Optimal Control and Reinforcement Learning

July 23, 2017

Contents

Preface

This notes is about optimal control and trajectory optimization. Most of it's content are cited from books and websites, I put them together in a more readable manner for myself and for all who are passionate about it..

I will use the following notations unless specified otherwise:

- $\mathbf{x} \in \mathbb{R}^n$: the vector of state variables
- $\mathbf{u} \in \mathbb{R}^m$: the vector of control variables
- $J(\mathbf{x}, \mathbf{u}, t_f)$: the objective function or performance index
- $L(\mathbf{x}, \mathbf{u})$: for discrete-time system, called one step cost function; for continuous-time system, called Lagrangian.
- $V(\mathbf{x}) \in \mathbb{R}$: the value function or the optimal cost function, or the optimal return function. $V(\mathbf{x}, t) = \min_{\mathbf{u}(\cdot)} J(\mathbf{x}, \mathbf{u}, t)$
- $R(\mathbf{x}, \mathbf{u})$: the reward function

Introduction

Optimal control theory, an extension of the calculus of variations, is a mathematical optimization method for deriving control policies. The method is largely due to the work of Lev Pontryagin and his collaborators in the Soviet Union and Richard Bellman in the United States.

Optimal control deals with the problem of finding a control law for a given system such that a certain optimality criterion is achieved. A control problem includes a cost functional (or called performance index) that is a function of state $\mathbf{x} \in \mathbb{R}^n$ and control $\mathbf{u} \in \mathbb{R}^m$ and/or the terminal time t_f. Minimize the continuous-time cost functional or discrete cost summation.

Three important elements:

1. dynamics model
2. cost (reward) function

3. policy or control law

For optimal control problem, in general 1) and 2) are known, the problem is to find an optimal 3); For Reinforcement Learning problem, 1) and 2) are often unknown, they are measured from the physical world or simulation world, the goal is the same, to find the optimal 3).

Problem formulation (continuous time)

Bolza Formulation

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

s.t.

$$\dot{\mathbf{x}} = f(\mathbf{x}, \mathbf{u}, t), \tag{2}$$

and the boundary conditions

$$\mathbf{x}(t_0) = \mathbf{x}_0, \quad \psi(\mathbf{x}_f, t_f) = 0 \tag{3}$$

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 sate vector is extended by one state $x_{n+1}(t)$.

$$x_{n+1}(t) = \int_{t_0}^{t} L(\mathbf{x}, \mathbf{u}, t)dt. \tag{4}$$

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

$$J(u(\cdot), t_f) = \phi(\mathbf{x}(t_f), t_f) \tag{5}$$

s.t.

$$\dot{\mathbf{x}} = f(\mathbf{x}, \mathbf{u}, t), \tag{6}$$

and the boundary conditions

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

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

in particular, there are constraints on the state of the form

$$S(\mathbf{x}(t)) \leq 0 \tag{7}$$

and on the control variable

$$C(\mathbf{x}(t), \mathbf{u}(t)) \leq 0 \tag{8}$$

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

Hamilton-Jacobi-Bellman equation (HJB) and Pontryagin Maximum Principle

The Hamilton-Jacobi-Bellman (HJB) equation is a partial differential equation which is central to optimal control theory. The solution of the HJB equation is the *value function*, which gives the optimal cost-to-go for a given dynamical system with an associated cost function. The solution is open loop, but it also permits the solution of the closed loop problem. Classical variational problems, for example, the brachistochrone problem can be solved using this method. The HJB method can be generalized to stochastic systems as well.

The equation is a result of the theory of dynamic programming which was pioneered in the 1950s by Richard Bellman and coworkers. The corresponding discrete-time equation is usually referred to as the *Bellman equation*. In continuous time, the result can be seen as an extension of earlier work in classical physics on the Hamilton-Jacobi equation by William Rowan Hamilton and Carl Gustav Jacob Jacobi.

Intuitively, HJB can be derived as follows:

$$V(\mathbf{x}(t),t) = \min_{\mathbf{u}}\{L(\mathbf{x}(t),\mathbf{u}(t),t)\mathrm{d}t + V(\mathbf{x}(t+\mathrm{d}t),t+\mathrm{d}t)\} \tag{9}$$

The Taylor expansion of the last term is:

$$V(\mathbf{x}(t+\mathrm{d}t),t+\mathrm{d}t) = V(\mathbf{x},t) + V_t\mathrm{d}t + V_{\mathbf{x}}\dot{\mathbf{x}}\mathrm{d}t + o(\mathrm{d}t^2) \tag{10}$$

where $o(\mathrm{d}t^2)$ denotes the terms in the Taylor expansion of higher order than one. Then if we cancel $V(\mathbf{x}(t),t)$ on both sides, divide by $\mathrm{d}t$, and take the limit as $\mathrm{d}t$ approaches zero, then we obtain the HJB equation as:

$$V_t(\mathbf{x},t) + \min_{\mathbf{u}}\{V_{\mathbf{x}}f(\mathbf{x},\mathbf{u}) + L(\mathbf{x},\mathbf{u})\} = 0 \tag{11}$$

The HJB equation is a **necessary** and **sufficient** condition for an optimum. If we can solve for V then we can find from it a control u that achieves the minimum cost.

Eq. 11 can be written as

$$-V_t(\mathbf{x},t) = \min_{\mathbf{u}} H(\mathbf{x},\mathbf{u},t,V_{\mathbf{x}}) \tag{12}$$

$$H := L(\mathbf{x},\mathbf{u}) + V_{\mathbf{x}}f(\mathbf{x},\mathbf{u}) \tag{13}$$

It shows that the optimal control $\mathbf{u}^*$ is the value of $\mathbf{u}$ that globally minimizes the Hamiltonian $H(\mathbf{x},\mathbf{u},t,V_{\mathbf{x}})$, holding $\mathbf{x}$, $V_{\mathbf{x}}$, and t constant. It was formulated by the Russian mathematician Lev Semenovich Pontryagin and his students and known as *Pontryagin's minimum principle*.

One of the most effective way to solve HJB, which is a first-order nonlinear partial differential equation, is the method of characteristic. It amounts to find a field of extremals. We can use Euler-Lagrange Equations, which is ordinary differential equation (ODE), to solve a particular optimal path and its associate optimal function, thus get the field of extremals.

Euler-Lagrange Equation

The *Euler-Lagrange equation* was developed in the 1750s by Euler and Lagrange in connection with their studies of the *tautochrone* problem. This is the problem

of determining a curve on which a weighted particle will fall to a fixed point in a fixed amount of time, independent of the starting point. Lagrange solved this problem in 1755 and sent the solution to Euler. The two further developed *Lagrange's method* and applied it to mechanics, which led to the formulation of *Lagrangian mechanics*. Their correspondence ultimately led to *the calculus of variations*, a term coined by Euler himself in 1766.

In *calculus of variations*, the *Euler-Lagrange equation*, or *Lagrange's equation*, is a differential equation whose solutions are the functions for which a given functional is stationary. Because a differentiable functional is stationary at its local maxima and minima, the *Euler-Lagrange equation* is useful for solving optimization problems in which, given some functional, one seeks the function minimizing (or maximizing) it.

The *Euler-Lagrange equation* is an equation satisfied by a function q of a real argument t which is a stationary point of the functional

$$S(q) = \int_a^b L(t, q, \dot{q}) dt \tag{14}$$

The *Euler-Lagrange equation*, then, is the ordinary differential equation

$$\frac{\partial L}{\partial q} - \frac{d}{dt}\frac{\partial L}{\partial \dot{q}} = 0 \tag{15}$$

The solution of this eq. 15 is the stationary solution of eq. 14[fn:prove].

Short proof:

$$\frac{\delta S}{\delta q} = 0 \tag{16}$$

$$\begin{aligned} \delta S(q(t)) &= \int \frac{\partial L}{\partial q}\delta q + \frac{\partial L}{\partial \dot{q}}\delta \dot{q} \mathrm{d}t & (17) \\ &= \frac{\partial L}{\partial \dot{q}}\delta q(t)\Big|_{t_0}^{t_f} + \int \Big(\frac{\partial L}{\partial q} - \frac{d}{dt}\frac{\partial L}{\partial \dot{q}}\Big)\delta q \mathrm{d}t & (18) \end{aligned}$$

The boundary conditions, $\delta q(t_0) = 0$ and $\delta q(t_f) = 0$, causes the first term to vanish

In physics, *Hamilton's principle* is William Rowan Hamilton's formulation of the principle of stationary action. It states that the dynamics of a physical system is determined by a variational problem for a functional based on a single function, the Lagrangian, which contains all physical information concerning the system and the forces acting on it. The evolution $q(t)$ of a system described by $q(t) \in \mathbb{R}^N$ between $(t_1, q(t_1))$ and $(t_2, q(t_2))$ is an extremum of the action functional

$$S(q) := \int_{t_1}^{t_2} L(t, q, \dot{q}) dt \tag{19}$$

where $L(t, q, \dot{q}) := K - V$ is the Lagrangian function for the system.
A good tutorial can be found at:

- `http://www.eftaylor.com/software/ActionApplets/LeastAction.html`

In Optimal Control, the scalar Hamiltonian function H is defined as

$$H[\mathbf{x}(t), \mathbf{u}(t), \lambda(t), t] := L(\mathbf{x}, \mathbf{u}, t) + \lambda^T(t) f(\mathbf{x}, \mathbf{u}, t), \tag{20}$$

and we choose the multiplier function $\lambda(t)$ to be $\dot{\lambda}^T = -\frac{\partial H}{\partial x} = -L_x - \lambda^T f_x$, which is also called co-state or adjoint variables, then the continuous-time EL equation:

$$\dot{x} = \frac{\partial H}{\partial \lambda^T} = f(x, u, t) \tag{21}$$

$$\dot{\lambda}^T = -\frac{\partial H}{\partial x} = -L_x - \lambda^T f_x \tag{22}$$

$$H_u = L_u + \lambda^T f_u 0, \quad t_0 \leq t \leq t_f \tag{23}$$

with boundary conditions

$$x(t_0) \quad \text{if specified,} \tag{24}$$

$$\lambda^T(t_f) = \phi_x(t_f) \tag{25}$$

The evolution of λ is given *reverse time*, from a final state to the initial. Hence we see the primary difficulty of solving optimal control problems: the state propagates forward in time, while the costate propogates backward. The state and costate are coordinated through the above equations.

Lagrange Equations

Lagrangian mechanics is a re-formulation of classical mechanics that combines conservation of momentum with conservation of energy. It was introduced by Italian mathematician Joseph-Louis Lagrange in 1788. In Lagrangian mechanics, the trajectory of a system of particles is derived by solving the Lagrange equations in one of two forms, named the Lagrange equations of the first kind, which treat constraints explicitly as extra equations, often using Lagrange multipliers; and the Lagrange equations of the second kind, which incorporate the constraints directly by judicious choice of generalized coordinates. The fundamental lemma of the calculus of variations shows that solving the Lagrange equations is equivalent to finding the path for which the action functional is stationary, a quantity that is the integral of the Lagrangian over time.

Define the following scalar function

$$L = K(\dot{q}) - V(q), \tag{26}$$

where K is kinetic energy and is a function of only $\dot{q}$, V is the potential energy and is a function of only q, q is called generalized coordinate. Lagrange equations are often written as

$$\frac{d}{dt}\frac{\partial L}{\partial \dot{q}} - \frac{\partial L}{\partial q} = 0. \tag{27}$$

Intuitive example:

$$L = \frac{1}{2}mv^2 - mgh \tag{28}$$

where $v := \dot{q}$ and $h := z$,the generalized coordinate is $q := (x, y, z)'$. The first term of 27

$$\frac{d}{dt}\frac{\partial L}{\partial \dot{q}} = m\ddot{q}. \tag{29}$$

The second term is

$$\frac{\partial L}{\partial q} = (0, 0, mg)' \tag{30}$$

According to Newton's second law, $m\ddot{q} - (0, 0, mg) = 0$.

Hamilton Equations

Hamiltonian mechanics is a reformulation of classical mechanics that was introduced in 1833 by Irish mathematician William Rowan Hamilton. It arose from Lagrangian mechanics, a previous reformulation of classical mechanics introduced by Joseph Louis Lagrange in 1788, but can be formulated without recourse to Lagrangian mechanics using symplectic spaces (see Mathematical formalism, below). The Hamiltonian method differs from the Lagrangian method in that instead of expressing second-order differential constraints on an n-dimensional coordinate space (where n is the number of degrees of freedom of the system), it expresses first-order constraints on a 2n-dimensional phase space.

$$\begin{aligned} \dot{p} &= -\frac{\partial H}{\partial q} \\ \dot{q} &= \frac{\partial H}{\partial p} \end{aligned} \tag{31}$$

where $p(t)$ is generalized momenta, $q(t)$ is generalized coordinates, H is the Hamilton function, or Hamiltonian. It is simply the total energy of the material point. Intuitive Example:

$$H = \frac{p^2}{2m} + V(q) \tag{32}$$

with the variable defined as:

- $p = m\dot{q}$: the momentum of the material point
- $q = (x, y, z)'$: the location of the material point
- $V(q)$: the potential energy

Recall that $F = \dot{p} = -\frac{\partial V}{\partial q}$, then Eq. 31 is true.
Using Hamilton's equations:

- Write out the Lagrangian $L = T - V$. Express T and V as though you were going to use Lagrange's equation.
- Calculate the momenta by differentiating the Lagrangian with respect to velocity, as $p = \frac{\partial L}{\partial \dot{q}}$
- $H = p\dot{q} - L$.

Introduction

Generally speaking, optimal control problems are nonlinear and, thus do not have analytic solutions (e.g., like the linear-quadratic optimal control problem). As a result, it is necessary to employ numerical methods to solve optimal control problems. In the early years of optimal control (circa 1950s to 1980s) the favored approach for solving optimal control problems was that of indirect methods. In an indirect method, the calculus of variations is employed to obtain the first-order optimality conditions. These conditions result in a two-point (or, in the case of a complex problem, a multi-point) boundary-value problem. This boundary-value problem actually has a special structure because it arises from taking the derivative of an Hamiltonian. Thus, the resulting dynamical system is an Hamiltonian system of the form

$$\dot{x} = \frac{\partial H}{\partial \lambda^T} \tag{33}$$

$$\dot{\lambda}^T = -\frac{\partial H}{\partial x} \tag{34}$$

where

$$H = L(x, u, t) + \lambda^T f(x, u, t), \tag{35}$$

is the augmented Hamiltonian and in an indirect method, the boundary-value problem is solved (using the appropriate boundary or transversality conditions). The beauty of using an indirect method is that the state and adjoint state, λ, are solved for and the resulting solution is readily verified to be an extremal trajectory. The disadvantage of indirect methods is that the boundary-value problem is often extremely difficult to solve (particularly for problems that span large time intervals or problems with interior point constraints).

In direct methods, optimal control problems are transform into nonlinear programming problems. By taking the sequence of control as decision variables and using explicit numerical integration, optimal control problems become parametric optimization problem and can be solved by so-call direct shooting method [**?**]. The shooting methods require forward simulation of the dynamics to compute the cost function (and its gradients), and therefore can be fairly expensive to evaluation. Moreover, it needs good starting trajectories. In direct collocation [**?**, **?**], the state and/or control variables are approximated using an appropriate function approximation (e.g., polynomial approximation or piecewise constant parameterization). Then, the coefficients of the function approximations are treated as optimization variables and the problem becomes an nonlinear programming problem (NLP), which can be solved by general nonlinear programming method, such as sequential quadratic programming (SQP). Because of sparse characteristic, the NLP is easier to solve than the boundary-value problem. Optimization software package, SNOPT, can be used to solve such large sparse NLPs [**?**]. % As a result, the range of problems that can be solved via direct % methods (particularly direct collocation methods which are very popular these days) is % significantly larger than the range of problems that can be % solved via indirect methods %[**?**, **?**, **?**, **?**]. Compared with indirect methods, this approach is found more robust in term of finding solution when little knowledge is known about the final result. As a result, the range of problems that can be solved via direct methods is larger than the range of problems

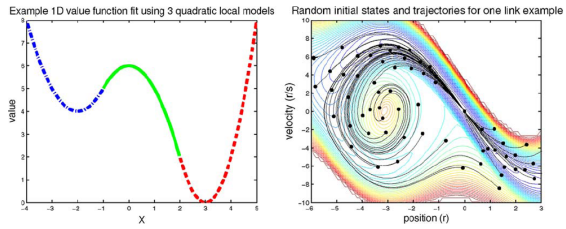


Figure 1: (Left) example of a local approximation of a 1-D value function using three quadratic models. (Right) (dots) random states used to plan one-link swingup superimposed on a contour map of the value function. (Black lines) optimized trajectories are shown starting from the random states.

that can be solved via indirect methods [**?**, **?**, **?**, **?**]. But one disadvantage is that they produce less accurate and sometimes physically incorrect results. Another disadvantage is local minima [**?**]. In order to improve the low accuracy of direct method results and to increase the convergence areas of indirect methods, combination methods were proposed in [**?**, **?**].

Dynamic Programming

A typical algorithm, such as Dynamic Programming (DP), often discretizes the states and controls into cells, refines a candidate optimal cost function and/or a corresponding control policy function iteratively [**?**].

Rangom Sampling of State in Dynamic Programming, which combine sparse random sampling of states with local models and local optimization, can solve some harder problem [**?**].

Direct Method

The approach that has risen to prominence in numerical optimal control over the past two decades (i.e., from the 1980s to the present) is that of so called direct methods. In a direct method, the state and/or control are approximated using an appropriate function approximation (e.g., polynomial approximation). Simultaneously, the cost functional is approximated as a cost function. Then, the coefficients of the function approximations are treated as optimization variables and the problem is "transcribed" to a nonlinear optimization problem.

Direct (Multiple) Shooting Method

In numerical analysis, direct shooting method is the method for solving a boundary value problem by reducing it to the solution of an initial value problem.

Intuitively speaking, we want to shoot some target, $p(t_f) = (x_f, y_f)$, from our current position, $p(0) = (x_0, y_0)$. it is a two-point boundary problem. We can solve the orbit of the bullet, $p(t)$, if we can find a appropriate initial shooting velocity $\dot{p}(0)$. So we convert this boundary value problem into the problem of finding the initial velocity problem. This is why it is called 'shooting method'.

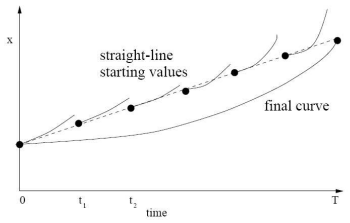


Figure 2: Multiple Shooting Method

The prescribed boundary conditions for the problem are initial values of the state $\mathbf{x}(0)$ and final values of the costate $\lambda^T(t_f)$. The traditional shooting method integrates the state equations beginning at $\mathbf{x}(0)$ and from a guess for $\lambda^T(t_0)$ for a pre-specified time t_f. When the numerical value obtained for $\lambda^T(t_f)$ from the integration does not match the boundary condition $\lambda^T(t_f) = \phi_x(t_f)$, a zero-finding problem may be set up in terms of the constraints, and Newton's method may then be used to iterate on $\lambda^T(t_0)$ until the boundary condition is satisfied.

Multiple Shooting Method: This procedure consists of dividing up the time interval $[0, t_f]$ into a series of grid points $[0, t_1, t_2, \ldots, t_f]$, and a shooting method is performed between successive grid points (see Figure 2). A set of starting values for the state and the co-state is required at each grid point in time, and continuity conditions for the solution trajectory introduce additional interior boundary conditions which are then incorporated into one large zero-finding problem to be solved.

Advantages and disadvantages:

- The strong sensitivity of initial value problems, and the small convergence

region for the Newton's method make solving these problems via simple shooting very difficult.

- Shooting methods require forward simulation of the dynamics to compute

the cost function (and its gradients), and therefore can be fairly expensive to evaluation.

- Additionally, they do not make very effective

use of the capabilities of modern nonlinear optimization routine (like SNOPT) to enforce constrains.

Direct Collocation Method

In direct collocation [**?**, **?**], the state and/or control variables are approximated using an appropriate function approximation (e.g., polynomial approximation or piecewise constant parameterization). Then, the coefficients of the function approximations are treated as optimization variables and the problem becomes an nonlinear programming problem (NLP), which can be solved by general nonlinear programming method, such as sequential quadratic programming (SQP).

Because of sparse characteristic, the NLP is easier to solve than the boundary-value problem. Optimization software package, SNOPT, can be used to solve such large sparse NLPs [**?**]. Compared with indirect methods, this approach is found more robust in term of finding solution when little knowledge is known about the final result. As a result, the range of problems that can be solved via direct methods is larger than the range of problems that can be solved via indirect methods [**?**, **?**, **?**, **?**]. But one disadvantage is that they produce less accurate and sometimes physically incorrect results. Another disadvantage is local minima [**?**].

Midpoint Discretization Method:
The system dynamics in state space is

$$\dot{\mathbf{x}} := \frac{d}{\mathrm{dt}} \begin{bmatrix} p(t) \\ v(t) \end{bmatrix} = \begin{bmatrix} v(t) \\ acc(p(t), v(t), \mathbf{u}(t)) \end{bmatrix} \tag{36}$$

The discrete approximations to the velocity and the acceleration at each interval are as follows:

$$v(i) = \frac{p(i+1) - p(i-1)}{2\mathrm{dT}} \quad i \in [1, N-1] \tag{37}$$

$$a(i) = \frac{p(i+1) + p(i-1) - 2p(i)}{\mathrm{dT}^2}. \tag{38}$$

which are called *midpoint discretization*. The approximation to the end velocity is

$$v(N) = v(N-1) + a(N-1)\mathrm{dT}. \tag{39}$$

$$\mathbf{x}(i) := (p^T(i), v^T(i))^T \quad i \in [1, N] \tag{40}$$

The performance index is then

$$J = \phi(\mathbf{x}(N)) + \sum_{i=1}^{N-1} L(\mathbf{x}(i), \mathbf{u}(i))\mathrm{dT} \tag{41}$$

s.t.

$$a(i) = acc(\mathbf{x}(i), \mathbf{u}(i)) \tag{42}$$

$$p(1) = p_0 \tag{43}$$

$$v(1) = v_0 \tag{44}$$

$$p(N) = p_f \tag{45}$$

$$v(N) = v_f \tag{46}$$

$$lb_q \leq p(j) \leq ub_q \quad j \in [0, N] \tag{47}$$

$$lb_v \leq v(k) \leq ub_v \quad k \in [1, N] \tag{48}$$

$$lb_u \leq u(i) \leq ub_u \quad i = [1, N-1] \tag{49}$$

Parameters to be optimized are $[p(0), p(1), \ldots, p(N), u(1), u(1), \ldots, u(N-1)]$.

Piece-wise Constant Integration Method: This method over-parameterizes the system with both $\mathbf{x}$ and $\mathbf{u}$ are explicit decision variables as $[\mathbf{x}(0), \mathbf{x}(1), \ldots, \mathbf{x}(N), \mathbf{u}(0), \mathbf{u}(1), \ldots, \mathbf{u}(N-1)]$. The control are chosen as piece-wise constant between $\mathbf{u}(k)$ and $\mathbf{u}(k+1)$. Besides constraints on state and constraints on control, the discrete time dynamics equations are also imposed as constraints on the state variables.

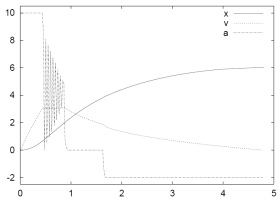
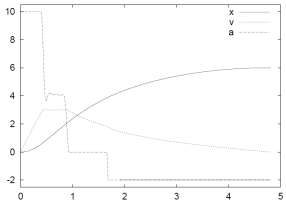
[Ringing]  [Smoothing] 

Figure 3: Results of Midpoint Discretization Method.

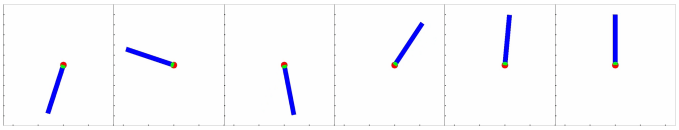


Figure 4: One-link pendulum swing-up problem. The frames are taken in intervals of 0.5 second.

"Ringing" Problem: As shown in Fig. , this method sometimes has the problem of "Ringing" [**?**].

Smoothing. The model can be improved by adding to the objective some penalty on the changes of the control variables, such as first-order smoothness $k\sum_1^{N-2}\left(\mathbf{u}(i+1)-\mathbf{u}(i)\right)^2$ and second-order smoothness as $k\sum_2^{N-2}\left(\mathbf{u}(i+1)+\mathbf{u}(i-1)-2\mathbf{u}(i)\right)^2$

Direct methods are very popular today. The update is fast, and it is very easy to implement constrains. A less obvious attribute of these methods is that they may be easier to initialize with trajectories that are in the vicinity of the desired minima. The most commonly stated limitation of these methods is their accuracy.

Illustration: One Pendulum Swing-up

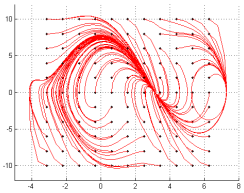
We use one-link pendulum swingup as an intuitive example. In one-link pendulum swingup, a motor at the base of the pendulum swings a rigid arm from the downward stable equilibrium to the upright unstable equilibrium and balances the arm there. What makes this challenging is that the one-step cost function penalizes the amount of torque used and the deviation of the current position from the goal. The controller must try to minimize the total cost of the trajectory. The performance index as

$$J=\sum_{i=0}^{N-1}L(p(i),u(i))\mathrm{dT} \tag{50}$$

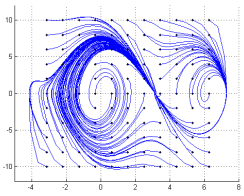
where $L=0.1(p(i)-\pi)^2+u(i)^2$, p is the position, u is the applied torque, dT is the discretization time. The optimization results are shown in 5.

{In order to try to avoid local minima, information should be exchanged between trajectories to enable convergence to globally optimal trajectories.}

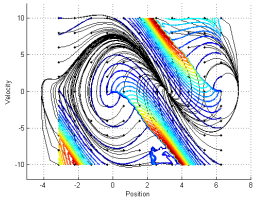
Some explain goes here ...



[Local minimum results] [Global minimum results]



[Optimal trajectories over the contour of value function of local minimum.]



[Optimal trajectories over the contour of value function of global minimum.]

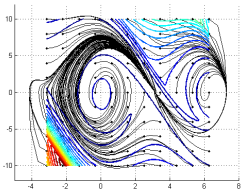


Figure 5: Trajectory optimization of one-link pendulum swing-up problem.

```
for level = 0 to MaxCost, step do
    if level <= Cost(T(i,j)) < level + step then
        Mark (i,j);
        N++;
    end
    while N > 0 do
        forall the (i,j)'s neighbors (i',j') do
            Re-optimize trajectory T(i',j') by using trajectory T(i,j) as
            initial guess;
            if Cost(T(i',j')) is reduced then
                Mark (i',j');
                N++;
            end
        end
        Un-mark (i,j);
        N--;
    end
end
```

Algorithm 1: Trajectory optimization algorithm

Indirect Method

The Gradient Method

- Newton's method
- gradient descent method
- conjugate method
- steep descent
- multiple shooting method

EL summery:

$$\dot{\mathbf{x}} = f(\mathbf{x}, \mathbf{u}) \tag{51}$$

$$\dot{\lambda}^T = = -L_x - \lambda^T f_x \tag{52}$$

$$L_u + \lambda^T f_u = 0 \tag{53}$$

$$\lambda^T(t_f) = \phi_x(\mathbf{x}(t_f)) \tag{54}$$

The gradient method is quite popular in applications and for complex problems may be the only way to effectively develop a solution.

- For a given $\mathbf{x}_0$, pick a control history $\mathbf{u}(t)$.
- Propagate $\dot{\mathbf{x}} = f(\mathbf{x}, \mathbf{u}, t)$ forward in time to create a state trajectory.
- Evaluate $\phi(\mathbf{x}(tf))$, and the propagate costate λ backward in time from t_f to t_0, using Eq. 34.

- At each time step, choose $\mathbf{u} = -K(L_u + \lambda^T f_u)$, where K is a positive scalar or a positive definite matrix in the case of multiple input channels.
- Let $\mathbf{u} = \mathbf{u} + \delta\mathbf{u}$.
- Go back to step 2 and repeat loop until solution has converged.

The major difficulties are computational expense, and the requirement of having a reasonable control trajectory to begin.

The Newton-Raphson Method

The indirect method we use is called Difference Dynamic Programming (DDP), which is a second order local trajectory optimization method. In stead of minimizing a cost function and calculate its gradients directly, it take advantage of the sequential forms of the discrete-time dynamic equation (51), and a local second-order expansion of the optimum cost function:

$$V(\mathbf{x}) \approx V_0 + V_x\delta\mathbf{x} + \frac{1}{2}\delta\mathbf{x}^T V_{xx}\delta\mathbf{x} \tag{55}$$

where $\delta\mathbf{x} = \mathbf{x} - \mathbf{x}^j$, a local second-order expansion of the dynamics:

$$F(\mathbf{x},\mathbf{u}) \approx F_0 + F_x\delta\mathbf{x} + F_u\delta\mathbf{u} + \frac{1}{2}[\delta\mathbf{x}^T\ \delta\mathbf{u}^T]\begin{bmatrix}F_{xx} & F_{xu}\\ F_{ux} & F_{uu}\end{bmatrix}\begin{bmatrix}\delta\mathbf{x}\\ \delta\mathbf{u}\end{bmatrix} \tag{56}$$

where $\delta\mathbf{u} = \mathbf{u} - \mathbf{u}^j$, and a local second-order expansion of one step cost function:

$$L(\mathbf{x},\mathbf{u}) \approx L_0 + L_x\delta\mathbf{x} + L_u\delta\mathbf{u} + \frac{1}{2}[\delta\mathbf{x}^T\ \delta\mathbf{u}^T]\begin{bmatrix}L_{xx} & L_{xu}\\ L_{ux} & L_{uu}\end{bmatrix}\begin{bmatrix}\delta\mathbf{x}\\ \delta\mathbf{u}\end{bmatrix}. \tag{57}$$

Given a starting trajectory $\mathbf{x}^j$, one can integrate the first and second partial derivatives of the cost function backward in time by:

$$\begin{aligned}
Z_x &= V_xF_x + L_x;\ Z_u = V_xF_u + L_u && (58)\\
Z_{xx} &= F_x^T V_{xx}F_x + V_xF_{xx} + L_{xx} && (59)\\
Z_{xu} &= F_x^T V_{xx}F_u + V_xF_{xu} + L_{xu} && (60)\\
Z_{uu} &= F_u^T V_{xx}F_u + V_xF_{uu} + L_{uu} && (61)\\
V_{x(k-1)} &= Z_x - Z_uZ_{uu}^{-1}Z_{ux} && (62)\\
V_{xx(k-1)} &= Z_{xx} - Z_{xu}Z_{uu}^{-1}Z_{ux} && (63)
\end{aligned}$$

where the initial conditions are

$$\begin{aligned}
V_x &= \phi_x(N) && (64)\\
V_{xx} &= \phi_{xx}(N) && (65)
\end{aligned}$$

and ϕ_x and ϕ_{xx} are the first and second partial derivatives of terminal penalty function ϕ with respect to $\mathbf{x}(N)$. A new updated trajectory $\mathbf{x}^{j+1}(k)$ is generated by integrating Eq. (51) forward in time using the linear feedback law, i.e.,

$$\mathbf{u}^{j+1} = \mathbf{u}^j - Z_{uu}^{-1}Z_{ux}[\mathbf{x}^{j+1} - \mathbf{x}^j] - Z_{uu}^{-1}Z_u. \tag{66}$$

Repeating these processes until the convergence to optimal state trajectory $\mathbf{x}^o$ and optimal control trajectory $\mathbf{u}^o$ is achieved. Constraints on state and control and on control can be handled by adding soft penalty to the one step cost function.

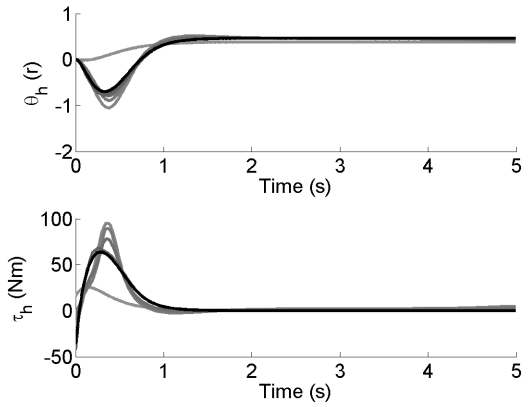


Figure 6: The convergence to the optimal trajectories.

Steady-state Local Models

A by-product of DDP is local models of optimal control law in the neighborhood of the optimal trajectory, $\mathbf{K}(k) \triangleq \mathbf{u}^o_{\mathbf{x}}$, where

$$\mathbf{u}(k) = \mathbf{u}^o(k) + \mathbf{K}(k)[\mathbf{x}(k) - \mathbf{x}^o(k)] \tag{67}$$

and $\mathbf{K}(k) = -Z^-_{uu}(k)Z_{ux}(k)$. For fixed-end time problem, we can calculate and store the optimal control and state vectors $\mathbf{u}^o(T)$ and $\mathbf{x}^o(T)$ and as the same time the optimal gain matrix $\mathbf{K}(T)$, where $T = N - k =$ step-to-go, then $\mathbf{u}(T)$ can be calculated by Eq. (67) [**?**]. One drawback is that this feedback control law is time-variant. In [**?**], a fixed-length Receding Horizon scheme was proposed to synthesize a time invariant controller from many local controllers. In the spirit of Model Predictive Control [**?**], we calculate time-invariant or steady-state local models of the optimal cost function and the control law by designing an appropriate terminal penalty term.

For a discrete-time dynamic system, the optimal cost function of infinite temporal horizon can be split up into two parts

$$V^\infty(\mathbf{x}_0) = \min_{\mathbf{u}(\cdot)} \Big(\sum_{k=0}^{N-1} L\big(\mathbf{x}(k), \mathbf{u}(k)\big) + \sum_{k=N}^{\infty} L\big(\mathbf{x}(k), \mathbf{u}(k)\big) \Big). \tag{68}$$

The goal is to approximate the second term on the right-hand side by a terminal penalty term, $\phi(\mathbf{x}(N))$. Without further restrictions, it is difficult for general nonlinear systems. However, if we ensure that the trajectories of the closed-loop system remain within the vicinity of a goal state (terminal region) for $k \in [N, \infty)$, then the upper bound approximation to the second term can be found. One solution is to determine the terminal region $\mathcal{R}$ such that a local state feedback law $\mathbf{u} = \pi(\mathbf{x})$ asymptotically stabilizes the nonlinear system and render $\mathcal{R}$ positively invariant under the closed-loop control. If $\mathbf{x}(N) \in \mathcal{R}$, then the second term of Eq. (68) can be upper bounded by the cost resulting from the application of this local controller $\mathbf{u} = \pi(\mathbf{x})$. Requiring that $\mathbf{x}(N) \in \mathcal{R}$ and

using the local controller $\mathbf{u} = \pi(\mathbf{x})$ for $k \in [N, \infty)$, we obtain:

$$V^{\infty}(\mathbf{x}_0) \approx \min_{\mathbf{u}(\cdot)} \Big(\sum_{k=0}^{N-1} L\big(\mathbf{x}(k), \mathbf{u}(k)\big) + \sum_{k=N}^{\infty} L\big(\mathbf{x}(k), \pi(\mathbf{x}(k))\big) \Big) \tag{69}$$

If one step cost function L is with the form of

$$L(\mathbf{x}, \mathbf{u}) = \frac{1}{2}(\mathbf{x} - \mathbf{x}_d)^T \mathbf{Q}(\mathbf{x} - \mathbf{x}_d) + \frac{1}{2}(\mathbf{u} - \mathbf{u}_d)^T \mathbf{R}(\mathbf{u} - \mathbf{u}_d), \tag{70}$$

a locally linear feedback law $\mathbf{u} = \mathbf{K}\mathbf{x}$ can be used and the terminal penalty term can be approximated with a quadratic form

$$\phi(\mathbf{x}(N)) = \frac{1}{2}(\mathbf{x}(N) - \mathbf{x}_d)^T \mathbf{P}_{ss}(\mathbf{x}(N) - \mathbf{x}_d), \tag{71}$$

$\mathbf{P}_{ss}$ is given by solving the following matrix Riccati equation,

$$\mathbf{P}_{ss}\mathbf{A} + \mathbf{A}^T\mathbf{P}_{ss} + \mathbf{Q} - \mathbf{P}_{ss}\mathbf{B}\mathbf{R}^{-1}\mathbf{B}^T\mathbf{P}_{ss} = 0, \tag{72}$$

where $\mathbf{A} = F_x$ and $\mathbf{B} = F_u$ are the partial derivatives of F with respect to desired state $\mathbf{x}_d$ and corresponding control $\mathbf{u}_d$. Substituting Eq. (71) into Eq. (69) we obtain

$$V^{\infty}(\mathbf{x}_0) \approx \min_{\mathbf{u}(\cdot)} \Big(\phi(\mathbf{x}(N)) + \sum_{k=0}^{N-1} L\big(\mathbf{x}(k), \mathbf{u}(k)\big) \Big). \tag{73}$$

Requiring $\mathbf{x}(N) \in \mathcal{R}$ and using Eq. (71) as the terminal penalty term in Eq. (1), the infinite temporal horizon optimal control can be upper approximated.

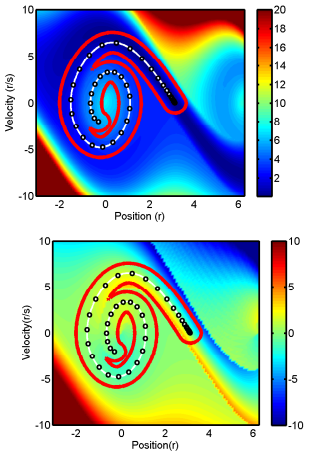
In general, only one optimal trajectory to the terminal manifold will pass through a given state $(\mathbf{x}, k)$, so that a unique optimal control vector $\mathbf{u}(\mathbf{x}, k)$ is associated with each state. Furthermore, by using Eq. (71) as the terminal penalty term and requiring $\mathbf{x}(t_f) \in \mathcal{R}$, we get steady-state linear models of the optimal control along the optimal trajectory x^o as

$$m_i(\mathbf{x}) := \mathbf{u}(i) + \mathbf{K}(i)(\mathbf{x} - \mathbf{x}(i)) \quad i \in [1, N] \tag{74}$$

The control law can then be modeled by means of these spatially localized linear models. A simple approach is using the closest local model to calculate control, $\mathbf{u}(\mathbf{x}) = m_i(\mathbf{x})$, where $||\mathbf{x}(i) - \mathbf{x}||_D \leq ||\mathbf{x}(j) - \mathbf{x}||_D$ for all $j \neq i$ and D is distance matrix.

One-link Pendulum Swingup Example Problem

Fig. 8 shows the optimal cost function, which indicates the minimum total cost starting from each state, and the optimal control policy, which indicates the optimal control for each state. Dynamic Programming requires storing the optimal control over an n-dimensional grid of the entire state space or in the neighborhood of the nominal path. The computation and even the storage of the optimal control becomes difficult when the dimension is high, which called "the curse of dimensionality". By contrast, if we have an optimal trajectory and local models of the optimal control policy along it, then we can get approximations to the optimal control in its neighborhood.

[The optimal cost function.] 

[The optimal control policy.]

Figure 7: The optimal cost function and the optimal control policy for a one-link pendulum swingup problem. One optimal trajectory is shown as a white line. Red lines in both plots show the boundary of its neighborhood. Black circles are states used to construct local models of the optimal cost function in the left plot and of the control policy in the right plot. The optimal cost function is cut off above 20. The goal is at the state (π, 0)

Introduction of Optimal Control

Infinite Time Horizon Optimal Control

$$\min J = \sum_{k=0}^{\infty} L(x_k, u_k)$$

subject to

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

$$x_0 = x(0)$$

$$\dots$$

- variables: state and control trajectories $x_0, x_1, \dots \in \mathrm{R}^N$, $u_0, u_1, \dots \in \mathbf{R}^M$

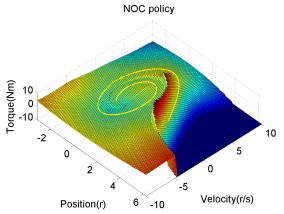
Solution via Dynamic Programming

- (Bellman) value function $V(x)$ is optimal value of control problem as a function of state x.

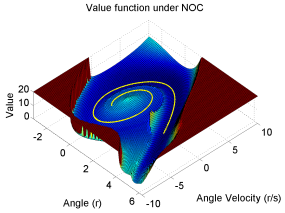
$$V(x_0) = \min \sum_{k=0}^{\infty} L(x_k, u_k)$$

- V satisfies Bellman or dynamic programming equation

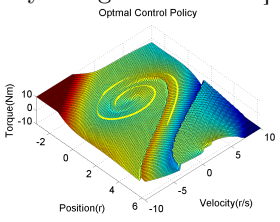
$$V(x_k) = \min_{u_k} \{L(x_k, u_k) + V(x_{k+1})\}$$



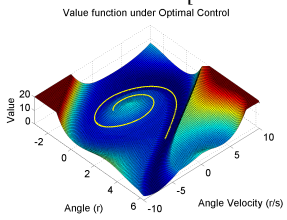
[Control policy using local models.]



[Value function of the control policy using local models]



[Optimal control policy]



[Value function of the optimal control policy]

Figure 8:

- Optimal control variable is given by

$$u^* = arg\min_{u_k}\{L(x,u) + V(F(x,u))\}$$

  NOTE: the optimal control has 'state feedback form' that $u_k = \pi(x_k)$

- For a time-invariant linear system, there is analytical solution for $V(x)$ (solution of Algebraic Riccati Equation)

Introduction of Model Predictive Control

Model Predictive Control (MPC)

$$\min J = \sum_{k=0}^{k=N} L(x_k, u_k)$$

subject to

$$x_{k+1} = F(x_k, u_k)$$
$$x_0 = x(0)$$
$$x \in X$$
$$u \in U$$
$$\dots$$

with variables $x_1, \dots, x_N, u_0, \dots, u_{N-1}$

- we take $u = u_0$
- this also gives a state feedback control $u_k = \pi_{mpc}(x_k)$

Variation on MPC

- Add final state cost $\hat{V}(x_f)$ such that

$$\min\{\sum_{k=0}^{k=N} L(x_k, u_k) + \hat{V}(x_{N+1})\}$$

- if $\hat{V} = V$, MPC gives optimal control
- Why?

  Recall Bellman's Principle of Optimality,

$$\min\sum_{k=0}^{\infty} L(x_k, u_k) = \min\{\sum_{k=0}^{N} L(x_k, u_k) + \sum_{k=N+1}^{\infty} L(x_k, u_k)\}$$

  then we have

$$\min\sum_{k=0}^{\infty} L(x_k, u_k) = \min\{\sum_{k=0}^{N} L(x_k, u_k) + V(x_f)\}$$

  where $x_f = x_{N+1}$

Variation on MPC (Continuation)

- what's the benefit?
  - red If we have a good approximation to V, MPC's time horizon can be reduced by a lot.

    NOTE: the computation benefit is at the cost of ignoring future dynamic changes.
- A MPC with a short time horizon can give almost the same result as a MPC with a long time horizon, which means we can save a lot of computation!

CHOMP (Co-variant Hamiltonian Optimization for Motion Planning)

Remarks

- Suited for gradient-cheap application, where gradient can be computed inexpensively
- Trajectory optimization is invariant to parameterization
  - trajectory: a mapping from time t to configuration q
  - objective function: a function of trajectory (functional)
- Compared with sampling-based method, optimal control, and trajectory optimization (DDP and iLQR)
- The meaning of the name: Co-variant gradient and Hamiltonian Monte Carlo

  `https://personalrobotics.ri.cmu.edu/files/courses/papers/CHOMP12-ijrr.pdf`

Problem formulation

$$\xi(t)^* = \arg\min_{\xi} U(\xi(t))$$

where $U(\xi(t)) = F_{smooth}(\xi(t)) + \lambda F_{obs}(\xi(t))$, and
The smoothness objective:

$$F_{smooth}(\xi(t)) = \frac{1}{2}\int_0^1 ||\frac{d}{dt}\xi(t)||^2 dt$$

The obstacle objective:

$$F_{obs}(\xi(t)) = \int_0^1 \int_B c(x(\xi(t),u))||\frac{d}{dt}x(\xi(t),u)||dudt$$

Functional gradient

- The functional gradient ∇U is the perturbation ϕ that maximizes $U[\xi+\epsilon\phi]$ as $\epsilon \to 0$. Choosing a traditional Euclidean norm $||\xi||^2 = \int \xi(t)^2 dt$, for any differential objective of form $F(\xi) = \int L(t, \xi, \xi')dt$, the Euclidean functional gradient:

$$\nabla F(\xi) = \frac{\partial L}{\partial \xi} - \frac{d}{dt}\frac{\partial L}{\partial \dot{\xi}}$$

- Euler-Lagrange equation

$$\frac{\partial L}{\partial q} - \frac{d}{dt}\frac{\partial L}{\partial \dot{q}} = 0$$

  Euler-Lagrange equation is the stationary condition for:

$$\min_{q(t)} S(q) = \int_a^b L(t, q, \dot{q})dt$$

Functional gradient

- The functional gradient of the smoothness objective:

$$\nabla F_{smooth} = -\ddot{\xi}$$

- The functional gradient of the obstacle objective

$$\nabla F_{obs} = \int_B J^T ||x'|| \big((I - \hat{x}'\hat{x}^T)\nabla c - ck\big) du$$

where $k = \frac{(I-\hat{x}'\hat{x}'^T)x''}{||x'||^2}$ is the curvature vector along the workspace trajectory.

The matrix $(I - \hat{x}'\hat{x}'^T)$ is a projection matrix that projects workspace gradients orthogonally to the trajectory's direction of motion.

Functionals and Co-variant Gradients for Trajectory Optimization

- To removing the dependence on the parametrization (finite difference), defining a norm on trajectory perturbations to depend solely on the dynamical quantities of the trajectory:

$$||\xi||_A := \int \sum_{n=1}^{k} \alpha_n (D^n \xi(t))^2 dt$$

  where D is differential operator, $A = D^T D$.

Invariant norm $||\xi(t)||_A :=< \xi, A\xi >= \int (D\xi(t))^2 dt$

- Functional gradients based on the invariant norm:

$$\bar{\nabla}_A U(\xi) := A^{-1}\bar{\nabla} U(\xi)$$

Waypoint Trajectory Parameterization

refer to the paper

Update rule:

$$\xi_{i+1} = \arg\min_{\xi} U(\xi_i) + (\xi - \xi_i)^T \nabla U(\xi_i) + \frac{\eta}{2}||\xi - \xi_i||_M$$

This update rule is covariant in the sense that the change to the trajectory that results from the update is a function only of the trajectory itself, and not the particular representation used (e.g. waypoint based), at least in the limit of small step size and fine discretization.

The overall CHOMP objective function is strongly convex – that is, it can be lower-bounded over the entire region by a quadratic with curvature A

$$\xi_{i+1} = \xi_i - \eta A^{-1} \nabla U(\xi_i)$$

Hamiltonian Monte Carlo

refer: `http://arogozhnikov.github.io/2016/12/19/markov_chain_monte_carlo.html`

- Intuition

  $$p(\xi, \alpha) \propto exp(-\alpha U(\xi))$$

- define a Hamilton function as

  $$H(\xi, \gamma) := U(\xi) + K(\gamma)$$

  where $K(\gamma) = \frac{1}{2}\gamma'\gamma$.

  $$p(\xi, \gamma) \propto exp(-H(\xi, \gamma))$$

  The system dynamics (Hamilton equation):

  $$\frac{d\xi}{dt} = \gamma$$

  $$\frac{\gamma}{dt} = -\nabla U(\xi)$$

  The full numerically robust Hamiltonian Monte Carlo. refer to the paper

Optimal Smoothing and Its Application

Optimal Smoothing Background

Optimal smoothing

To quantify the uncertainty associated with an unobserved variable, X, given some information pertaining to that variable, Y. In a sequential context, this relates to the calculation of the posterior density, $p(x_{1:T}|y_{1:T})$, where $x_{1:T} :=$

$x_1, \ldots, x_T$, $y_{1:T} := y_1, \ldots, y_T$, are the state to estimate and the observation processes, and the discrete time index $t \in \mathbb{N}^+$.

To facilitate sequential estimation, a first order Markovian state space model is often assumed and defined by its initial probability $X_1 \sim \mu(\cdot)$ and for $t > 0$:

$$X_t|(X_{t-1} = x_{t-1}) \sim f(\cdot|x_{t-1});$$

Furthermore, the observations are conditional upon the state and are assumed to be statistically independent and distributed according to:

$$Y_t|(X_t = x_t) \sim g(\cdot|x_t),$$

where we assume that $\mu(\cdot)$, $f(\cdot|x_{t-1})$ and $g(\cdot|x_t)$ are probability densities with respect to some dominating measure.

The joint posterior density $p(x_{1:T}|y_{1:T})$ is simply given by:

$$p(x_{1:t}|y_{1:t}) \propto \mu(x_1) \prod_{k=2}^{t} f(x_k|x_{k-1}) \prod_{k=1}^{t} g(y_k|x_k)$$

Given the empirical observations $y_{1:T}$, the optimal fix-interval smoother tries to find a sequence of the state variables that gives the maximum a posterior probability (MAP):

$$\max_{x_{1:T}} \mu(x_1) \prod_{k=2}^{t} f(x_k|x_{k-1}) \prod_{k=1}^{t} g(y_k|x_k) \tag{75}$$

subjective to

$$X_t|(X_{t-1} = x_{t-1}) \sim f(\cdot|x_{t-1});$$
$$Y_t|(X_t = x_t) \sim g(\cdot|x_t),$$
$$X_1 \sim \mu(\cdot)$$

Optimal linear quadratic smoother

The process model:

$$x_k = A_k x_{k-1} + w_{k-1}$$

The observation model:

$$y_k = H_k x_k + v_k$$

where w_k is the process noise which is assumed to be drawn from a zero mean multivariate normal distribution, $w_k \sim \mathcal{N}(0, Q_k)$, and v_k is the observation noise which is assumed to be zero mean Gaussian white noise with covariance R_k, $v_k \sim \mathcal{N}(0, R_k)$. The covariances of the two noise models are assumed stationary over time and are given by:

$$Q = E(w_k, w_k)$$

$$R = E(v_k, v_k)$$

Recall Eq. **??**, for the linear process model and observation model, we have:

$$\mu(x_1) = |2\pi P|^{-2} \exp(-(x_1 - \hat{x}_1)^\top P^{-1}(x_1 - \hat{x}_1))$$

$$f(x_k|x_{k-1}) = |2\pi Q|^{-2} \exp(-(x_k - A_{k-1}x_{k-1})^\top Q^{-1}(x_k - A_{k-1}x_{k-1}))$$
$$g(y_k|x_k) = |2\pi R|^{-2} \exp(-(y_k - H_k x_k)^\top R^{-1}(y_k - H_k x_k))$$

where $\hat{x}_1$ and P are the initial state's prior estimation and covariance.

Then, the fix-interval optimal smoother problem can be reformulated as:

$$\min_{x_{1:T}} \frac{1}{2} \sum_{k=1}^{N} v_k^\top R^{-1} v_k + \frac{1}{2} \sum_{k=2}^{T} w_k^\top Q^{-1} w_k + \frac{1}{2}(x_1 - \hat{x}_1)^\top P^{-1}(x_1 - \hat{x}_1) \qquad (76)$$

where

$$v_k = y_k - H_k x_k$$
$$w_k = x_k - A_{k-1}x_{k-1}$$

In practice, Eq. 20 can be solved by batch least square (BLS) estimation methods or sequential recursive estimation methods, such as Fraser and Potter (1969), Rauch, Tung, and Striebel (1965), and etc.