1 The problem

The big picture is the following. We have a system which behaves in some complicated, but known way. For example it may be described by an ODE or a PDE (a *model*). We want to optimize the behavior of the model with respect to some objective function. For example this could be a control problem (where we want the system to match some trajectory), or a data assimilation problem (we want to find the model which closely matches the system seen experimentally).

The mathematical setup for the problem is simple. The model is characterized by a set of parameters p (in principle the parameters may also be determined at each point in time and space, $p(x,t)$, but we suppress these arguments). The state of our system is given by $u(x,t)$ (for purely ODEs, this only depends on time, while for static PDEs it only depends on space). The model puts constraints on possible states we see, which are determined by the parameters, so that we may write solutions of the model as $u(x,t;p)$. The objective function is given by a functional $J(p)$ which might look like

$$J(p) = \frac{1}{2}\int \|u(x,t;p) - u_{\mathrm{target}}(x,t)\|^2 \,\mathrm{d}x\,\mathrm{d}t \tag{1}$$

where target could be experimental measurements (for data assimilation) or a desired system behavior (for control). Note that J only "sees" the parameters through their effect on the solution $u(x,t;p)$.

Our goal is to find the optimal set of parameters

$$p^* = \operatorname*{argmin}_p J(p). \tag{2}$$

To do so, we'd like to take derivatives $\frac{\mathrm{d}J}{\mathrm{d}p}$ which would allow us to do gradient descent

$$p^{n+1} = p^n - \eta\frac{\mathrm{d}J}{\mathrm{d}p}. \tag{3}$$

Why can't we do this? The issue, as hinted at above, is that J only sees p through its effect on the solution u, so that to compute the derivative we need to use the chain rule

$$\frac{\mathrm{d}J}{\mathrm{d}p} = \int \frac{\delta J}{\delta u(x,t;p)}\frac{\partial u(x,t;p)}{\partial p}\,\mathrm{d}x\,\mathrm{d}t \tag{4}$$

$$= \left\langle \frac{\delta J}{\delta u}, \frac{\partial u}{\partial p}\right\rangle \tag{5}$$

where we introduced the inner product notation to indicate integration over space and/or time. The first term $\frac{\delta J}{\delta u}$ is easy to calculate (for the mean square error term above it's just $\frac{\delta J}{\delta u} = (u - u_{\mathrm{target}})$). However, the second term $\frac{\partial u}{\partial p}$ (called the forward sensitivity) is generally not known and cannot be efficiently computed! This is bad.

2 The solution (the adjoint method)

The adjoint method provides us a way to compute the above gradients. From an operational perspective, the core insight is that one should just formulate the optimization problem as a constrained optimization problem (where the constraint is your ODE/PDE model) using the method of Lagrange multipliers. From a conceptual perspective, this amounts to introducing a new field $\lambda(x,t)$ which tells you how sensitive your loss is to instantaneous changes in p (at each t and x) which can be integrated to get the full derivative. This may not make sense now but hopefully it is a little bit more clear after we go through the derivation.

As mentioned above, we are going to write our objective as a constrained optimization problem. What is our constraint? Consider the simple ODE

$$\partial_t u(t) = f(u;p). \tag{6}$$

This can be written as a constraint that should be satisfied at each point in time:

$$\mathcal{G}[u;p] = \partial_t u - f(u;p) \overset{!}{=} 0. \tag{7}$$

With this we can use the method of Lagrange multipliers to write our Lagrangian, which we will minimize to find the optimum. The Lagrangian is

$$\begin{aligned} \mathcal{L}(u,p,\lambda) &= J(u,p) - \int \lambda(x,t)\mathcal{G}[u;p](x,t)\,\mathrm{d}x\,\mathrm{d}t & (8)\\ &= J(u,p) - \langle \lambda, \mathcal{G}[u;p]\rangle & (9)\end{aligned}$$

Note that the constraint must be satisfied at each x and t, so that the Langrange multiplier λ becomes a field. Second note that now the field u has become an argument of the Lagrangian (and the objective J) because we will treat it as a free variable that is determined by our minimization procedure. The optimum is the point (p^*, u^*, λ^*) where

$$\frac{\partial \mathcal{L}}{\partial p}\Big|_{(p^*,u^*,\lambda^*)} = 0 \text{ and } \frac{\delta \mathcal{L}}{\delta u}\Big|_{(p^*,u^*,\lambda^*)} = 0 \text{ and } \frac{\delta \mathcal{L}}{\delta \lambda}\Big|_{(p^*,u^*,\lambda^*)} = 0, \tag{10}$$

where δ denotes the variational derivative. Explicitly, these are

$$\begin{aligned} \frac{\partial \mathcal{L}}{\partial p} &= 0 \overset{!}{=} \frac{\partial J}{\partial p} - \left\langle \lambda, \frac{\partial \mathcal{G}}{\partial p} \right\rangle & (11)\\ \frac{\delta \mathcal{L}}{\delta u} &= 0 = \frac{\delta J}{\delta u} - \frac{\delta}{\delta u}\langle \lambda, \mathcal{G}\rangle & (12)\\ \frac{\delta \mathcal{L}}{\delta \lambda} &= 0 = -\mathcal{G} & (13)\end{aligned}$$

Ideally, we'd be able to solve all of these equations for our optimum. However, in practice only the last two are "easy" to solve. The last equation Eq. (13)

amounts to solving the constraint $\mathcal{G} = 0$ which, for any p, is achieved by solving a O/PDE for $u*$. The second to last Eq. (12), called the **adjoint equation**, provides the solution λ^* given a value of u^*. In general, it is also straightforward to solve and requires solving a O/PDE of very similar form to the one governing the dynamics of u. This will become clear in concrete examples below.

Unfortunately, the first condition Eq. (11) is not tractable in general which means that we cannot directly find the optimal p^* (which is what we want). However, it turns out that the entire right hand side can be computed easily given u^* and λ^*. This means that we *do* have enough information at this point to compute gradients

$$\frac{\mathrm{d}J}{\mathrm{d}p} \tag{14}$$

and hence to perform gradient descent. How? The logic consists of two steps.

First, note that given a solution (u^*, λ^*), the Lagrangian $\mathcal{L}$ becomes a function only of p, $\mathcal{L}(u^*_p, \lambda^*_p, p)$, where we write the subscript p suggestively to indicate that the solutions depend on a fixed value of p. Moreover, the Lagrangian, as a function of p, coincides exactly with the objective J

$$\mathcal{L}(u^*_p, \lambda^*_p, p) = J(u^*_p, p) \tag{15}$$

where the second term drops out because $\mathcal{G} = 0$. Because the functions are identical, their derivatives are identical,

$$\frac{\mathrm{d}J(u^*_p, p)}{\mathrm{d}p} = \frac{\mathrm{d}\mathcal{L}(u^*_p, \lambda^*_p, p)}{\mathrm{d}p} \tag{16}$$

The second step of the logic is straightforward; one simply expresses this derivative in terms that are easy to compute.

$$\begin{aligned}
\frac{\mathrm{d}J(u^*_p, p)}{\mathrm{d}p} &= \frac{\mathrm{d}\mathcal{L}(u^*_p, \lambda^*_p, p)}{\mathrm{d}p} \\
&= \frac{\partial \mathcal{L}}{\partial p}\Big|_{u^*_p, \lambda^*_p} + \underbrace{\left\langle \frac{\delta \mathcal{L}}{\delta u}, \frac{\partial u}{\partial p} \right\rangle}_{=0} + \underbrace{\left\langle \frac{\delta \mathcal{L}}{\delta \lambda}, \frac{\partial \lambda}{\partial p} \right\rangle}_{=0} \\
&= \frac{\partial J}{\partial p}\Big|_{u^*_p} - \left\langle \lambda^*, \frac{\partial \mathcal{G}}{\partial p}\Big|_{u^*_p, \lambda^*_p} \right\rangle
\end{aligned} \tag{17}$$

In going from the first line to the second, we used that at (u^*, λ^*) the partial derivatives $\partial\mathcal{L}/\partial\,(\cdot)$ are zero (cf. Eqs. (12) and (13)). In the second line, we just took the partial derivative that we already computed above in Eq. (11).

That's it! To summarize, the logic is as follows:

1. We have an optimization problem. Unfortunately, the parameters we want to optimize only appear through their influence on some O/PDE that determines a field u which we observe. This can be understood as a constrained optimization problem, where the O/PDE is our constraint.

2. We can formulate the problem using a Lagrangian $\mathcal{L}(u, \lambda, p)$

3. To find an optimum of the Lagrangian we have to solve three equations (Eqs. (11), (12), and (13)). Only two of these (the last two) are easily solved.

4. We solve the last two equations to get u^*, λ^*. Even though we can't solve the first equation directly to get the optimal p^*, we can still compute gradients!

5. It turns out that (evaluated at u_p^*) the gradients of $\mathcal{L}$ with respect to the parameters are the exact same as the gradients of J with respect to the parameters. This is great, because really at the very beginning all we want are gradients of the objective J in the first place.

6. Even better is that by taking this perhaps non-obvious sequence of steps, we arrive at an expression for the gradients which is easy to compute (Eq. (17)).

3 The adjoint method for an ODE

Let's make the above formulation a little more concrete (although you may want to just to an actual concrete example in the next section). Above I mentioned a few things above in passing, like "the adjoint equation is easy to solve", that "the adjoint equation results in an ODE of similar form to the ODE that u must obey" which are much clearer for a slightly more concrete example (the real concrete examples come next). I also used some notation for the inner product that was not properly defined.

To clarify these points let's consider a simple PDE example:

$$\partial_t \vec{u} = \vec{f}(\vec{u}, \vec{p}) \tag{18}$$

Let the state be $\vec{u} \in \mathbb{R}^n$ and the parameter vector be $\vec{p} \in \mathbb{R}^q$. For concreteness, let's say we want to use the parameters p to control the solution to follow a desired trajectory $\vec{u}_{\text{target}}(t)$. We want to minimize the average deviation over the whole trajectory

$$J(u, p) = \frac{1}{T} \int \|u(t) - u_{\text{target}}(t)\|^2 \, \mathrm{d}t \tag{19}$$

where it is important not to forget that the solution u will depend on the parameters p. I have also abandoned using vector notation for simplicity. The constraint is

$$\mathcal{G}[u; p](t) = \partial_t u(t) - f(u(t), p) \overset{!}{=} 0 \tag{20}$$

Note that this is actually many constraints: first, it is a constraint on each component of u, and second, because we require that the equation is satisfied *for all t*. This will become more obvious when we go to the discretized versions below. We can now write our Lagrangian,

$$\begin{aligned}
\mathcal{L}(u,p,\lambda) &= J(u,p) - \int \lambda^T(t)\mathcal{G}[u;p](t)\,\mathrm{d}t && (21)\\
&= J(u,p) - \langle \lambda, \mathcal{G}[u;p]\rangle && (22)\\
&= \frac{1}{T}\int \|u(t) - u_{\text{target}}(t)\|^2\,\mathrm{d}t - \int \lambda_i(t)\left(\partial_t u_i - f_i(u,p)\right)\mathrm{d}t && (23)
\end{aligned}$$

The inner product indicates integration over time

$$\langle f, g\rangle = \int f^T(t)g(t)\,\mathrm{d}t \tag{24}$$

for vectors which are functions of time, although we will also use Einstein summation convention according to which repeated indices are summed over.

We may now look at our partial derivatives. The first equation is

$$\begin{aligned}
\frac{\partial \mathcal{L}}{\partial p} &= -\left\langle \lambda, \frac{\partial \mathcal{G}}{\partial p}\right\rangle && (25)\\
&= \int \lambda_i(t)\frac{\partial}{\partial p} f_i(u(t),p)\,\mathrm{d}t. && (26)
\end{aligned}$$

The second, the adjoint equation, is

$$\begin{aligned}
\frac{\delta \mathcal{L}}{\delta u_i} &= \frac{\delta J}{\delta u} - \frac{\delta}{\delta u}\langle \lambda, \mathcal{G}\rangle && (27)\\
&= \frac{2}{T}(u_i(t) - u_{\text{target},i}(t)) - \frac{\delta}{\delta u_i}\int \lambda_j(t)\left(\partial_t u_j - f_j(u,p)\right)\mathrm{d}t && (28)\\
&= \frac{2}{T}(u_i(t) - u_{\text{target},i}(t)) + \frac{\delta}{\delta u_i}\left[\int \partial_t\lambda_j(t)u_j(t) + \lambda_j(t)f_j(u(t),p)\,\mathrm{d}t - \lambda^T(T)u(T) + \lambda^T(0)u(0)\right] && (29)\\
&= \frac{2}{T}(u_i(t) - u_{\text{target},i}(t)) + \partial_t\lambda_i(t) + \lambda_j(t)\frac{\partial}{\partial u_i}f_j(u,p) && (30)
\end{aligned}$$

Rearranged, we get

$$\partial_t\lambda_i(t) + \lambda_j(t)[Df]_{ij}(u,p) = -\frac{2}{T}(u_i(t) - u_{\text{target},i}(t)) \tag{31}$$

where we write $\partial_{u_i} f_j = [Df]_{ij}$. This is nothing but an ODE for λ! In general, because $f(u,p)$ may be a non-linear function of u, the dynamics of $\lambda(t)$ will be coupled to those of $u(t)$. Also, it is conceptually noteworthy that there is also a "source" term for the dynamics given by the derivative of J.

One tricky point which we must not forget are the boundary conditions. To look at these it helps to look at the variation $\delta\mathcal{L}$ instead of at the functional derivative. The variation is given by

$$\begin{aligned}
\delta\mathcal{L} &= \frac{\delta \mathcal{L}}{\delta u(t)}\delta u(t) - \lambda^T(T)\delta u(T) + \lambda^T(0)\delta u(0) && (32)\\
&= -\lambda^T(T)\delta u(T) + \lambda^T(0)\delta u(0) && (33)
\end{aligned}$$

This must be zero for any choice of δu. Let us assume that we have a fixed initial condition $u(0)$. Then the variation $\delta u(0) = 0$, and so we have

$$0 = \lambda^T(T)\delta u(T) \tag{34}$$

which is true for any $\delta u(T)$ only if $\lambda(T) = 0$. This gives us a boundary condition. (In hindsight I think one can just straightforwardly take the functional derivative, but it was a little less apparent to me that way). We see in this way that we will have to evolve from a fixed final state $\lambda(T)$: our ODE will be running *backwards* in time! This is standard for the adjoint method, and will become even more apparent when we do this exercise again for discretized dynamics.

The final condition is simply

$$\frac{\delta \mathcal{L}}{\delta \lambda} = 0 = -\mathcal{G} \tag{35}$$

which is satisfied if u is a solution to the differential equation.

Once we have solved all of these equations (which amounts to solving 2 ODEs – one for u, one for λ – we have all we need to compute the gradient,

$$\frac{\mathrm{d}J}{\mathrm{d}p} = \frac{\partial \mathcal{L}}{\partial p} = \int \lambda_i(t) \frac{\partial}{\partial p} f_i(u(t), p)\, \mathrm{d}t. \tag{36}$$

By doing gradient descent, we can therefore solve our problem for p^*.

4 A concrete ODE example

Let's take the equation for the van der Pol oscillator.

$$\begin{aligned} \frac{dx}{dt} &= y \\ \frac{dy}{dt} &= \mu(1 - x^2)y - x \end{aligned}$$

Given an observed trajectory, can we find what μ is? Let's imagine our trajectory is measured at the time points t_i, $i \in \{1, .., M\}$. Let's imagine that we can simulate our system at a slightly higher resolution, at t_j, $j \in \{1, .., N\}$, $\frac{N}{M} = 10$ (so we measure every 10 steps).

We will solve our ODE with a simple Euler scheme,

$$u_{k+1} = u_k + \Delta t\, f(u_k, \mu) \tag{37}$$

so our constraint is

$$G(u, \mu) = u_{k+1} - u_k - \Delta t\, f(u_k, \mu) \tag{38}$$

Just as $u = (x, y)$, we will let $\lambda = (\rho, \xi)$

The full Lagrangian is

$$\mathcal{L}(u, \lambda, \mu) = \frac{1}{M}\sum_{i=1}^{M}\|u(t_i) - u_{\text{target}}(t_i)\|^2 \tag{39}$$

$$- \sum_{k=0}^{N-1} \rho_k(x_{k+1} - x_k - \Delta t\, y_k) \tag{40}$$

$$- \sum_{k=0}^{N-1} \xi_k(y_{k+1} - y_k - \Delta t\,(\mu(1 - x_k^2)y_k - x_k)). \tag{41}$$

4.1 The adjoint equation

The adjoint equation is determined by the derivatives of the Lagrangian with respect to the state variable. These derivatives are

$$\frac{\partial \mathcal{L}}{\partial x_k} = \frac{2}{M}(x_k - x_{\text{targ},k})\delta(k \bmod N/M) - \rho_{k-1} + \rho_k - \Delta t\, \xi_k\,(2\mu x_k y_k + 1) \tag{42}$$

$$\frac{\partial \mathcal{L}}{\partial y_k} = \frac{2}{M}(y_k - y_{\text{targ},k})\delta(k \bmod N/M) - \xi_{k-1} + \xi_k(1 + \Delta t\mu(1 - x_k^2)) + \Delta t\, \rho_k. \tag{43}$$

By setting these to zero and rearranging in the form of an update rule we find

$$\rho_{k-1} = \rho_k - \Delta t\, \xi_k\,(2\mu x_k y_k + 1) + \epsilon_{x,k} \tag{44}$$

$$\xi_{k-1} = \xi_k + \Delta t\,\left(\rho_k + \xi_k \mu(1 - x_k^2)\right) + \epsilon_{y,k} \tag{45}$$

where $\epsilon_k(t)$ can be viewed as an external drive given by

$$\epsilon_k = \frac{2}{M}\delta(k \bmod N/M)\begin{pmatrix} x_k - x_{\text{targ},k} \\ y_k - y_{\text{targ},k}\end{pmatrix} \tag{46}$$

which we see consists of a series of "pulses" at the measurement times. This is an update rule that solves for $(\rho(t), \xi(t))$ *backwards* in time, just as we had alluded to above. What about the boundary conditions? Assuming our initial conditions are fixed, we need not compute the variation with respect to (x_0, y_0) because they won't vary. The derivative with respect to the final state is

$$\frac{\partial \mathcal{L}}{\partial x_N} = -\rho_{N-1} + \frac{2}{M}(x_N - x_{\text{targ},N}) \tag{47}$$

$$\frac{\partial \mathcal{L}}{\partial y_N} = -\xi_{N-1} + \frac{2}{M}(y_N - y_{\text{targ},N}) \tag{48}$$

We see that this gives conditions on the final adjoint state,

$$\begin{pmatrix}\rho_{N-1} \\ \xi_{N-1}\end{pmatrix} = \vec{\lambda}_{N-1} = \vec{\epsilon}_N \tag{49}$$

which is simply set by the final error term. Why does the adjoint state end at index $N-1$? This is an artifact of our indexing, but it is correct. To see this, note that our state u_k, with $k \in \{0, ..., N\}$ has values at $N+1$ points. The first is the initial condition $k=0$, plus an additional N points that are simulated. The adjoint state describes the response to errors at all u_k which are determined by the ODE (and are not fixed). As there are only N of these, there are only N adjoint points to determine. The first is set by a boundary condition, and the remaining $N-1$ are solved for with the recurrence relations defined above.

4.2 The parameter update rule

We start by recalling the derivative of the objective with respect to the parameters and writing it for our discrete example:

$$\begin{aligned}\frac{\mathrm{d}J}{\mathrm{d}\mu} &= \sum_{k=0}^{N-1} \Delta t \, \lambda_{i,k} \frac{\partial}{\partial \mu} f_i(x_k, y_k, \mu) \\ &= \Delta t \sum_{k=0}^{N-1} \xi_k \left(1 - x_k^2\right) y_k\end{aligned}$$

The sum only goes to $N-1$ because there is never an update $f(u_N, p)$ which would add another dependence on μ, so that in total there are only N updates (from $k=0$ to $N-1$).

5 Discretize-then-optimize vs. optimize-then-discretize

TO DO!