

Optimal control via second order sensitivity analysis

Simon Zimmermann Roi Poranne Stelian Coros

We show how to solve optimal control problems of the form

$$\min_u O(\mathbf{x}, \mathbf{u}), \quad (1a)$$

$$\text{s.t.} \quad \mathbf{g}(\mathbf{x}, \mathbf{u}) = 0, \quad (1b)$$

where \mathbf{x} are *state* variables, \mathbf{u} are *control* variables. We use second order sensitivity analysis, and this short write-up presents a very sanitized derivation. For a conceptually similar but more formal derivation, we refer the reader to [1]. The challenge of solving (1) usually comes from the non-linearity of (1b). Ideally, (1b) would yield an explicit relationship $\mathbf{x}(\mathbf{u})$, which would then be substituted into (1a), resulting in the *unconstrained* problem

$$\min_u O(\mathbf{x}(\mathbf{u}), \mathbf{u}). \quad (2)$$

However, this not generally the case, and in fact, not quite that necessary. Indeed, the main reason for transforming (1) into (2) is for easily computing the derivatives of $O(\mathbf{x}(\mathbf{u}), \mathbf{u})$, to use with a gradient-based optimization. However, the derivative can actually be readily computed using the implicit function theorem.

We begin by applying the chain rule on $O(\mathbf{x}(\mathbf{u}), \mathbf{u})$:

$$\frac{dO}{d\mathbf{u}} = \frac{\partial O}{\partial \mathbf{x}} \frac{d\mathbf{x}}{d\mathbf{u}} + \frac{\partial O}{\partial \mathbf{u}}, \quad (3)$$

The term $\mathbf{S} := \frac{d\mathbf{x}}{d\mathbf{u}}$ is known as the *sensitivity* term. The analytic expression for it can be found using the fact that $\mathbf{g}(\mathbf{x}, \mathbf{u})$ is always zero (i.e. we assume that for *any* \mathbf{u} we can compute $\mathbf{x}(\mathbf{u})$ such that Eq. 1b is satisfied), which implies:

$$\frac{d\mathbf{g}}{d\mathbf{u}} = \frac{\partial \mathbf{g}}{\partial \mathbf{x}} \mathbf{S} + \frac{\partial \mathbf{g}}{\partial \mathbf{u}} = 0. \quad (4)$$

By rearranging this equation, we get:

$$\mathbf{S} = - \left(\frac{\partial \mathbf{g}}{\partial \mathbf{x}} \right)^{-1} \frac{\partial \mathbf{g}}{\partial \mathbf{u}}, \quad (5)$$

and plugging into (3), we obtain:

$$\frac{dO}{d\mathbf{u}} = -\frac{\partial O}{\partial \mathbf{x}} \left(\frac{\partial \mathbf{g}}{\partial \mathbf{x}} \right)^{-1} \frac{\partial \mathbf{g}}{\partial \mathbf{u}} + \frac{\partial O}{\partial \mathbf{u}}. \quad (6)$$

We note that through a reordering of matrix multiplications, the well-known adjoint method avoids computing $\frac{d\mathbf{x}}{d\mathbf{u}}$ directly as it evaluates $\frac{dO}{d\mathbf{u}}$. This is often-times more computationally efficient. However, we can leverage $\frac{d\mathbf{x}}{d\mathbf{u}}$ to derive a second-order solver that exhibits much better convergence properties than first order alternatives.

To begin with, we differentiate (3):

$$\frac{d^2O}{d\mathbf{u}^2} = \frac{d}{d\mathbf{u}} \frac{dO}{d\mathbf{u}} = \frac{d}{d\mathbf{u}} \left(\frac{\partial O}{\partial \mathbf{x}} \mathbf{S} \right) + \frac{d}{d\mathbf{u}} \frac{\partial O}{\partial \mathbf{u}}. \quad (7)$$

The formulas above involve third-order tensors, which lead to notation that is slightly cumbersome. For conciseness, we treat tensors as matrices and assume that contractions are clear from context. The second term in Eq. 7 is straightforward:

$$\frac{d}{d\mathbf{u}} \frac{\partial O}{\partial \mathbf{u}} = \mathbf{S}^T \frac{\partial^2 O}{\partial \mathbf{x} \partial \mathbf{u}} + \frac{\partial^2 O}{\partial \mathbf{u}^2}, \quad (8)$$

while the first term evaluates to

$$\frac{d}{d\mathbf{u}} \left(\frac{\partial O}{\partial \mathbf{x}} \mathbf{S} \right) = \left(\frac{d}{d\mathbf{u}} \frac{\partial O}{\partial \mathbf{x}} \right) \mathbf{S} + \frac{\partial O}{\partial \mathbf{x}} \left(\frac{d}{d\mathbf{u}} \mathbf{S} \right), \quad (9)$$

with

$$\frac{d}{d\mathbf{u}} \frac{\partial O}{\partial \mathbf{x}} = \mathbf{S}^T \frac{\partial^2 O}{\partial \mathbf{x}^2} + \frac{\partial^2 O}{\partial \mathbf{x} \partial \mathbf{u}}. \quad (10)$$

Here, $\frac{d}{d\mathbf{u}} \mathbf{S}$ is a third-order tensor, and $\frac{\partial O}{\partial \mathbf{x}} \left(\frac{d}{d\mathbf{u}} \mathbf{S} \right)$ stands for

$$\frac{\partial O}{\partial \mathbf{x}} \left(\frac{d}{d\mathbf{u}} \mathbf{S} \right) = \sum_i \frac{\partial O}{\partial x_i} \left(\frac{d^2 x_i}{d\mathbf{u}^2} \right).$$

The second-order sensitivity term $\frac{d}{d\mathbf{u}} \mathbf{S}$ must be further broken down:

$$\frac{d}{d\mathbf{u}} \mathbf{S} = \left(\mathbf{S}^T \frac{\partial}{\partial \mathbf{x}} \mathbf{S} + \frac{\partial}{\partial \mathbf{u}} \mathbf{S} \right). \quad (11)$$

The partial derivatives of \mathbf{S} can be found by taking the second derivatives in (4) and rearranging the terms. This results in

$$\frac{\partial}{\partial \mathbf{x}} \mathbf{S} = - \left(\frac{\partial \mathbf{g}}{\partial \mathbf{x}} \right)^{-1} \left(\frac{\partial^2 \mathbf{g}}{\partial \mathbf{x}^2} \mathbf{S} + \frac{\partial^2 \mathbf{g}}{\partial \mathbf{u} \partial \mathbf{x}} \right), \quad (12)$$

$$\frac{\partial}{\partial \mathbf{u}} \mathbf{S} = - \left(\frac{\partial \mathbf{g}}{\partial \mathbf{x}} \right)^{-1} \left(\frac{\partial^2 \mathbf{g}}{\partial \mathbf{x} \partial \mathbf{u}} \mathbf{S} + \frac{\partial^2 \mathbf{g}}{\partial \mathbf{u}^2} \right), \quad (13)$$

where once again we assume that the tensor expressions are self-evident. Combining all of the terms above leads to the following formula for the Hessian:

$$\frac{d^2O}{d\mathbf{u}^2} = \frac{\partial O}{\partial \mathbf{x}} \left(\mathbf{S}^T \frac{\partial}{\partial \mathbf{x}} \mathbf{S} + \frac{\partial}{\partial \mathbf{u}} \mathbf{S} \right) + \mathbf{S}^T \left(\frac{\partial^2 O}{\partial \mathbf{x}^2} \mathbf{S} + 2 \frac{\partial^2 O}{\partial \mathbf{x} \partial \mathbf{u}} \right) + \frac{\partial^2 O}{\partial \mathbf{u}^2}. \quad (14)$$

Generalized Gauss-Newton Although Newton’s method generally converges much faster than L-BFGS or gradient descent, there are two issues with it. First, evaluating the second-order sensitivity term takes a non-negligible amount of time. Second, the Hessian is often indefinite, and needs to be regularized. Both problems can be dealt with by simply excluding the tensor terms in (14). The result is a generalized Gauss-Newton approximation for the Hessian:

$$\mathbf{H} = \mathbf{S}^T \frac{\partial^2 O}{\partial \mathbf{x}^2} \mathbf{S} + 2\mathbf{S}^T \frac{\partial^2 O}{\partial \mathbf{x} \partial \mathbf{u}} + \frac{\partial^2 O}{\partial \mathbf{u}^2}. \quad (15)$$

Although \mathbf{H} is not guaranteed to be positive-definite, we note that in many cases, O is a convex function of \mathbf{x} and \mathbf{u} , and therefore the first and last terms are always positive definite. Additionally, O commonly does not explicitly couple \mathbf{x} and \mathbf{u} , and therefore the mixed derivative (i.e. the second) term vanishes, which means that overall \mathbf{H} is positive definite.

Sensitivity analysis for trajectories The implicit relationship described by Eq. (1b) is very general, and can easily be derived for different types of dynamical systems. Turning directly to the time-discretized setting, the state vector \mathbf{x} has dimension nT , where T is the number of time steps and n is the number of variables representing the state at one time step. In this setting, we let \mathbf{x}_i , the i -th n -block in \mathbf{x} , denote the configuration of the system at time t_i . The dynamical system obeys the *state equation*

$$\mathbf{g}(\mathbf{x}_i, \dot{\mathbf{x}}_i, \ddot{\mathbf{x}}_i, \mathbf{u}_i) = 0 \quad (16)$$

where $\dot{\mathbf{x}}_i, \ddot{\mathbf{x}}_i$ are time-discretized velocity and acceleration. For a simple, first order approximation, $\dot{\mathbf{x}} = (\frac{\mathbf{x}_i - \mathbf{x}_{i-1}}{h})$ and $\ddot{\mathbf{x}} = (\frac{\mathbf{x}_i - 2\mathbf{x}_{i-1} + \mathbf{x}_{i-2}}{h^2})$, and so (16) can also be written as

$$\mathbf{g}(\mathbf{x}_i, \mathbf{x}_{i-1}, \mathbf{x}_{i-2}, \mathbf{u}_i) = 0 \quad (17)$$

Starting from an input *control* trajectory \mathbf{u} , and two fixed configurations, \mathbf{x}_0 and \mathbf{x}_{-1} , the entirety of \mathbf{x} can be via forward simulation.

The standard numerical integration scheme also reveals the structure of the implicit relationship described by Eq. (1b). It is easy to see that the time domain imposes a very specific structure on the system of equations that must be solved to compute the Jacobian $\frac{d\mathbf{x}}{d\mathbf{u}}$ in Eq. 4. This structure is visualized in Fig. 1, and can be exploited to speed up computations. In particular, since \mathbf{g} depends explicitly only on $\mathbf{x}_i, \mathbf{x}_{i-1}, \mathbf{x}_{i-2}$ and \mathbf{u}_i (i.e. all other partial derivatives are 0), $\frac{\partial \mathbf{g}}{\partial \mathbf{u}}$ has a block diagonal form, and $\frac{\partial \mathbf{g}}{\partial \mathbf{x}}$ has a *banded* block diagonal form. This allows us to solve the resulting system using block forward-substitution, rather than storing and solving the entire linear system represented by $\frac{\partial \mathbf{g}}{\partial \mathbf{x}}$. We also note that the resulting \mathbf{S} is block triangular, which correctly indicates that \mathbf{x}_i does not depend on \mathbf{u}_j if $j > i$, or intuitively, the control at any moment in time only affect future states.

