Reinforcement Learning and Optimal Control

IFT6760C, Fall 2021

Pierre-Luc Bacon

December 12, 2021

Linear Quadratic Regulation

Assumption: cost function is quadratic and the dynamics are linear.

$$
\begin{aligned}
&\text{minimize} \;\; x_T Q_T x_T + \sum_{t=1}^{T} \left(x_t^\top Q x_t + u_t^\top R u_t \right) \\
&\text{subject to} \;\; x_{t+1} = A x_t + B u_t, \;\; t = 1, \ldots T - 1 \\
&\text{given} \;\; x_1 \;\; .
\end{aligned}
$$

Backward Induction for LQR: Ricatti Equation

We can show that the value function is quadratic. Therefore, we can write the cost-to-go function as $J_t(x_t) \triangleq x_t^\top P_t x_t$. Hence, we need to find a set of matrices $\{P_1, \ldots, P_T\}$ by backward induction:

- Set $P_T = Q_T$
- From $t = T - 1, \ldots, 1$:
  - Set $P_t = Q + A^\top P_{t+1} A - A^\top P_{t+1} B (R + B^\top P_{t+1} B)^{-1} B^\top P_{t+1} A$
  - Set $K_t = -(R + B^\top P_{t+1} B)^{-1} B^\top P_{t+1} A$

You can then compute the optimal control at time t in state x_t with $u_t^\star = K_t x_t$ (the optimal controls are linear in the states).

Forward Sensitivity Equation

Let's consider a parameterized ODE of the form:

$$\dot{x}(t,\theta) = f(x(t),\theta), \quad x(t_0,\theta) = x_0(\theta) \quad ,$$

Taking the total derivative:

$$\begin{aligned} D_2 D_1 x(t,\theta) &= D_1 f(x(t,\theta),\theta) D_2 x(t,\theta) + D_2 f(x(t,\theta),\theta) \\ D_2 x(t,\theta) &= D x_0(\theta) \quad . \end{aligned}$$

Using the symmetry of second derivatives (see Schwarz's theorem, Clairaut's theorem, or Young's theorem), we can write instead:

$$D_1 D_2 x(t,\theta) = D_1 f(x(t,\theta),\theta) D_2 x(t,\theta) + D_2 f(x(t,\theta),\theta) \quad .$$

Forward Sensitivity Equation

If we define $s(t,\theta) \triangleq D_2 x(t,\theta)$ ("s" as in *sensitivity*), we get the *forward sensitivity equation*:

$$
\begin{aligned}
D_1 s(t,\theta) &= D_1 f(x(t,\theta),\theta) s(t,\theta) + D_2 f(x(t,\theta),\theta) \\
s(t_0,\theta) &= D x_0(\theta) \quad .
\end{aligned}
$$

We can easily evaluate the Jacobian of f by augmenting the dynamics system f with the corresponding sensitivity equation. We can then solve for the original IVP simultaneously with its sensitivities.

Indirect Single Shooting for Parameterized IVPs

Goal: find the parameters θ which minimize a given cost function.

$$
\begin{aligned}
&\text{minimize} \;\; c(x(t_f, \theta)) \\
&\text{subject to} \;\; \dot{x}(t, \theta) = f(x(t, \theta), \theta) \\
&\text{given} \;\; x(t_0, \theta) = x_0(\theta) \quad .
\end{aligned}
$$

Next slide: we will minimize this objective by gradient descent.

Indirect Single Shooting for Parameterized IVPs

Repeat:

- Solve the augmented IVP (eg. Euler, RK4):

$$\dot{z}(t, \theta^{(k)}) = \tilde{f}(z(t, \theta^{(k)}), \theta^{(k)}) = \begin{pmatrix} f(x(t, \theta^{(k)}), \theta^{(k)}) \\ D_1 f(x(t, \theta^{(k)}), \theta^{(k)}) s(t, \theta^{(k)}) + D_2 f(x(t, \theta), \theta) \end{pmatrix}$$

$$\text{with} \quad z(t_0, \theta^{(k)}) = \begin{pmatrix} x_0(\theta^{(k)}) \\ s(t_0, \theta^{(k)}) \end{pmatrix}$$

- Compute the gradient: $\Delta^{(k)} = Dc(x(t_f))s(t_f, \theta^{(k)})$
- Take a gradient step: $\theta^{(k+1)} = \theta^{(k)} - \eta \Delta^{(k)}$

Direct Single Shooting for Parameterized IVPs

A "direct" approach to this problem is to first discretize the system, and then differentiate through it. For example, let's use Euler discretization with n intervals:

$$x^{(i+1)} = x^{(i)} + h^{(i)} f(x^{(i)}, \theta^{(k)}) \quad .$$

What is the corresponding **discrete** forward sensitivity equation? Using the approach used earlier, define $\phi_i(\theta) = x_i$. We want $D\phi_{n+1}(\theta^{(k)})$:

$$D\phi_{n+1}(\theta^{(k)}) = D_1 f(x^{(n)}, \theta^{(k)}) D\phi_n(\theta^{(k)}) + D_2 f(x^{(n)}, \theta^{(k)}) \quad .$$

Here again, we can form an augmented system to get both the state (approximate) and sensitivity (exact wrt to approximate sequence) at once.

Direct Single Shooting for Parameterized IVPs

- Iterate the augmented system:

$$z^{(i+1)} = \tilde{f}(z^{(i)}, \theta) = \begin{pmatrix} z^{(i)} + h^{(i)} f(x^{(i)}, \theta^{(k)}) \\ D_1 f(x^{(i)}, \theta^{(k)}) D\phi_i(\theta^{(k)}) + D_2 f(x^{(i)}, \theta^{(k)}) \end{pmatrix}$$

$$\text{with } \; z^{(1)} = \begin{pmatrix} x_0(\theta) \\ I \end{pmatrix}$$

- Compute the gradient: $\Delta^{(k)} = Dc(x^{(n+1)}) D\phi_{n+1}(\theta^{(k)})$
- Take a gradient step: $\theta^{(k+1)} = \theta^{(k)} - \eta \Delta^{(k)}$

Multiple Shooting in BVPs

Consider a two-point boundary value problem of the form:

$$\dot{x}(t) = f(x(t)), \quad a < t < b$$
$$\psi(x(a), x(b)) = 0 \quad ,$$

That is, we need to find a trajectory which obeys the dynamics and satisfies the boundary constraint.

Multiple Shooting

The idea is to subdivide (a, b) into n intervals which will be treated as independent problems, whose solution is then stiched back together.

$$h(x_{t_1}, \ldots, x_{t_n}) = \begin{bmatrix} \phi_{t_2}(x_{t_1}) - x_{t_2} \\ \vdots \\ \phi_{t_{i+1}}(x_{t_i}) - x_{t_{i+1}} \\ \vdots \\ \phi_{t_N}(x_{t_n}) - x_{t_n} \\ \psi(x_{t_1}, \phi_b(x_{t_n})) \end{bmatrix} .$$

This is a nonlinear equation of the form $h(x_{t_1}, \ldots, x_{t_n}) = 0$ which we can solve by Newton's method.

Measle Outbreak

Ppopulation stays constant and can be partitioned into *susceptible* cases $S(t)$ (people who can contract measles), *infectives* $I(t)$ (people who can infect other people), *latents* $L(t)$ (people who have measles but can't spread it yet) and *immunes* $M(t)$ (recovered cases).

$$S(t) + I(t) + L(t) + M(t) = N \quad ,$$

where N is the total population. The dynamics are:

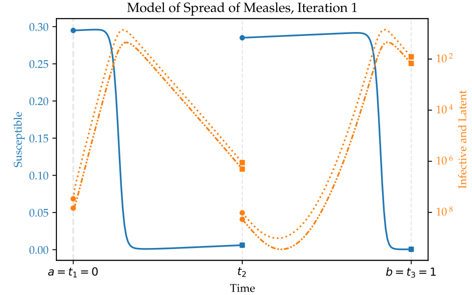
$$\begin{aligned}
\dot{x}_1 &= \mu - \beta(t)x_1x_3 \\
\dot{x}_2 &= \beta(t)x_1x_3 - \frac{x_2}{\lambda} \\
\dot{x}_3 &= \frac{x_2}{\lambda} - \frac{x_3}{\eta}, \quad 0 < t < 1 \\
\beta(t) &= \beta_0(1 + \cos(2\pi t)) \quad .
\end{aligned}$$

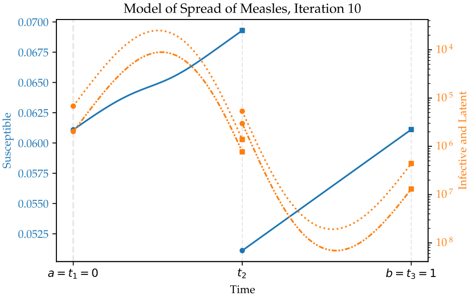
Boundary Condition

Find a value for the initial conditions such that at the end of the interval at $t = 1$ we got back to the samepoint: a periodic solution. That is:

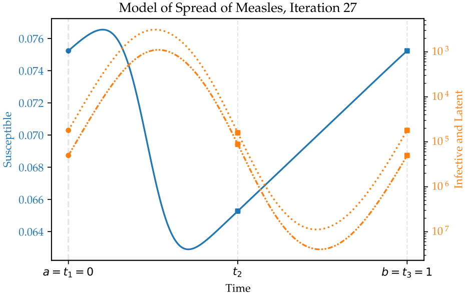
$$\psi(x(0), x(1)) \triangleq x(1) - x(0) = 0 \quad .$$

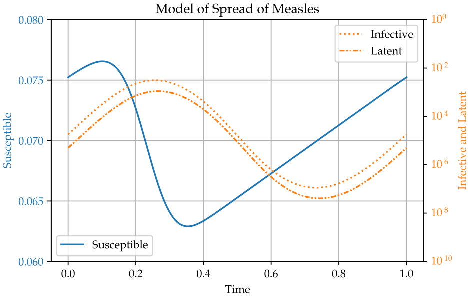
Visualization





Visualization





Collocation

The idea of collocation method in trajectory optimization is to approximate both the trajectory and the control function with a polynomial. This leads to a form of *implicit* simulation or integration in which the value of the states at the given *collocation points* is obtained *simultaneously*.

$$\dot{x}(t) = f(x(t), t), \quad x(t_0) = x_0 \quad .$$

Collocation Conditions

We can construct a collocation method by choosing a set of real numbers $0 \leq c_i \leq 1$ for $i = 1, \ldots, N$ and a class of polynomial functions. We then require that our *collocation polynomial* of degree N satisfies:

$$
\begin{aligned}
&p(t_0) = x_0 \\
&p'(t_0 + hc_i) = f(p(t_0 + hc_i), t_0 + hc_i), \quad i = 1, \ldots, N \quad ,
\end{aligned}
$$

where p' denotes the derivative. The value at the end of the interval is then:

$$
\tilde{x}_1 \triangleq p(t_0 + h) \quad .
$$

Example: Explicit Euler

Let's choose $N = 1$ with $c_1 = 0$, which means that we have to use a polynomial of the form:

$$p(t) = a_1(t - t_0) + a_0 \text{ and } p'(t) = a_1 \quad .$$

Applying the collocation conditions, we have:

$$\begin{aligned} p(t_0) &= a_0 = x_0 \\ p'(t_0) &= a_1 = f(p(t_0), t_0) \quad . \end{aligned}$$

Therefore, the value at the end of the time step must be:

$$p(t_0 + h) = x_0 + hf(x_0, t_0), t_0) \quad ,$$

which is the *explicit Euler method*