3 Integral Equations and Nonlinear Iterative Optimization

In this lecture we largely will talk about derivatives of integral equations so that we can apply chain rule to an objective function $J(x(t), u(t))$. First, we need to be able to represent an ordinary differential equation as an integral equation, to make it easier to differentiate with respect to state and control variables.

3.1 Differential and integral equations

Let us remind ourselves of some things. Suppose we have a system such as:

$$\dot{x} = f(x(t), u(t)) \quad x(0) = x_0.$$

(We will typically suppress the dependence on t to avoid too much notation.) Then we know that an equivalent statement is that

$$x(t) = x_0 + \int_0^t f(x(\tau), u(\tau))d\tau.$$

This latter equation is the *integral* form of the differential equation. To see that the two are equivalent, note that we can check if $x(t)$ satisfies the differential equation. By the Leibniz rule we know that

$$\frac{d}{dt}\left[x_0 + \int_0^t f(x(\tau), u(\tau))d\tau\right] = f(x(t), u(t)).$$

So $x(t)$ satisfies the differential equation. Note that the differential equation incorporates the initial condition nicely.

This *integral representation* of the ordinary differential equation is now much easier to differentiate, specifically with respect to u. As a result, we can apply chain rule to a cost function that depends on $x(t)$ and $u(t)$ and evaluate the first derivative of $J(x(t), u(t)$ with respect to $u(t)$. When that derivative is equal to zero (for all possible directions $v(t)$), we know that we have a local minimizer, a local maximizer, or possibly a saddle point.

3.2 Derivatives of Objective Functions with Dynamic Constraints

Let us say we have an optimal control system where we cannot globally optimize the system by setting the derivative equal to zero. We might want to approach this problem by taking something along the lines of a gradient and then do gradient descent.[1] This would mean that our cost function looked like:

$$J = \int_0^T \ell(x, u)dt$$

and that we were minimizing J with respect to u subject to the constraint[2]

$$\dot{x} = f(x, u) \quad x(0) = x_0.$$

(Oftentimes ℓ will also depend explicitly on time t, but we will only include that later (e.g., when there is a time-varying reference trajectory); it does not impact the analysis.) To keep track of

[1] In fact, taking a second-order approach can be even better, but for the purposes of these notes we will focus on first-order methods similar to gradient descent.

[2] What we should minimize with respect to is a key decision; we could also minimize with respect to the pair (x, u) that satisfies the dynamics, or, possibly, the pair (α, μ) that does not satisfy the dynamics—we will discuss this at length in the next lecture. We could also minimize solely with respect to u, which is also a common choice.

notation a bit more easily, we set notation $\xi = (x, u)$ and then differentiate $J(\xi) = J((x, u))$ in the direction $\zeta = (z, v)$. We then get

$$DJ(\xi) \cdot \zeta = \int_0^T D\ell(\xi) \cdot \zeta dt,$$

where the Lagrangian ℓ is allowed to vary with time (such as if there is a reference trajectory). Just as in the finite dimensional case, discussed shortly where the gradient—and thus the descent direction—is *defined* by minimizing a quadratic model, we can now potentially *choose* a quadratic model and obtain a ζ as a minimizer of a quadratic model.

For instance, we can simply choose the quadratic model $g(\zeta)$ (yes, ζ, not ξ!) in the following way:

$$g(\zeta) = \int_0^T D_2\ell(t, \xi) \cdot \zeta dt + \frac{1}{2} \int_0^T \|\zeta\|^2 dt$$

where the norm on the space of ζ (we haven't really specified what space that is yet) is something we get to define. For instance, it could just be the Euclidean 2-norm (where $Q = I_{n\times n}$ and $R = I_{m\times m}$) or the weighted Euclidean 2-norm (where $Q = Q^T \geq 0$ and $R = R^T > 0$). The key thing is that we get to choose this quadratic model in whatever way is most useful for generating the descent direction.

We have not yet discussed what happens to the constraint, but you can probably imagine that since we are only *locally* optimizing the cost function using the quadratic model, the *linearization* might be an appropriate way of representing the constraint. In this case, the quadratic model optimal control problem ends up simply being a standard optimal control problem! *This is the key that makes these techniques work, because we have a globally optimal method for optimizing linear quadratic cost functions with linear dynamic constraints.*

How can we see that the constraint involves the linearization? Assume that $\xi(t)$ is such that $x(t)$ and $u(t)$ are related through the differential equation. That means that $x(t) = x_0 + \int_0^t f(\xi(\tau))d\tau$. Hence,

$$\xi(t) = \begin{bmatrix} x(t) \\ u(t) \end{bmatrix} = \begin{bmatrix} x_0 + \int_0^t f(\xi(\tau))d\tau \\ u(t) \end{bmatrix}$$

which implies (by taking the derivative of both sides with respect to ζ)

$$D\xi(t) \cdot \zeta(t) = \frac{d}{d\epsilon} \begin{bmatrix} x(t) + \epsilon z(t) \\ u(t) + \epsilon v(t) \end{bmatrix}_{\epsilon=0} = \frac{d}{d\epsilon} \begin{bmatrix} x_0 + \epsilon z_0 + \int_0^t f(\xi(\tau) + \epsilon\zeta(\tau))d\tau \\ u(t) + \epsilon v(t) \end{bmatrix}_{\epsilon=0}$$

which in turn implies

$$D\xi(t) \cdot \zeta(t) = \begin{bmatrix} z(t) \\ v(t) \end{bmatrix} = \begin{bmatrix} z_0 + \int_0^t \overbrace{D_1 f(x(\tau), u(\tau))}^{A(\tau)} z(\tau) + \overbrace{D_2 f(x(\tau), u(\tau))}^{B(\tau)} v(\tau) \\ v(t) \end{bmatrix} d\tau.$$

Note that this implies that $z(t)$ satisfies the differential equation

$$\dot{z} = A(t)z + B(t)v = D_1 f(x, u)z + D_2 f(x, u)v$$

such that $z(0) = z_0$.

Note that this derivation only requires that differentiating both sides of the integral equation $x(t) = x_0 + \int_0^t f(x(\tau), u(\tau))d\tau$ be a valid mathematical operation. Clearly one can differentiate both sides of an equality and still have equality; that is all we are taking advantage of here.

It is also important to recognize that the optimization is over ζ, *not* over ξ. (This hopefully reminds you of gradient descent in finite dimensions.) Hence, the $D\ell(\xi)$ is *fixed*–it is just some time-varying signal in the optimization problem. How do we optimize with respect to ζ? This turns out to be *very* similar to gradient descent in finite dimensions, but one has to interpret gradient descent appropriately.

3.3 Gradient Descent in Finite Dimensions

The fact that the above always depends on trajectories $(x(t), u(t))$, and is therefore infinite dimensional, often makes it challenging to have intuition about the optimization and what it means. To get better intuition, I will draw a (very solid) analogy to the finite dimensional setting of gradient descent.

The method of gradient descent is the algorithm most of us learned (in an ad-hoc way) to extremize functions. The basic idea is to find a direction that points "downhill" and then take a step in that direction. This is formalized in the following algorithm.

- Given initial data
- For $i = 0, 1, \ldots$
  - Determine a descent direction
    $$z_i = -\nabla f(x_i)$$
    One doesn't *need* to make this choice, but z_i should be capable of sufficient decrease.
  - Determine a step size (aka step length)
    $$\gamma_i = \arg\min_{\gamma > 0} f(x_i + \gamma z_i)$$
    One does not need to make this particular choice either.
  - update
    $$x_{i+1} = x_i + \gamma_i z_i$$
  - Repeat as needed

This algorithm will construct a sequence $\{x_i\}_{i=0}^{\infty}$ with $f(x_{i+1}) < f(x_i)\ \forall\ i$. What makes this complicated is illustrated below in the figure. A descent direction does not need to point directly at the local minimum. In particular, it will generally point in a direction that leads to the cost going *up* for a large enough step size.

What questions can we ask about the gradient descent algorithm? If we employ gradient descent, are we guaranteed to descend? To converge? To meet necessary conditions of optimility if we do successfully converge? These are the fundamental questions that have to be answered, and it turns out that a relatively simple algorithm has guarantees on all these questions.

Now, where does gradient descent come from? That is, how does one obtain $z_i = -\nabla f(x_i)$? We look at the relationship between the derivative and the gradient to find out. The gradient is *defined* by the following relationship.

$$Df(x) \cdot z = \langle \nabla f(x), z \rangle$$

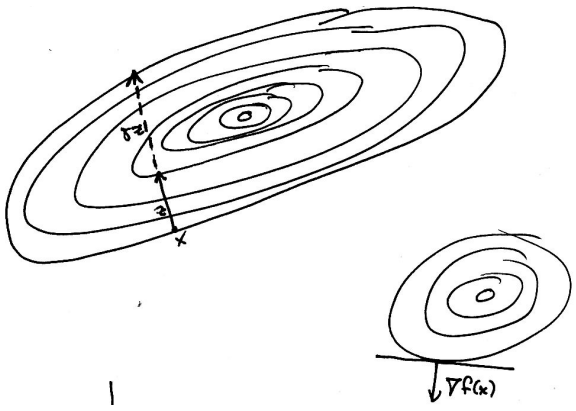

Figure 7: Level set with gradient/line search included

Hence, one might want to maximize the amount of change in f one gets by choosing the right direction z, subject to not using too large of a z (modeled by a quadratic term).

$$\min_z Df(x) \cdot z + \frac{1}{2}||z||^2$$

We can rewrite this as

$$\min_z \langle \nabla f(x), z \rangle + \langle z, z \rangle$$

and then minimize by taking the derivative with respect to a perturbation w to z (not to x) and setting the result equal to zero.

$$\langle \nabla f(x), w \rangle + \langle z, w \rangle = \langle \nabla f(x) + z, w \rangle = 0 \quad \forall w$$

which implies that

$$z = -\nabla f(x).$$

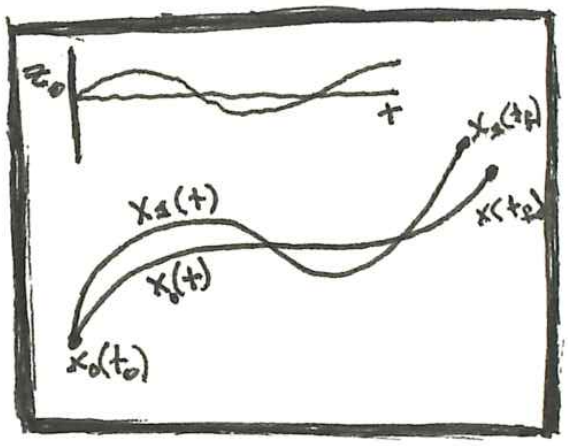

Figure 8: A trajectory $x_0(t)$ and a perturbation of the trajectory $x_1(t) = x_0(t) + \gamma z_0(t)$.

The analog to finite dimensional gradient descent uses the descent direction from the equation shown earlier, and if we can solve for $(z(t), v(t))$ by doing that minimization, we can potentially do minimization in a way that directly mimics gradient descent in finite dimensions. That iterative process is illustrated in Fig. 8 for the first iterate of an iterative scheme.

3.4 Armijo line searches

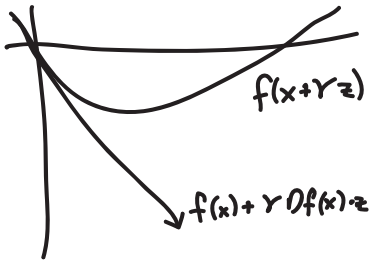

Figure 9: The Armijo line search

Let's go back and consider the map from $\gamma \mapsto f(x + \gamma z)$. We can approximate this map using different expansions of $f(\cdot)$. In particular, we can construct local quadratic models for determing the step size if we wish.

We must ensure that our step or update produces a sufficient decrease. Let's suppose that z_i is a descent direction for $f(\cdot)$ at x_i (i.e., $Df(x_i) \cdot z < 0$) (e.g., determined by a quadratic model). Suppose we want to avoid the situation where we converge *before* we get to an element of the vector space with zero derivative. This means we want to not only decrease at each step, we want to decrease *enough*. This is what Armijo realized.

The Armijo general sufficient decrease condition is:

$$f(x_i + \gamma z_i) \leq f(x_i) + \alpha\gamma Df(x_i) \cdot z_i \tag{4}$$

where $\alpha \in (0, 1)$ is some algorithmic parameter chosen by the designer.

The variable α is simply a constant chosen by the algorithm designer.[3] How should we choose γ? Armijo[4] says to use a sequence of form $\gamma = \beta^k$ ($k = 0, 1, 2, \ldots$) until Eq.(4) is satisfied and take the smallest k that satisfies Eq.(4). Modifications of this basic algorithm are certainly possible. Also, note that β is just another algorithmic parameter.[5]

Formal Properties of Armijo Line Searches

If you apply Armijo line searches, you will only converge to a point where the directional derivative is equal to zero.

[3] For α a good choice can be $\alpha = 0.4$. Others, like Kelley, like $\alpha = 10^{-4}$.

[4] The Armijo step size rule is also sometimes known as backtracking line search.

[5] A good choice is $\beta = 0.7$.

Optimization Using Armijo

Given x_0, $f(\cdot)$, $Df(\cdot)$, $\alpha \in (0, \frac{1}{2})$, $\beta \in (0,1)$, and maybe ϵ
While $\|\nabla f(x_i)\| > \epsilon$
Choose z_i by solving $z_i = \arg\min Df(x_i) \circ z + \langle z, z \rangle$
$n = 0$ $\gamma = \beta^n$
While $f(x_i + \gamma z_i) > f(x_i) + \alpha\gamma Df(x_i) \circ z_i$
$n = n + 1$
$\gamma = \beta^n$
end while
$x_{i+1} = x_i + \gamma z_i$
end while

Questions the student should be asking include the following. Does Armijo converge? Can we get an idea of how small a γ may be required? (Yes, in particular if we obtain z_i by minimizing a quadratic model function and even for general z_i.) Does Armijo guarantee that the algorithm converges to a point that satisfies some condition of optimality? Before getting to those questions we need to show that a γ even exists that satisfies Eq.(4) for γ small enough.

For formal results on Armijo line searches, including the fact that using an Armijo line guarantees convergence to a point that satisfies $Df(x) = 0$, see the terrific book *Iterative Methods for Optimization* by C.T. Kelley.

Examples

Consider the system $\begin{bmatrix} \dot{x}_1 \\ \dot{x}_2 \end{bmatrix} = \begin{bmatrix} \cos(x_2) \\ -\sin(x_1) + \cos(u) \end{bmatrix}$ What ordinary differential equation does a perturbation to a particulate $(x(t), u(t))$ look like?

$$A(t) = D_1 f(x,u) = \begin{bmatrix} 0 & -\sin(x_2) \\ -\cos(x_1) & 0 \end{bmatrix} \text{ and } B(t) = D_2 f(x,u) = \begin{bmatrix} 0 \\ -\sin(u) \end{bmatrix}$$

As a result, perturbations $z(t)$ must satisfy the differential equation $\dot{z} = A(t)z + B(t)v$ with $z(0) = z_0$, where z_0 and $v(t)$ parameterize all possible perturbations.