagger}\cdot\sum_{j=0}^{N-1}\left[\prod_{i=j+2}^{N-1}\boldsymbol{\Omega}_i\cdot\boldsymbol{\Gamma}_{i+1}\right]\cdot\left[\frac{\mathrm{d}\boldsymbol{\Omega}_{j+1}}{\mathrm{d}\psi}\cdot\boldsymbol{\Gamma}_{j+2}+\boldsymbol{\Omega}_{j+1}\cdot\frac{\mathrm{d}\boldsymbol{\Gamma}_{j+2}}{\mathrm{d}\psi}\right]\cdot\left[\prod_{k=1}^{j}\boldsymbol{\Omega}_k\cdot\boldsymbol{\Gamma}_{k+1}\right]\cdot\boldsymbol{\Omega}_0\cdot\boldsymbol{\rho}\\
&\quad+\mathbf{1}^{\dagger}\cdot\left[\prod_{n=1}^{N-1}\boldsymbol{\Omega}_n\cdot\boldsymbol{\Gamma}_{n+1}\right]\cdot\left[\frac{\mathrm{d}\boldsymbol{\Omega}_0}{\mathrm{d}\psi}\cdot\boldsymbol{\rho}+\boldsymbol{\Omega}_0\cdot\frac{\mathrm{d}\boldsymbol{\rho}}{\mathrm{d}\psi}\right]\\
&= \sum_{j=0}^{N-1}\left[\mathbf{1}^{\dagger}\cdot\prod_{i=j+2}^{N-1}\boldsymbol{\Omega}_i\cdot\boldsymbol{\Gamma}_{i+1}\right]\cdot\left[\frac{\mathrm{d}\boldsymbol{\Omega}_{j+1}}{\mathrm{d}\psi}\cdot\boldsymbol{\Gamma}_{j+2}+\boldsymbol{\Omega}_{j+1}\cdot\frac{\mathrm{d}\boldsymbol{\Gamma}_{j+2}}{\mathrm{d}\psi}\right]\cdot\boldsymbol{\alpha}_j\\
&\quad+\mathbf{1}^{\dagger}\cdot\left[\prod_{n=1}^{N-1}\boldsymbol{\Omega}_n\cdot\boldsymbol{\Gamma}_{n+1}\right]\cdot\left[\frac{\mathrm{d}\boldsymbol{\Omega}_0}{\mathrm{d}\psi}\cdot\boldsymbol{\rho}+\boldsymbol{\Omega}_0\cdot\frac{\mathrm{d}\boldsymbol{\rho}}{\mathrm{d}\psi}\right]\\
&= \sum_{j=0}^{N-1}\left[\left[\mathbf{1}^{\dagger}\cdot\prod_{i=j+2}^{N-1}\boldsymbol{\Omega}_i\cdot\boldsymbol{\Gamma}_{i+1}\right]^{\dagger}\right]^{\dagger}\cdot\left[\frac{\mathrm{d}\boldsymbol{\Omega}_{j+1}}{\mathrm{d}\psi}\cdot\boldsymbol{\Gamma}_{j+2}+\boldsymbol{\Omega}_{j+1}\cdot\frac{\mathrm{d}\boldsymbol{\Gamma}_{j+2}}{\mathrm{d}\psi}\right]\cdot\boldsymbol{\alpha}_j\\
&\quad+\left[\left[\mathbf{1}^{\dagger}\cdot\prod_{i=1}^{N-1}\boldsymbol{\Omega}_i\cdot\boldsymbol{\Gamma}_{i+1}\right]^{\dagger}\right]^{\dagger}\cdot\left[\frac{\mathrm{d}\boldsymbol{\Omega}_0}{\mathrm{d}\psi}\cdot\boldsymbol{\rho}+\boldsymbol{\Omega}_0\cdot\frac{\mathrm{d}\boldsymbol{\rho}}{\mathrm{d}\psi}\right]\\
&= \sum_{j=0}^{N-1}\left[\left[\prod_{i=N-1}^{j+2}\boldsymbol{\Gamma}_{i+1}^{\dagger}\cdot\boldsymbol{\Omega}_i^{\dagger}\right]\cdot\mathbf{1}\right]^{\dagger}\cdot\left[\frac{\mathrm{d}\boldsymbol{\Omega}_{j+1}}{\mathrm{d}\psi}\cdot\boldsymbol{\Gamma}_{j+2}+\boldsymbol{\Omega}_{j+1}\cdot\frac{\mathrm{d}\boldsymbol{\Gamma}_{j+2}}{\mathrm{d}\psi}\right]\cdot\boldsymbol{\alpha}_j\\
&\quad+\left[\left[\prod_{i=N-1}^{1}\boldsymbol{\Gamma}_{i+1}^{\dagger}\cdot\boldsymbol{\Omega}_i^{\dagger}\right]\cdot\mathbf{1}\right]^{\dagger}\cdot\left[\frac{\mathrm{d}\boldsymbol{\Omega}_0}{\mathrm{d}\psi}\cdot\boldsymbol{\rho}+\boldsymbol{\Omega}_0\cdot\frac{\mathrm{d}\boldsymbol{\rho}}{\mathrm{d}\psi}\right]\\
&= \sum_{j=0}^{N-1}\left[\left[\prod_{i=N-1}^{j+2}\boldsymbol{\Gamma}_{i+1}^{\dagger}\cdot\boldsymbol{\Omega}_i\right]\cdot\mathbf{1}\right]^{\dagger}\cdot\left[\frac{\mathrm{d}\boldsymbol{\Omega}_{j+1}}{\mathrm{d}\psi}\cdot\boldsymbol{\Gamma}_{j+2}+\boldsymbol{\Omega}_{j+1}\cdot\frac{\mathrm{d}\boldsymbol{\Gamma}_{j+2}}{\mathrm{d}\psi}\right]\cdot\boldsymbol{\alpha}_j\\
&\quad+\left[\left[\prod_{i=N-1}^{1}\boldsymbol{\Gamma}_{i+1}^{\dagger}\cdot\boldsymbol{\Omega}_i\right]\cdot\mathbf{1}\right]^{\dagger}\cdot\left[\frac{\mathrm{d}\boldsymbol{\Omega}_0}{\mathrm{d}\psi}\cdot\boldsymbol{\rho}+\boldsymbol{\Omega}_0\cdot\frac{\mathrm{d}\boldsymbol{\rho}}{\mathrm{d}\psi}\right]\\
&= \sum_{j=0}^{N-1}\left[\boldsymbol{\beta}_{j+1}\right]^{\dagger}\cdot\left[\frac{\mathrm{d}\boldsymbol{\Omega}_{j+1}}{\mathrm{d}\psi}\cdot\boldsymbol{\Gamma}_{j+2}+\boldsymbol{\Omega}_{j+1}\cdot\frac{\mathrm{d}\boldsymbol{\Gamma}_{j+2}}{\mathrm{d}\psi}\right]\cdot\boldsymbol{\alpha}_j\\
&\quad+\left[\boldsymbol{\beta}_0\right]^{\dagger}\cdot\left[\frac{\mathrm{d}\boldsymbol{\Omega}_0}{\mathrm{d}\psi}\cdot\boldsymbol{\rho}+\boldsymbol{\Omega}_0\cdot\frac{\mathrm{d}\boldsymbol{\rho}}{\mathrm{d}\psi}\right],
\end{aligned}
$$

where we have defined the *backwards states*

$$\boldsymbol{\beta}_j = \left[\prod_{i=N-1}^{j+1} \boldsymbol{\Gamma}_{i+1}^{\dagger} \cdot \boldsymbol{\Omega}_i \right] \cdot \mathbf{1}$$

A third, novel approach to deriving the marginal likelihood gradient is to interpret the recursion as a forward difference equation and apply the discrete adjoint method. Let $\mathbf{u}_n = \boldsymbol{\alpha}_n$ and manipulate the defining recursion relation into a forward difference

$$\boldsymbol{\Delta}_n = \boldsymbol{\omega}_{n+1} \circ (\boldsymbol{\Gamma}_{n+1} \cdot \boldsymbol{\alpha}_n) - \boldsymbol{\alpha}_n,$$

and take the summand

$$j = \mathbf{1}^{\dagger} \cdot \boldsymbol{\Delta}_n$$

to give the discrete functional

$$J = \mathbf{1}^{\dagger} \cdot (\boldsymbol{\alpha}_N - \boldsymbol{\upsilon}).$$

The total derivative of the discrete functional can be used to derive the derivative of the marginal likelihood,

$$\begin{aligned} \frac{\mathrm{d}\mathcal{J}}{\mathrm{d}\psi} &= \frac{\mathrm{d}}{\mathrm{d}\psi}\left(\mathbf{1}^{\dagger} \cdot \boldsymbol{\alpha}_N\right) - \frac{\mathrm{d}}{\mathrm{d}\psi}\left(\mathbf{1}^{\dagger} \cdot \boldsymbol{\alpha}_0\right) \\ &= \frac{\mathrm{d}}{\mathrm{d}\psi}\pi(y_1, \ldots, y_N) - \mathbf{1}^{\dagger} \cdot \frac{\mathrm{d}\boldsymbol{\alpha}_0}{\mathrm{d}\psi}, \end{aligned}$$

or

$$\frac{\mathrm{d}}{\mathrm{d}\psi}\pi(y_1, \ldots, y_N) = \frac{\mathrm{d}\mathcal{J}}{\mathrm{d}\psi} + \mathbf{1}^{\dagger} \cdot \frac{\mathrm{d}\boldsymbol{\alpha}_0}{\mathrm{d}\psi}.$$

In this case the adjoint system is defined as

$$\begin{aligned} \boldsymbol{\lambda}_n - \boldsymbol{\lambda}_{n+1} &= -\frac{\partial j_{n+1}}{\partial \boldsymbol{\alpha}_{n+1}} + \boldsymbol{\lambda}_{n+1} \cdot \frac{\partial \boldsymbol{\Delta}_{n+1}}{\partial \boldsymbol{\alpha}_{n+1}} \\ &= -\mathbf{1} \cdot \frac{\partial \boldsymbol{\Delta}_{n+1}}{\partial \boldsymbol{\alpha}_{n+1}} + \boldsymbol{\lambda}_{n+1} \cdot \frac{\partial \boldsymbol{\Delta}_{n+1}}{\partial \boldsymbol{\alpha}_{n+1}} \\ &= (\boldsymbol{\lambda}_{n+1} - \mathbf{1}) \cdot \frac{\partial \boldsymbol{\Delta}_{n+1}}{\partial \boldsymbol{\alpha}_{n+1}}. \end{aligned}$$

The partial derivative reduces to

$$\begin{aligned} \frac{\partial \Delta_{n,i}}{\partial \alpha_{n,j}} &= \frac{\partial}{\partial \alpha_{n,j}} \left(\omega_{n+1,i} \sum_{k=1}^{K} \Gamma_{n+1,ik}\, \alpha_{n,k} - \alpha_{n,i} \right) \\ &= \omega_{n+1,i} \sum_{k=1}^{K} \Gamma_{n+1,ik}\, \delta_{jk} - \delta_{ij} \\ &= \omega_{n+1,i}\, \Gamma_{n+1,ij} - \delta_{ij} \end{aligned}$$

so that

$$\sum_{i=1}^{K} (\lambda_{n,i} - 1) \frac{\partial \Delta_{n,i}}{\partial \alpha_{n,j}} = \sum_{i=1}^{K} (\lambda_{n,i} - 1) \, \omega_{n+1,i} \, \Gamma_{n+1,ij} - (\lambda_{n,j} - 1),$$

or in matrix notation,

$$(\boldsymbol{\lambda}_n - \mathbf{1}) \cdot \frac{\partial \boldsymbol{\Delta}_n}{\partial \boldsymbol{\alpha}_n} = \boldsymbol{\Gamma}_{n+1}^{\dagger} \cdot (\boldsymbol{\omega}_{n+1} \circ (\boldsymbol{\lambda}_n - \mathbf{1})) - \boldsymbol{\lambda}_n + \mathbf{1}.$$

The backwards updates then become

$$\begin{aligned} \boldsymbol{\lambda}_n - \boldsymbol{\lambda}_{n+1} &= (\boldsymbol{\lambda}_{n+1} - \mathbf{1}) \cdot \frac{\partial \boldsymbol{\Delta}_{n+1}}{\partial \boldsymbol{\alpha}_{n+1}} \\ \boldsymbol{\lambda}_n - \boldsymbol{\lambda}_{n+1} &= \boldsymbol{\Gamma}_{n+2}^{\dagger} \cdot (\boldsymbol{\omega}_{n+2} \circ (\boldsymbol{\lambda}_{n+1} - \mathbf{1})) - \boldsymbol{\lambda}_{n+1} + \mathbf{1} \\ \boldsymbol{\lambda}_n &= \boldsymbol{\Gamma}_{n+2}^{\dagger} \cdot (\boldsymbol{\omega}_{n+2} \circ (\boldsymbol{\lambda}_{n+1} - \mathbf{1})) + \mathbf{1}. \end{aligned}$$

If we make the substitution

$$\boldsymbol{\kappa}_n = 1 - \boldsymbol{\lambda}_n$$

then this further simplifies to

$$\boldsymbol{\kappa}_n = \boldsymbol{\Gamma}_{n+2}^{\dagger} \cdot (\boldsymbol{\omega}_{n+2} \circ \boldsymbol{\kappa}_{n+1}),$$

which is just the backward states encountered above with a shifted index,

$$\boldsymbol{\kappa}_n = \boldsymbol{\beta}_{n-1}.$$

For the explicit derivative of the functional we also need

$$\frac{\partial j_n}{\partial \psi} - \boldsymbol{\lambda}_n \cdot \frac{\partial \boldsymbol{\Delta}_n}{\partial \psi} = (\mathbf{1} - \boldsymbol{\lambda}_n) \cdot \frac{\partial \boldsymbol{\Delta}_n}{\partial \psi} = \boldsymbol{\kappa}_n \cdot \frac{\partial \boldsymbol{\Delta}_n}{\partial \psi},$$

where

$$\frac{\partial \boldsymbol{\Delta}_n}{\partial \psi} = \frac{\partial \boldsymbol{\omega}_{n+1}}{\partial \psi} \circ (\boldsymbol{\Gamma}_{n+1} \cdot \boldsymbol{\alpha}_n) + \boldsymbol{\omega}_{n+1} \circ \left(\frac{\partial \boldsymbol{\Gamma}_{n+1}}{\partial \psi} \cdot \boldsymbol{\alpha}_n \right).$$

Lastly we work out the boundary term. Recalling $\upsilon = \omega_0 \circ \rho$, the boundary term is

$$\begin{aligned}
\left[\mathbf{1} + \frac{\partial j_0}{\partial \boldsymbol{\alpha}_0} - \boldsymbol{\lambda}_0 \cdot \frac{\partial \boldsymbol{\Delta}_0}{\partial \boldsymbol{\alpha}_0} - \boldsymbol{\lambda}_0\right]^\dagger \cdot \frac{\partial(\boldsymbol{\omega}_0 \circ \rho)}{\partial \psi} &= \left[\mathbf{1} + \mathbf{1} \cdot \frac{\partial \boldsymbol{\Delta}_0}{\partial \boldsymbol{\alpha}_0} - \boldsymbol{\lambda}_0 \cdot \frac{\partial \boldsymbol{\Delta}_0}{\partial \boldsymbol{\alpha}_0} - \boldsymbol{\lambda}_0\right]^\dagger \cdot \frac{\partial(\boldsymbol{\omega}_0 \circ \rho)}{\partial \psi} \\
&= \left[(\mathbf{1} - \boldsymbol{\lambda}_0) \cdot \frac{\partial \boldsymbol{\Delta}_0}{\partial \boldsymbol{\alpha}_0} + \mathbf{1} - \boldsymbol{\lambda}_0\right]^\dagger \cdot \frac{\partial(\boldsymbol{\omega}_0 \circ \rho)}{\partial \psi} \\
&= \left[\boldsymbol{\Gamma}_1^\dagger \cdot (\boldsymbol{\omega}_1 \circ (\mathbf{1} - \boldsymbol{\lambda}_0) - (\mathbf{1} - \boldsymbol{\lambda}_0) + \mathbf{1} - \boldsymbol{\lambda}_0\right]^\dagger \cdot \frac{\partial(\boldsymbol{\omega}_0 \circ \rho)}{\partial \psi} \\
&= \left[\boldsymbol{\Gamma}_1^\dagger \cdot (\boldsymbol{\omega}_1 \circ (\mathbf{1} - \boldsymbol{\lambda}_0)\right]^\dagger \cdot \frac{\partial(\boldsymbol{\omega}_0 \circ \rho)}{\partial \psi} \\
&= \left[\boldsymbol{\Gamma}_1^\dagger \cdot (\boldsymbol{\omega}_1 \circ \boldsymbol{\kappa}_0)\right]^\dagger \cdot \frac{\partial(\boldsymbol{\omega}_0 \circ \rho)}{\partial \psi} \\
&= \left[\boldsymbol{\Gamma}_1^\dagger \cdot (\boldsymbol{\omega}_1 \circ \boldsymbol{\kappa}_0)\right]^\dagger \cdot \left[\boldsymbol{\omega}_0 \circ \frac{\partial \boldsymbol{\rho}}{\partial \psi} + \frac{\partial \boldsymbol{\omega}_0}{\partial \psi} \circ \boldsymbol{\rho}\right].
\end{aligned}$$

Putting all of this together we can recover the derivative of the marginal likelihood by computing

$$\begin{aligned}
\frac{\mathrm{d}}{\mathrm{d}\psi}\pi(y_1, \ldots, y_N) &= \mathbf{1}^\dagger \cdot \frac{\mathrm{d}\boldsymbol{\alpha}_N}{\mathrm{d}\psi} \\
&= \left[\mathbf{1} + \frac{\partial j_0}{\partial \boldsymbol{\alpha}_0} - \boldsymbol{\lambda}_0 \cdot \frac{\partial \boldsymbol{\Delta}_0}{\partial \boldsymbol{\alpha}_0} - \boldsymbol{\lambda}_0\right]^\dagger \cdot \frac{\partial(\boldsymbol{\omega}_0 \circ \rho)}{\partial \psi} + \sum_{n=0}^{N-1} \frac{\partial j_n}{\partial \psi} - \boldsymbol{\lambda}_n^\dagger \cdot \frac{\partial \boldsymbol{\Delta}_n}{\partial \psi} \\
&= \left[\boldsymbol{\Gamma}_1^\dagger \cdot (\boldsymbol{\omega}_1 \circ \boldsymbol{\kappa}_0)\right]^\dagger \cdot \left[\boldsymbol{\omega}_0 \circ \frac{\partial \boldsymbol{\rho}}{\partial \psi} + \frac{\partial \boldsymbol{\omega}_0}{\partial \psi} \circ \boldsymbol{\rho}\right] \\
&\quad + \sum_{n=0}^{N-1} \boldsymbol{\kappa}_n^\dagger \cdot \left[\frac{\partial \boldsymbol{\omega}_{n+1}}{\partial \psi} \circ \left(\boldsymbol{\Gamma}_{n+1} \cdot \boldsymbol{\alpha}_n\right) + \boldsymbol{\omega}_{n+1} \circ \left(\frac{\partial \boldsymbol{\Gamma}_{n+1}}{\partial \psi} \cdot \boldsymbol{\alpha}_n\right)\right],
\end{aligned}$$

equivalent to the result from differentiating the expanded recursion.

One advantage to the discrete adjoint method is that we don't have to completely expand the recursion analytically, as done in the above derivation, or computationally, as would be done in a direct application of automatic differentiation. Instead we can reason about the derivatives *sequentially* in the same way that the system is originally defined.

4. CONCLUSION

In analogy to the continuous adjoint methods used with ordinary differential equations, the discrete adjoint method defines a procedure to efficiently evaluate the derivatives of functionals over the evolution of discrete sequences. Because this procedure is fully defined by the derivatives of the forward difference equations and the summands defining the functional, it defines an efficient sequential differentiation algorithm that mirrors the structure of the original sequence. The beneficial scaling of this procedure makes the resulting implementations especially useful in practical applications.

We can apply the method to any mathematical model that depends on the parameters through an (implicit) forward difference equation. Once we have made this equation explicit the derivation of a differentiation algorithm is completely mechanical, minimizing the burden of its implementation.

ACKNOWLEDGEMENTS

We thank Bob Carpenter for helpful discussions.

REFERENCES

Bücker, H. M., Corliss, G., Hovland, P., Naumann, U. and Norris, B. (2006). *Automatic Differentiation: Applications, Theory, and Implementations*. Springer.

Cappé, O., Moulines, E. and Rydén, T. (2005). *Inference in hidden Markov models. Springer Series in Statistics*. Springer, New York.

Griewank, A. and Walther, A. (2008). *Evaluating derivatives*, Second ed. Society for Industrial and Applied Mathematics (SIAM), Philadelphia, PA.

Hannemann-Tamás, R., Muñoz, D. A. and Marquardt, W. (2015). Adjoint sensitivity analysis for nonsmooth differential-algebraic equation systems. *SIAM J. Sci. Comput.* **37** A2380–A2402.

Hindmarsh, A. and Serban, R. (2020). User Documentation for CVODES v5.1.0 Technical Report, Lawrence Livermore National Laboratory.

Margossian, C. C. (2019). A Review of automatic differentiation and its efficient implementation. *Wiley interdisciplinary reviews: data mining and knowledge discovery* **9**.