Trajectory Optimization

Chenggang Liu

Contents

Control vs. Planning

Control and planning are the two ends of a continuous spectrum. Control is more in a real-time and close-loop fashion while planning is more in an offline and open-loop fashion. There is no clear boundary between control and planning, for example, we can apply a control law and simulate forwards in time to make a open-loop long-term plan or we can apply a open-loop plan in an iterative fashion online to get a close-loop control, such as Model Predictive Control (MPC).

With the development of computer speed, open-loop planning methods, specifically trajectory optimization methods, become popular in real-time control and attract a lot of attention.

Optimal control problem

Continuous-time optimal control problem

- Functional forms:
  - Lagrange Formulation

$$J(u(\cdot), t_f) = \int_{t_0}^{t_f} L(\mathbf{x}, \mathbf{u}, t) dt \tag{1}$$

  - Bolza Formulation

$$J(u(\cdot), t_f) = \phi(\mathbf{x}(t_f), t_f) + \int_{t_0}^{t_f} L(\mathbf{x}, \mathbf{u}, t) dt \tag{2}$$

s.t.

$$\dot{\mathbf{x}} = f(\mathbf{x}, \mathbf{u}, t), \tag{3}$$

and the boundary conditions

$$\mathbf{x}(t_0) = \mathbf{x}_0, \quad \psi(\mathbf{x}_f, t_f) = 0 \tag{4}$$

where $\mathbf{x}(t) \in \mathbb{R}^n$ is the state, $\mathbf{u}(t) \in \mathbb{R}^m$ is the control, t is the independent variable (generally speaking, time), t_0 is the initial time, and t_f is the terminal time. The terms ϕ and L are called the terminal penalty term and Lagrangian, respectively.

  - Mayer Formulation In the Mayer formulation, the state vector is extended by: $x_{n+1}(t)$.

$$x_{n+1}(t) = \int_{t_0}^{t} L(\mathbf{x}, \mathbf{u}, t) dt. \tag{5}$$

Then the objective is to choose $\mathbf{u}(t)$ to minimize

$$J(u(\cdot), t_f) = \phi(\mathbf{x}(t_f), t_f) \tag{6}$$

s.t.

$$\dot{\mathbf{x}} = \begin{bmatrix} f(\mathbf{x}, \mathbf{u}, t) \\ l(\mathbf{x}, \mathbf{u}, t) \end{bmatrix} \tag{7}$$

and the boundary conditions

$$\mathbf{x}(t_0) = \begin{bmatrix} \mathbf{x}_0 \\ 0 \end{bmatrix} \tag{8}$$

$$\psi(\mathbf{x}_f, t_f) = 0$$

Mayer formulation and Bolza formulation are equivalent, but Mayer form yield a simpler expression.

Lagrange, Bolza, and Mayer forms are equivalent. In particular, there are constraints on the state of the form

$$S(\mathbf{x}(t)) \leq 0 \tag{9}$$

and on the control variable

$$C(\mathbf{x}(t), \mathbf{u}(t)) \leq 0 \tag{10}$$

The optimal control can be derived using Pontryagin's maximum principle (a necessary condition), or by solving the Hamilton-Jacobi-Bellman equation (a necessary and sufficient condition).

Discrete-time optimal control problem

The discrete-time optimal control problem:

$$J_k(x_k) = \min_{u_k \in \mathcal{U}} \{\phi(x_N) + \sum_{k=0}^{N-1} L_k(x_k, u_k)\}$$

s.t.

$$x_{k+1} = F(x_k, u_k, k)$$

Continuous DDP (Differential Dynamic Programming)

Calculus of Variation

In optimal control, we are trying to solve:

$$u = \arg\min_u J = \psi(x(t_f)) + \int_{t_0}^{t_f} L(x(t), u(t), t)dt$$

s.t.

$$\dot{x}(t) = f(x(t), u(t), t)$$

$$x(t_0) = x_0$$

The total cost J is a sum of the terminal cost and an integral along the way. The total cost J is also called performance index.

Using Lagrange method to augment the cost:

$$\bar{J} = L + \lambda^\top (f - \dot{x})$$

Assuming J is chosen to be continuous in x, u, and t, we write the variation as [1]:

$$\delta\bar{J} = \psi_x \delta x(t_f) + \int_{t_0}^{t_f} [L_x \delta x + L_u \delta u + \lambda^\top f_x \delta x + \lambda^\top f_u \delta u - \lambda^\top \delta\dot{(x)}]dt$$

where subscripts denote partial derivatives.

The last term:

$$-\int_{t_0}^{t_f} \lambda^\top \delta\dot{x} dt = -\lambda^\top \delta x|_{t_0}^{t_f} + \int_{t_0}^{t_f} \dot{\lambda} \delta x dt$$

and thus

$$\boxed{\begin{aligned}\delta\bar{J} = [\psi_x(x(t_f)) - \lambda(t_f)^\top]\delta x(t_f) + \lambda(t_0)^\top \delta x(t_0) + \\ \int_{t_0}^{t_f} (H_x + \dot{\lambda}^\top)\delta x dt + \int_{t_0}^{t_f} H_u \delta u dt\end{aligned}} \tag{11}$$

where $H = L + \lambda^\top f$ and is called Hamiltonian [2].

To extreme Eq. 11, there are three components of variation that must independently be zero since we can vary any of δx, δu, or $x(t_f)$:

$$L_x + \lambda^\top f_x + \dot{\lambda}^\top = 0$$

$$L_u + \lambda^\top f_u = 0$$

$$\psi_x(x(t_f)) - \lambda^\top(t_f) = 0$$

[1] Here we are assuming the augmented Lagrangian is differentiable. In fact, it is quite often for problems with bounded controls and terminal cost to have nondifferentiable optimal cost-to-go functions.

[2] Recall that we can get Hamiltonian from Lagrangian by calculating the momenta by differentiating the augmented Lagrangian with respect to the 'velocity', $\dot{x}$, as $p = \frac{\partial L}{\partial \dot{q}}$. Then the Hamiltonian is then given by $H = p\dot{q} - L$. As we can see that the Lagrange multiplier λ in Eq. 11 is actually the 'momenta', but with an opposite sign. This gives us an interesting insight into the optimal control problem.

The evolution of λ is given in reverse time, from the final state to the initial. This is the primary difficulty of solving optimal control problems.

Finally, we have the following equations that extreme the our objective function:

$$\begin{aligned} \dot{x} &= f(x(t), u(t), t) && (12)\\ \dot{\lambda} &= -L_x^\top - f_x^\top \lambda && (13)\\ L_u + \lambda^\top f_u &= 0 && (14) \end{aligned}$$

s.t.

$$x(t_0) = x_0$$
$$\lambda^\top(t_f) = \psi_x(x(t_f))$$

Note:

- $\delta x(t_0) = 0$ if the initial state is fixed.
- The function $L + \lambda^\top f$ is called Hamiltonian [3]. It is counterpart in discrete time is the Bellmen equation.
- If H is not differentiable w.r.t. u, then u can still be derived using the Pontryagin Maximum Principle:

$$u = \arg\min_u H(x, u, t, \lambda)$$

- The Lagrange multiplier is often called co-state.
- If we replace u in Eqs. 12 and 13 with the solution of the optimal control, then Eqs. 12 and 13 become equations only of x and λ, which are often called Hamiltonian Differential Equations.
- Denote V as the value function, then $V_x^\top$ is a solution of λ.

Gradient method

Gradient method is outlined as follows:

1. For a given x_0, pick a control history $u(t)$.
2. Solve ODE forwards in time to get state history:
   Propagate $\dot{x} = f(x(t), u(t), t)$ forward in time to create a state history
3. Solve ODE backwards in time to get co-state history:
   Evaluate $\psi_x(x(t_f))$, and propagate the co-state backwards in time from t_f to t_0
4. At each time step, minimize the Hamiltonian by:
   - choosing $\delta u = -\epsilon H_u$, where $\epsilon \in (0, 1]$
   - update $u = u + \delta u$
5. If $J^{i+1} < J^i$, go back to step 2 with $i = i + 1$. Otherwise, reduce the step size ϵ and repeat 4 (backtracking line search).
6. Exit if the early exit conditions are satisfied.

Newton-Raphson method (successive sweep method)

Newton-Raphson method uses second order variation to minimize the Hamiltonian, $H = L + \lambda^\top f$.

LaTeX

$$H(x + \delta x, u + \delta u, t + dt) = \frac{1}{2} \begin{bmatrix} 1 \\ \delta x \\ \delta u \end{bmatrix}^\top \left[\begin{array}{ccc} 2H & H_x & H_u \\ H_x^\top & H_{xx} & H_{xu} \\ \hline H_u^\top & H_{ux} & H_{uu} \end{array}\right] \begin{bmatrix} 1 \\ \delta x \\ \delta u \end{bmatrix} + H_t dt \quad (15)$$

The Newton step of δu is given by:

$$\begin{aligned} \delta u &= \arg\min_{\delta u} \delta H && (16)\\ &= -H_{uu}^{-1} H_u - H_{uu}^{-1} H_{ux} \delta x && (17) \end{aligned}$$

Compared with gradient method, there is an additional term of $-H_{uu}^{-1} H_{ux} \delta x$, which means λ is changed because δu depends on δx.

$$\dot{\lambda}^\top = -H_x - H_u u_x = -H_x + H_u H_{uu}^{-1} H_{ux} \quad (18)$$

$$\dot{\lambda_x}^\top = -H_{xx} - H_{ux} u_x = -H_{xx} + H_{ux} H_{uu}^{-1} H_{ux} \quad (19)$$

$$\begin{aligned} H_u &= L_u + \lambda f_u && (20)\\ H_{ux} &= L_{ux} + \lambda_x f_u + \lambda f_{ux} && (21)\\ H_{uu} &= L_{uu} + \lambda f_{ux} && (22) \end{aligned}$$

[3] 'Cool things in Physics and optimization' Link

Newton's method is outlined as follows:

1. For a given x_0, pick a control history $u(t)$.
2. For iteration i, solve ODE forwards in time to get state history $x(t)^i$: Propagate $\dot{x} = f(x(t), u(t), t)$ forwards in time to create a state history
3. Solve ODE backwards in time to get co-state history $\lambda(t)^i$: Evaluate $\psi_x(x(t_f))$ and $\psi_{xx}(x(t_f))$, and propagate $\lambda(t)$ and $\lambda_x(t)$ backwards in time from t_f to t_0
4. At each time step, minimize the Hamiltonian by: $u^{i+1} = u^i - \epsilon H_{uu}^{-1} H_u - H_{uu}^{-1} H_{ux} \delta x$
5. If $J^{i+1} < J^i$, go back to step 2 with $i = i + 1$. Otherwise, reduce the step size ϵ and repeat step 4 (backtracking line search).

Note:

- If the system dynamics is linear and the cost function is quadratic, one step convergence can be achieved with $\epsilon = 1$

Control parameters

Control parameters are constant variables in the optimal control problem structure, for example: $L(x, u, \alpha, t)$. According to the maximum principle: LaTeX

$$H(x + \delta x, \alpha + \delta\alpha, u + \delta u, , t) = \frac{1}{2} \begin{bmatrix} 1 \\ \delta x \\ \delta\alpha \\ \delta u \end{bmatrix}^\top \begin{bmatrix} 2H & H_x & H_\alpha & H_u \\ H_x^\top & H_{xx} & H_{x\alpha} & H_{xu} \\ H_\alpha^\top & H_{\alpha x} & H_{\alpha\alpha} & H_{\alpha u} \\ H_u^\top & H_{ux} & H_{u\alpha} & H_{uu} \end{bmatrix} \begin{bmatrix} 1 \\ \delta x \\ \delta\alpha \\ \delta u \end{bmatrix}$$

Thus, we have:

$$\delta u = -H_{uu}^{-1} H_u - H_{uu}^{-1} H_{ux} \delta x - H_{uu}^{-1} H_{u\alpha} \delta\alpha$$

when it converge, $H_u = 0$ and:

$$u_x^* = -H_{uu}^{-1} H_{ux}$$

and

$$u_\alpha^* = -H_{uu}^{-1} H_{u\alpha}$$

As you can see, α has similar coefficient as the sate variable. We can actually take α as a state variable and then solve it in the same way: define $v = [x^\top, \alpha^\top]^\top$, the dynamics: LaTeX

$$\dot{v} = \begin{bmatrix} f(v, u, t) \\ 0 \end{bmatrix} \tag{23}$$

Discrete DDP

Discrete time optimal control is to find a control sequence that:

$$u(0, \dots, N-1) = \arg\min_u \phi(x_N) + \sum_{i=k}^{N-1} a\big(L(x_i, u_i, i) + V(x_{i+1}, i+1)\big)$$

s.t.

$$x_{k+1} = F(x_k, u_k, k)$$
$$x(0) = x_0$$

Define an optimal cost-to-go function (value function) as:

$$V(x_k, k) = \min_u \sum_{i=k}^{N-1} \big(L(x_i, u_i, i) + V(x_{i+1}, i+1)\big) + \phi(x(N))$$

and a Q function:

$$Q(x_k, u_k, k) = L(x_k, u_k, k) + V(x_{k+1}, k+1)$$

The optimal control at time k is given by:

$$u^*(k) = \arg\min_{u(k)} Q(x_k, u_k, k)$$

Second order expansion:

$$Q(x_k + \delta x, u_k + \delta u) = \frac{1}{2} \begin{bmatrix} 1 \\ \delta x_k \\ \delta u_k \end{bmatrix}^\top \begin{bmatrix} 2Q & Q_x & Q_u \\ Q_x^\top & Q_{xx} & Q_{xu} \\ Q_u^\top & Q_{ux} & Q_{uu} \end{bmatrix} \begin{bmatrix} 1 \\ \delta x_k \\ \delta u_k \end{bmatrix} \tag{24}$$

The Newton step of u at time k is given by:

$$\delta u = -Q_{uu}^{-1}Q_u^\top - Q_{uu}^{-1}Q_{ux}\delta x$$

For Q approximation:

$$\begin{aligned}
Q_x(k) &= L_x(k) + V_x(k+1)F_x(k) \\
Q_u(k) &= L_u(k) + V_x(k+1)F_u(k) \\
Q_{xx}(k) &= L_{xx}(k) + F_x(k)^\top V_{xx}(k+1)F_x(k) + V_x(k+1)F_{xx}(k) \\
Q_{xu}(k) &= L_{xu}(k) + F_u(k)^\top V_{xx}(k+1)F_x(k) + V_x(k+1)F_{xu}(k) \\
Q_{ux}(k) &= L_{ux}(k) + F_x(k)^\top V_{xx}(k+1)F_u(k) + V_x(k+1)F_{ux}(k) \\
Q_{uu}(k) &= L_{uu}(k) + F_u(k)^\top V_{xx}(k+1)F_u(k) + V_x(k+1)F_{uu}(k)
\end{aligned}$$

The value function (optimal cost to go) is the Q function when the control is optimal. Therefore, the value function approximation is given by:

$$\begin{aligned}
V(k) &= Q(k) - Q_u Q_{uu}^{-1} Q_u^\top \\
V_x(k) &= Q_x(k) - Q_u(k)Q_{uu}^{-1}(k)Q_{ux}(k) \\
V_{xx}(k) &= Q_{xx}(k) - Q_{xu}(k)Q_{uu}^{-1}(k)Q_{ux}(k)
\end{aligned}$$

The terminal conditions are:

$$V_x(N) = \phi_x(N)$$

$$V_{xx}(N) = \phi_{xx}(N)$$

iLQR

The iLQR is a special form of DDP that it uses first order approximation of the system dynamics.

Problem definition and notations

Problem:

$$u(0, \ldots, N-1) = \arg\min_u \sum_{k=0}^{N-1} L(x_k, u_k, k) + \phi(x_N)$$

s.t.

$$x_{k+1} = F(x_k, u_k, k) \tag{25}$$

The first order approximation of the dynamics is given by:

$$x(k+1) = F_x\delta x(k) + F_u\delta u(k) + F(x_k, u_k, k)$$

Define a vector as $v := [1\ \delta x^\top\ \delta u^\top]^\top$.
The second order expansion of the cost function:

$$L(x, u, k) := v^\top \begin{bmatrix} L & L_x^\top & L_u^\top \\ L_x & L_{xx} & L_{xu} \\ L_u & L_{ux} & L_{uu} \end{bmatrix} v \tag{26}$$

Define $Z(x(k), u(k), k) := L(x(k), u(k), k) + V(x(k+1), k+1)$, which is often called Q function. The second order approximation of the Q function is given by:

$$Z(x_k, u_k, k) = \frac{1}{2}\begin{bmatrix} 1 \\ \delta x \\ \delta u \end{bmatrix}^\top \begin{bmatrix} 2Z & Z_x^\top & Z_u^\top \\ Z_x & Z_{xx} & Z_{xu} \\ Z_u & Z_{ux} & Z_{uu} \end{bmatrix} \begin{bmatrix} 1 \\ \delta x \\ \delta u \end{bmatrix} \tag{27}$$

For the shake of simplicity, let us denote the hessian matrix as:

$$\left[\begin{array}{cc|c} 2Z & Z_x^\top & Z_u^\top \\ Z_x & Z_{xx} & Z_{xu} \\ \hline Z_u & Z_{ux} & Z_{uu} \end{array}\right] = \left[\begin{array}{c|c} Z_{11} & Z_{12} \\ \hline Z_{21} & Z_{22} \end{array}\right] \tag{28}$$

where $Z_{11} \in R^{(1+N)\times(1+N)}$, $Z_{12} \in R^{(1+N)\times M}$, $Z_{21} \in R^{M\times(1+N)}$ and $Z_{22} \in R^{M\times M}$

Then the optimal control taken a Newton step is given by:

$$u^*(k) = u(k) - \alpha k - K\delta x(k)$$

where

$$[k|K] = -Z_{22}^{-1}(k)Z_{21}(k) \in R^{M\times(1+N)}$$

The value function is the Q function when the control is optimal. Its second order approximation is thus given by[4]:

$$V(x, k) = \begin{bmatrix} 1 \\ \delta x \end{bmatrix}^\top (Z_{11}(k) - Z_{12}(k)Z_{22}^{-1}(k)Z_{21}(k)) \begin{bmatrix} 1 \\ \delta x \end{bmatrix} \tag{29}$$

[4]This is not a 'real' value function since we haven't found the optimal trajectory yet. However, this is the best one that we can get.

To make it symmetric,

$$V = 0.5(V^\top + V)$$

For convenience, let's define a dynamics Jacobian as:

$$D := \begin{bmatrix} 1 & 0 & 0 \\ 0 & F_x^\top & F_u^\top \end{bmatrix} \tag{30}$$

The the chain rule for Q function update is given by:

$$Z(k) = L + D^\top V(k+1) D \tag{31}$$

where

$$V(N) = \phi(x_N, N)$$
$$V_x(N) = \phi_x(x_N, N)$$
$$V_{xx}(N) = \phi_{xx}(x_N, N)$$

The algorithm

The iLQR is outlined as follows:

1. For a given x_0, pick a control history $u(0, \ldots, N-1)$
2. Simulate forwards in time to get the state history, $x(0, \ldots, N)$, according to the system dynamics of Eq. 25
3. Simulate backwards in time to update the Q history, $Z(N, \ldots, 0)$, according to Eqs. 29, 30 and 31.
4. if the initial state is open, $\delta x(0) = V_{xx}^{-1}(0) V_x(0)$. Otherwise, $\delta x(0) = 0$. recorder the current cost as J^0 and its maximum possible decrease as:
$$\Delta J = V(0) - J^0$$
5. early stop:
   - if $\Delta J > 0$, return ERROR.
   - else if $\Delta J > -abs\_tols$ or $|\Delta J / J^0| < rel\_tol$, return SUCCESS
   - if exceed iteration number, return WARNING
6. Backtracking line search[5]:
   (a) Start with $\alpha = 1.0$ and iteration number i as 1
   (b) Simulate forwards in time to get the state and control history.
   $$x(0) \leftarrow x(0) - \alpha \delta x(0)$$
   $$u(k) \leftarrow u(k) - \alpha k - K \delta x(k)$$
   (c) Evaluate the resultant trajectory and denote its cost as J^i.
   (d) if $J^i < J^0 - \alpha * c * |\Delta J|$, where $c \in (0, 0.5)$ is the Armijo factor (typically $c = 0.25$), then return SUCCESS. Otherwise, $\alpha \leftarrow \tau\alpha$, where $\tau \in (0, 1)$ is the backtracking parameter, and repeat b). If $i > i_{max}$ or α becomes zero, return ERROR.

Constraints

Considering the following equality constraints:

$$c(x, k) = 0$$

We can convert the constrained optimization problem to unconstrained problem using augmented Lagrange method:

$$Z(x, u, k) = L(x, u, k) + V(F(x, u, k), k+1) + \frac{1}{2\rho} ||\rho c(x, k) + \lambda(k)||$$

where ρ is a constant scalar.

After linear search, the Lagrange multiplier is update according to the rule:

$$\lambda(k) \leftarrow \lambda(k) + \rho c(x, k)$$

For inequality constraints:

$$c(x, k) \leq U$$
$$c(x, k) \geq L$$

We can solve it by defining a new function:

$$h(x, k) = \max(c(x, k) - U, 0) + \min(c(x, k) - L, 0)$$

and then solve an equality constrained problem:

$$h(x, k) = 0$$

[5]: Refer to `https://en.wikipedia.org/wiki/Backtracking_line_search`

Constraint handling in iLQR

Lagrange cost:

$0.5*(min(0.0, \lambda+\mu^t op(c(x)-L))+max(0.0, lambda+\mu^t op(C(x)-U)))^2-\lambda^2/\mu$

update Lagrange

$$\lambda = min(0, \lambda + \mu^t op(c(x) - L) + max(0, \lambda + \mu^t op(c(x) - U))$$

Continuous LQR (Linear Quadratic Regulator)

Finite horizon LQR

For finite horizon LQR, we are solving:

$$u(t) = \arg\min_u x_f^\top Q(t_f) x_f + \int_{t_0}^{t_f} \frac{1}{2} x^\top(t) Q(t) x(t) + \frac{1}{2} u^\top(t) R(t) u(t)$$

s.t.

$$\dot{x}(t) = A(t)x(t) + B(t)u(t)$$

The Hamiltonian:

$$H = \frac{1}{2} x^\top Q x + \frac{1}{2} u^\top R u + \lambda^\top (Ax + Bu)$$

The optimal control is given by:

$$H_u = 0$$

then we have:

$$u^* = -R^{-1} B^\top \lambda$$

For the co-state:

$$\dot{\lambda} = -H_x^\top = -Qx - A^\top \lambda$$

Using the optimal control, the system dynamics with co-state can be written as:

$$\frac{d}{dt} \begin{bmatrix} x \\ \lambda \end{bmatrix} = \begin{bmatrix} A(t) & -B(t)R^{-1}(t)B^\top(t) \\ -Q(t) & -A^\top(t) \end{bmatrix} \begin{bmatrix} x \\ \lambda \end{bmatrix} \tag{32}$$

Note:

- Eq. 32 is called Hamiltonian DE.
- The $2n \times 2n$ matrix is called Hamiltonian for the problem.
- with conditions: $x(0) = x_0$ and $\lambda(t_f) = Q_f x(t_f)$, the problem is a Two-Point Boundary Value problem.

Since $\lambda(t_f) = Q_f x(t_f)$, let's try a connection $\lambda = P(t)x(t)$, we have:

$$-\dot{P} = P(t)A(t) + A(t)^\top P(t) + Q(t) - P(t)B(t)R^{-1}(t)B(t)^\top P(t) \tag{33}$$

s.t.

$$P(t_f) = Q_f$$

Note:

- Eq. 33 is Riccati differential equation.
- It can be derived by Eq. 19.
- we can integrate backwards in time to get $P(t)$ and then get optimal control:

$$u^* = -R^{-1}(t)B^\top(t)P(t)x(t)$$

Infinite horizon LQR

For finite horizon LQR, we are solving:

$$\min_u \int_0^\infty \frac{1}{2} x^\top(t) Q x(t) + \frac{1}{2} u^\top(t) R u(t)$$

s.t.

$$\dot{x}(t) = Ax(t) + Bu(t)$$

P is found by solving the continuous time algebraic Riccati equation:

$$0 = PA + A^\top P + Q - PBR^{-1}B^\top P$$

The optimal control is given by:

$$u = -R^{-1}B^\top P x$$

Discrete LQR

For discrete time LQR, we are interested in the following problem:

$$u(0,\dots,N-1) = \arg\min_u \sum_{k=0}^{N-1} \frac{1}{2} x_k^\top Q_k x_k + \frac{1}{2} u_k^\top R_k u_k + x_N^\top P_N x_N$$

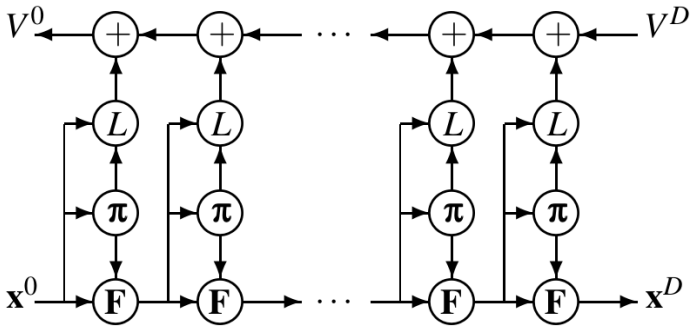
s.t.

$$x_{k+1} = A_k x_k + B_k u_k$$
$$x_0 = x(0)$$

We can derive discrete time LQR using the results of discrete time DDP, where $L_x(k) = x^\top Q$, $L_u = u^\top R$, $L_{xx} = Q$, $Luu = Q$, $F_x = A$, $F_u = B$.

Backpropagate as a network

The following diagram best describes the nature of the optimal control problem, which is a Two-point Boundary Value Problem, where the initial state and the final co-state are defined [6].



Note: There is a connection between π and v(k+1), which is not shown in the original paper, because:

$$\pi(x_k) = \arg\min_{u_k} L(x_k, u_k, k) + V(x_{k+1}, k+1)$$

[6] Efficient robust policy optimization, American Control Conference (ACC), 2012, Christopher G. Atkeson, Robotics Institute, Carnegie Mellon University, 5000 Forbes Avenue, Pittsburgh, PA, USA

Numerical Integration

The model ODE

The first order ODE:

$$\frac{dy}{dt} = f(t, y)$$

where $y \in R^N$. It is interesting because high order ODEs can be written in form of a system of first order ODEs.

Euler method

- forward Euler:

$$y_{k+1} = y_k + \Delta T f(y(k))$$

- backward Euler:

$$y_{k+1} = y_k + \Delta T f(y(k+1))$$

The forward Euler is also called explicit method because you can evaluate the next time step by an explicit equation. It is simple but less stable.

The backward Euler is also called implicit method because you have to solve nonlinear equations to get the next step. It is more complicate by more stable [7].

Runge-Kutta methods

The Runge-Katta methods are the most popular methods of solving ODE numerically. They can be derived for any order of accuracy. The most popular method is the fourth order Runge-Kutta method, or RK4 method,

[7] `https://web.stanford.edu/~fringer/teaching/numerical_methods_02/handouts/lecture7.pdf`

which is given by

$$\begin{aligned} y_{k+1} &= y_k + \frac{1}{6}(k_1 + 2K_2 + 2K_3 + k_4), && (34)\\ k_1 &= hf(t_k, y_k) && (35)\\ k_2 &= hf(t_k + \frac{h}{2}, y_k + \frac{1}{2}k_1) && (36)\\ K_3 &= hf(t_k + \frac{h}{2}, y_k + \frac{1}{2}k_2) && (37)\\ k_4 &= hf(t_k + h, y_k + k_3) && (38) \end{aligned}$$

Quadrature

Quadrature is more or less a a synonym for numerical integration, especially as applied to one-dimensional integrals. Some authors refer to numerical integration over more than one dimension as cubature; others take quadrature to include higher-dimensional integration.

$$\int_a^b f(y)dt \approx \sum_{i=1}^{N} w_i f(t_i)$$

where w_i are the quadrature weights and t_i are the quadrature points or nodes.

An interpolatory quadrature formula can be created for arbitrary support points by approximating the integrand by Lagrange polynomials, so that

$$\int_a^b f(y)dt \approx \int_a^b \sum_{i=1}^{N} L_i(t) \cdot f(t_i)dt$$

The quadrature weights can be easily determined as

$$w_i = \int_a^b L_i(t)dt$$

The quadrature formula with the maximum degree of precision is the **Gauss quadrature** formula, which is exact for polynomials of degree $2N-1$ or less. The Gauss formula is found by choosing the weights w_i and points t_i which make the formula exact for the highest degree polynomial possible. The points and weights are determined so that

$$\int_{-1}^{1} f(t)dt = \sum_{i=1}^{N} w_i \cdot f(t_i) + E_N$$

and the error E_N is zero for a polynomial of degree $2N-1$. The Gauss points are determined as the zeros of the N^{th} degree Legendre polynomial and the weights are the integrals of the resulting Lagrange interpolating polynomials, so that

$$w_i = \int_{-1}^{1} \prod_{k=1,k\neq i}^{N} \frac{t - t_k}{t_i - t_k}dt,\ i = 1, \ldots, N$$

There are other types of quadrature formula, such LGL, which is similar to the Gauss formula, except the boundary points are fixed at -1 and 1.

The error in the pesudospectral integration at the i^{th} point is bounded by

$$\|E_i\| \leq \|\frac{d^N f(\xi)}{dt^N}\| \frac{2}{N!},\ \xi \in [-1, 1]$$

and the pseudospectral integral will converge for any $f(t)$ whose derivatives are bounded, as the number of nodes used approaches infinity [8].

Solve ODEs backwards in time

Having a dynamics equation, it seems normal to integrate forward to get the time history. Backward ODE sounds odd because how can you integrate a dynamic system backward in time. But actually, it is still normal in math. We can still use Runge-Kutta to integrate ODEs backward in time using a negative time step!

$$\begin{aligned} y_{k-1} &= y_k + \frac{1}{6}(k_1 + 2K_2 + 2K_3 + k_4), && (39)\\ k_1 &= hf(t_k, y_k) && (40)\\ k_2 &= hf(t_k - \frac{h}{2}, y_k - \frac{1}{2}k_1) && (41)\\ K_3 &= hf(t_k - \frac{h}{2}, y_k - \frac{1}{2}k_2) && (42)\\ k_4 &= hf(t_k - h, y_k - k_3) && (43) \end{aligned}$$

Code

[8] A Gauss Pseudospectral Transcription for Optimal Control by David Benson, MIT, 2005

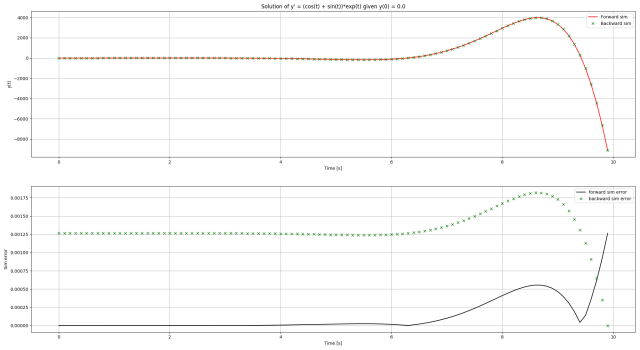


Figure 1: Comparison between solving ODEs forwards and backwards in time

Lagrange interpolation

Given N arbitrary support points of the function $f(t)$, on interval $t \in [a, b]$, there exists a unique polynomial P(t), of degree N-1 so that

$$P(t_i) = f_i, i = 1, \ldots, N$$

The unique polynomial can be found using Lagrange interpolation formula so that

$$P(t) = \sum_{i=1}^{N} f_i L_i(t)$$

where $L_i(t)$ are the Lagrange interpolation polynomials

$$L_i(t) = \prod_{k=1, k\neq i}^{N} \frac{t - t_k}{t_i - t_k} dt, \ i = 1, \ldots, N$$

Spectral method for ODE and PDE

- Galerkin, tau method
- Pseudospectral method, or collocation method satisfy boundary constraints and collocation points

Direct transcription

Take the state and the control at collocation points as optimization variables

Euler method

Runge-Kutta method

Lagrange Pseudospectral method

it is outlined as:

- **Lagrange interpolation + Gauss quadrature**
- the state and control are interpolated using Lagrange interpolation at N LGL points
- the dynamics constraints are enforced at the LGL points
- boundary constraints are enforced using the boundary points of approximating polynomial
- the integration in the cost function is discretized using GL quadrature rule

problems:

- the cost is evaluated only at the collocation points, which in general is sparse. It may miss some important details in the cost functions. **more details here**

Gauss Pseudospectral method

outlined as:

- convert to Bolza formulation use Lagrange interpolation + Gauss quadrature
- integration approximation matrix