Static and Dynamic Optimization (42111)

Niels Kjølstad Poulsen

Build. 303b, room 016
Section for Dynamical Systems
Dept. of Applied Mathematics and Computer Science
The Technical University of Denmark

Email: nkpo@dtu.dk
phone: +45 4525 3356
mobile: +45 2890 3797

2017-10-30 11:21

Lecture 8: Free dynamic optimization (D+C)

Outline of lecture

- Recap L7
- Analytical solutions and Numerical methods
- Continuous time system
- Exercise DO.2
- Reading guidance (DO: 11-14, 27-34)

Dynamic Optimization (D)

Minimize J (ie. determine the sequence $u_i \in \mathbb{R}^m$, $i = 0, \ldots, N-1$) where:

$$J = \phi(x_N) + \sum_{i=0}^{N-1} L_i(x_i, u_i)$$

subject to (for $i = 0,\ 1,\ \ldots\ N-1$):

$$x_{i+1} = f_i(x_i, u_i) \qquad\qquad x_0 = \underline{x}_0$$

Given:

- N horizon (number of intervals)
- $\underline{x}_0 \in \mathbb{R}^n$ initial state is known
- f_i dynamics ($\mathbb{R}^{n+m+1} \to \mathbb{R}^n$)
- $L_i(x_i, u_i)$ stage cost
- $\phi(x_N)$ terminal cost

DTU

Euler-Lagrange equations

Defining the Hamiltonian function

$$H_i = L_i(x_i, u_i) + \lambda_{i+1}^T f_i(x_i, u_i)$$

The KKT conditions on this problem (with equality constraints) results in:

$$\boxed{\begin{aligned} x_{i+1} &= f_i &&= \left(\frac{\partial}{\partial \lambda_i} H_i\right)^T \\ \lambda_i^T &= \frac{\partial}{\partial x_i} H_i &&= \frac{\partial}{\partial x_i} L_i + \lambda_{i+1}^T \frac{\partial}{\partial x_i} f_i \\ 0^T &= \frac{\partial}{\partial u_i} H_i &&= \frac{\partial}{\partial u_i} L_i + \lambda_{i+1}^T \frac{\partial}{\partial u_i} f_i \end{aligned}}$$

with boundary conditions:

$$x_0 = \underline{x}_0 \qquad \lambda_N^T = \frac{\partial}{\partial x_N} \phi_N$$

where for short:

$$f_i = f_i(x_i, u_i) \qquad L_i = L_i(x_i, u_i) \qquad H_i = H_i(x_i, u_i, \lambda_{i+1})$$

- This is a two-point boundary value problem (TPBVP) with $N(2n+m)$ unknowns and equations.

- The quantity
$$\frac{\partial}{\partial u_i} H_i$$
is the gradient of J wrt. u_i. Equal to zero on optimal trajectories.

- λ_0 is the gradient of J wrt. x_0.

Type of solutions:

- Analytical solutions (for very simple problems)
- Semi analytical solutions (eg. the LQ problem)
- Numerical solutions

Let us now focus on the problem of bringing the linear, first order system given by:

$$x_{i+1} = ax_i + bu_i \qquad\qquad x_0 = \underline{x}_0$$

along a trajectory from the initial state, such the cost function (for chosen $p \geq 0$, $q \geq 0$ and $r > 0$):

$$J = \frac{1}{2}px_N^2 + \sum_{i=0}^{N-1} \frac{1}{2}qx_i^2 + \frac{1}{2}ru_i^2$$

is minimized. The Hamiltonian for this problem is

$$H_i = \frac{1}{2}qx_i^2 + \frac{1}{2}ru_i^2 + \lambda_{i+1}\left[ax_i + bu_i\right]$$

and the Euler-Lagrange equations are:

$$x_{i+1} = ax_i + bu_i \tag{1}$$

$$\lambda_i = qx_i + a\lambda_{i+1} \tag{2}$$

$$0 = ru_i + b\lambda_{i+1} \tag{3}$$

which has the two boundary conditions

$$x_0 = \underline{x}_0 \qquad \lambda_N = px_N$$

The stationarity conditions (3),

$$0 \quad = \quad r u_i + b\lambda_{i+1}$$

give us a sequence of decisions

$$u_i = -\frac{b}{r}\lambda_{i+1} \tag{4}$$

if the costate is known.

Inspired from the boundary condition we will postulate a relationship

$$\lambda_i = s_i x_i \tag{5}$$

Actually separation of the variables (i and x). If we insert the control law, (4), and the costate candidate, (5), in the state equation, (1), we find

$$x_{i+1} = a x_i - b\frac{b}{r} s_{i+1} x_{i+1}$$

or

$$x_{i+1} = \left[1 + \frac{b^2}{r} s_{i+1}\right]^{-1} a x_i$$

From the costate equation, (2), we have

$$s_i x_i = q x_i + a s_{i+1} x_{i+1} = \left[q + a s_{i+1}(1 + \frac{b^2}{r} s_{i+1})^{-1} a\right] x_i$$

which has to be fulfilled for any x_i. This is the case if s_i is given by the backwards recursion

$$s_i = a s_{i+1}(1 + \underbrace{\frac{b^2}{r} s_{i+1}}_{x})^{-1} a + q \qquad \frac{1}{1+x} = 1 - \frac{x}{1+x}$$

or

$$\boxed{s_i = q + s_{i+1} a^2 - \frac{(a b s_{i+1})^2}{r + b^2 s_{i+1}} \qquad s_N = p} \tag{6}$$

This can be solved (recursively and backwards).

With this solution (the sequence of s_i) we can determine the (sequence of) control actions

$$u_i = -\frac{b}{r}\lambda_{i+1} = -\frac{b}{r}s_{i+1}x_{i+1} = -\frac{b}{r}s_{i+1}(ax_i + bu_i)$$

or

$$u_i = -\frac{abs_{i+1}}{r + b^2 s_{i+1}}x_i \qquad = -K_i\ x_i$$

where

$$K_i = \frac{s_{i+1}ab}{r + s_{i+1}b^2}$$

For the costate we have:

$$\lambda_i = s_i x_i$$

which can be compared with (which can be proven)

$$J^* = \frac{1}{2}s_0 x_0^2$$

LQ problem - II

Linear Dynamics:

$$x_{i+1} = Ax + Bu \qquad x_0 = \underline{x}_0 \qquad x_i \in \mathbb{R}^n \qquad u_i \in \mathbb{R}^m$$

and a Quadratic objective function:

$$J = \frac{1}{2} x_N^T P x_N + \frac{1}{2} \sum_{i=0}^{N-1} \left(x_i^T Q x_i + u_i^T R u_i \right) \qquad R > 0$$

The matrices, Q, R and P, are symmetric and positive semidefinite.

The problem has the Hamiltonian:

$$H_i = \frac{1}{2} \left(x_i^T Q x_i + u_i^T R u_i \right) + \lambda_{i+1}^T \left(A x_i + B u_i \right)$$

---

and the Euler-Lagrange equations are (necessary conditions):

$$\begin{aligned}
\left(\frac{\partial}{\partial \lambda} H_i \right)^T &= & x_{i+1} &= Ax + Bu & x_0 &= \underline{x}_0 \\
\frac{\partial}{\partial x} H_i &= & \lambda_i^T &= x_i^T Q + \lambda_{i+1}^T A & \lambda_N^T &= x_N^T P \\
\frac{\partial}{\partial u} H_i &= & 0^T &= u_i^T R + \lambda_{i+1}^T B & &
\end{aligned}$$

The **solution** to the LQ problem is:

$$\boxed{u_i = -K_i \; x_i}$$

where the gain is given by

$$K_i = \left(B^T S_{i+1} B + R\right)^{-1} B^T S_{i+1} A$$

and S_i is a solution to the Riccati equation

$$S_i = Q + A^T S_{i+1} A - A^T S_{i+1} B \left[B^T S_{i+1} B + R\right]^{-1} B^T S_{i+1} A \qquad S_N = P$$

The matrix, S_i is a symmetric, positive semidefinite matrix.

Notice the Costate

$$\lambda_i = S_i x_i \qquad S_i \geq 0$$

which might be compared to:

$$J^\star = \frac{1}{2} x_0^T S_0 x_0$$

Riccati

- Count Jacopo Francesco Riccati
- Born: 28 May 1676, Venice
- Dead: 15 April 1754, Treviso
- University of Padua

Source: Wikipedia

Pause

Numerical methods

- Shooting methods (forward or backward)
- Gradient methods
- Brute force

(LQ problem in the simple version). Let us now focus on the problem of bringing the linear, first order system given by:

$$x_{i+1} = ax_i + bu_i \qquad\qquad x_0 = \underline{x}_0$$

along a trajectory from the initial state, such the cost function:

$$J = \frac{1}{2}px_N^2 + \sum_{i=0}^{N-1} \frac{1}{2}qx_i^2 + \frac{1}{2}ru_i^2$$

is minimized. The Euler-Lagrange equations are:

$$\boxed{\begin{aligned} x_{i+1} &= ax_i + bu_i && (7)\\ \lambda_i &= qx_i + a\lambda_{i+1} && (8)\\ 0 &= ru_i + b\lambda_{i+1} && (9) \end{aligned}}$$

which has the two boundary conditions

$$x_0 = \underline{x}_0 \qquad \lambda_N = px_N$$

Forward shooting

The stationarity condition (9)

$$0 = ru_i + b\lambda_{i+1}$$

gives us simply:

$$u_i = -\frac{b}{r}\lambda_{i+1}$$

We <u>can</u> (for $a \neq 0$) reverse the costate equation

$$\lambda_i = qx_i + a\lambda_{i+1}$$

into

$$\lambda_{i+1} = \frac{\lambda_i - qx_i}{a}$$

Starting with $\underline{x}_0$ and λ_0 (guessed value) we can for $i = 0,\ 1,\ \dots\ N-1$ iterate:

$$\lambda_{i+1} = \frac{\lambda_i - q x_i}{a}$$

$$u_i = -\frac{b}{r}\lambda_{i+1}$$

$$x_{i+1} = a x_i + b u_i$$

We end up with x_N and λ_N which (for correct λ_0) should fulfill

$$\lambda_N = p x_N \qquad\qquad \varepsilon_N = \lambda_N - p x_N = 0$$

Notice: a problem if $a << 1$ or $a >> 1$.

Contents of a file (parms.m) setting the parameters.

```
% Constants etc.

alf=0.05;
a=1+alf; b=-1;
x0=50000;
N=10;
q=alf^2; r=q; p=q;
```

The following code (fejlf.m) solves these recursions.

```
function err=fejlf(la0)

parms % set parameters a,b,p,q,r,x0

la=la0; x=x0;
for i=0:N-1,
 la=(la-q*x)/a;
 u=-b*la/r;
 x=a*x+b*u;
end
err=la-p*x;
```

$$\lambda_{i+1} = \frac{\lambda_i - q x_i}{a}$$

$$u_i = -\frac{b}{r}\lambda_{i+1}$$

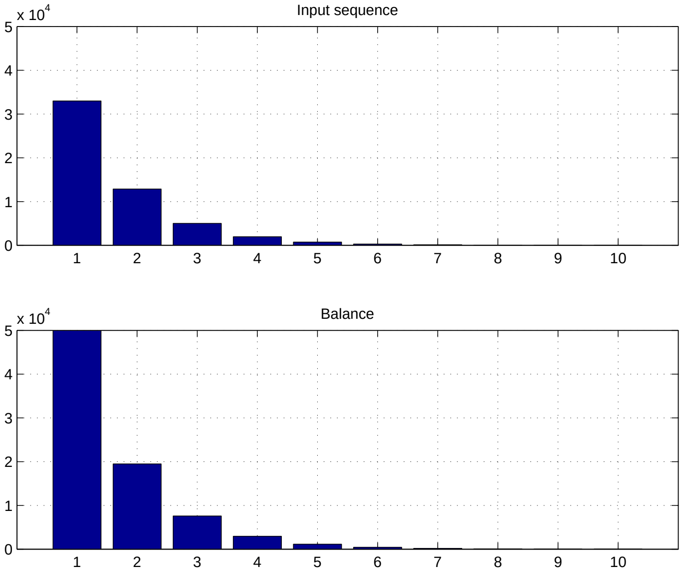
$$x_{i+1} = a x_i + b u_i$$

Extented version of fejlf.m (for plotting).

```
function [err,xt,ut,lat]=fejlf(la0)

parms % set parameters a,b,p,q,r,x0

la=la0; x=x0;
ut=[]; lat=la; xt=x;
for i=0:N-1,
 la=(la-q*x)/a;
 u=-b*la/r;
 x=a*x+b*u;
 xt=[xt;x]; lat=[lat;la]; ut=[ut;u];
end
err=la-p*x;
```

Master program (script).

```
% The search for la0
la0g=10; % a wild guess%
la0=fsolve('fejlf',la0g)

% The simulation with the correct la0
[err,xt,ut,lat]=fejlf(la0);

subplot(211); bar(ut); grid; title('Input sequence');
subplot(212); bar(xt); grid; title('Saldo');
```


Input sequence
x 10^4
5
4
3
2
1
0
1 2 3 4 5 6 7 8 9 10
Balance
x 10^4
5
4
3
2
1
0
1 2 3 4 5 6 7 8 9 10

Forward shooting method - Simplified

If separation possible: reverse the costate equation and find u_i from the stationarity condition.

The Euler-Lagrange equations

$$\begin{aligned}
x_{i+1} &= f_i(x_i, u_i) \\
\lambda_i^T &= \frac{\partial}{\partial x_i} L_i(x_i, u_i) + \lambda_{i+1}^T \frac{\partial}{\partial x_i} f_i(x_i, u_i) \qquad \rightarrow \qquad \lambda_{i+1} = h_i(x_i, \lambda_i) \\
0^T &= \frac{\partial}{\partial u_i} L_i(x_i, u_i) + \lambda_{i+1}^T \frac{\partial}{\partial u_i} f_i(x_i, u_i) \qquad \rightarrow \qquad u_i = g_i(x_i, \lambda_{i+1})
\end{aligned}$$

Guess λ_0 (<u>or another parameterization</u>) and use $\underline{x}_0$.

Iterate for $i = 0\ ,1\ \dots\ N-1$:

1. Knowing x_i and λ_i, determine u_i and λ_{i+1} from the stationarity and the costate equation.
2. Update the state equation i.e. find x_{i+1} from x_i and u_i.

At the end ($i = N$) check if

$$\lambda_N^T = \frac{\partial}{\partial x_N} \phi(x_N) \qquad \rightarrow \qquad \varepsilon = \lambda_N^T - \frac{\partial}{\partial x_N} \phi(x_N) = 0^T$$

Forward shooting method - General

The Euler-Lagrange equations

$$\begin{aligned} x_{i+1} &= f_i(x_i, u_i) \\ \lambda_i^T &= \frac{\partial}{\partial x_i} L_i(x_i, u_i) + \lambda_{i+1}^T \frac{\partial}{\partial x_i} f_i(x_i, u_i) \\ 0^T &= \frac{\partial}{\partial u_i} L_i(x_i, u_i) + \lambda_{i+1}^T \frac{\partial}{\partial u_i} f_i(x_i, u_i) \end{aligned} \qquad \begin{pmatrix} \lambda_{i+1} \\ u_i \end{pmatrix} = g_i \begin{pmatrix} x_i \\ \lambda_i \end{pmatrix}$$

Guess λ_0 (or another parameterization) and use $\underline{x}_0$.

Iterate for $i = 0\ ,1\ \ldots\ N-1$:

1. Knowing x_i and λ_i, determine u_i and λ_{i+1} from the stationarity and the costate equation.
2. Update the state equation i.e. find x_{i+1} from x_i and u_i.

At the end ($i = N$) check if

$$\lambda_N^T = \frac{\partial}{\partial x_N} \phi(x_N) \qquad \rightarrow \qquad \varepsilon = \lambda_N^T - \frac{\partial}{\partial x_N} \phi(x_N) = 0^T$$

Gradient methods

The Euler-Lagrange equations are:

$$\boxed{\begin{aligned} x_{i+1} &= f_i(x_i, u_i) \\ \lambda_i^T &= \frac{\partial}{\partial x_i} L_i(x_i, u_i) + \lambda_{i+1}^T \frac{\partial}{\partial x_i} f_i(x_i, u_i) && = \frac{\partial}{\partial x_i} H_i \\ 0^T &= \frac{\partial}{\partial u_i} L_i(x_i, u_i) + \lambda_{i+1}^T \frac{\partial}{\partial u_i} f_i(x_i, u_i) && = \frac{\partial}{\partial u_i} H_i \end{aligned}}$$

which has the two boundary conditions

$$x_0 = \underline{x}_0 \qquad \lambda_N^T = \frac{\partial}{\partial x_N} \phi(x_N)$$

Guess a sequence of decisions, $u_i \;\; i = 0,\; 1,\; \dots\; N-1$.

Search for an optimal sequence of decisions, $u_i \;\; i = 0,\; 1,\; \dots\; N-1$ using e.g. a Newton Raphson iteration:

$$u_i^{j+1} = u_i^j - \left[\frac{\partial^2}{\partial u_i^2} H_i \right]^{-1} \frac{\partial}{\partial u_i} H_i$$

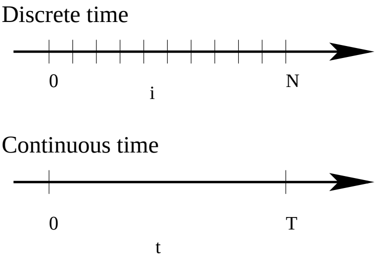
Brute force

Guess a sequence of decisions, $u_i \quad i = 0,\ 1,\ \ldots\ N-1$.

1. Start in x_0 and iterate the state equation forwards i.e. determine the state sequence, x_i.
2. Determine the performance index.

Search (e.g. using an amoeba method) for an optimal sequence of decisions, $u_i \quad i = 0,\ 1,\ \ldots\ N-1$.

Pause



The **Schaefer model** (Fish in the Baltics)

$$x_{i+1} = x_i + rhx_i(1 - \alpha x_i) \qquad x_0 = \underline{u}_0$$

h is the length of the intervals. The model can in continuous time be given as:

$$\boxed{\dot{x}_t = \frac{dx_t}{dt} = rx_t(1 - \alpha x_t) \qquad x_0 = \underline{x}_0}$$

The **fox(F) and rabbit(r)** example.

$$\begin{bmatrix} \dot{r} \\ \dot{F} \end{bmatrix} = \begin{bmatrix} \alpha_1 r - \beta_1 r F \\ -\alpha_2 F + \beta_2 r F \end{bmatrix} \qquad \begin{bmatrix} r \\ F \end{bmatrix}_0 = \begin{bmatrix} \underline{r}_0 \\ \underline{F}_0 \end{bmatrix}$$

In general Dynamic (continuous time) state space model:

$$\begin{bmatrix} \dot{x}_1 \\ \dot{x}_2 \\ \vdots \\ \dot{x}_n \end{bmatrix} = f_t\left(\begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix}_t , \begin{bmatrix} u_1 \\ \vdots \\ u_m \end{bmatrix}_t \right) \qquad \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix}_0 = \begin{bmatrix} \underline{x}_1 \\ \underline{x}_2 \\ \vdots \\ \underline{x}_n \end{bmatrix}_0$$

or in short:

$$\dot{x}_t = f_t(x_t, u_t) \qquad x_0 = \underline{x}_0 \qquad f : \mathbb{R}^{n+m+1} \rightarrow \mathbb{R}^n$$

The function, f, should be sufficiently smooth (existence and uniqueness).

Solution to ODE

- Analytical methods
- Numerical methods

$$\dot{x} = f_t(x_t) \qquad x_0 = \underline{x}_0$$

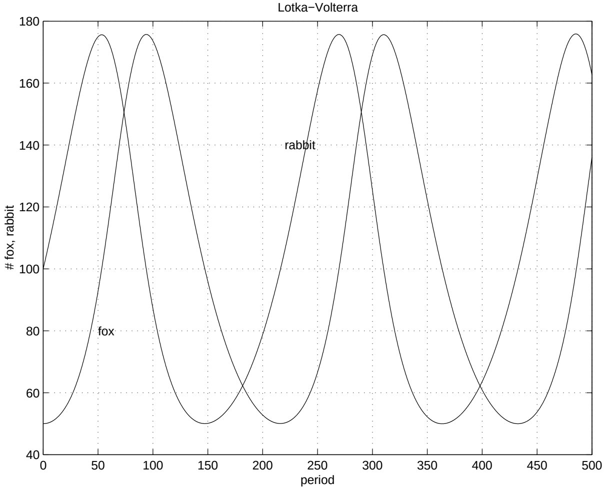
Euler integration (the most simple method)

$$x_{t+h} = x_t + h f_t(x_t)$$

is the most simple method. Others and more efficient numerical methods do exists.

The **fox(F) and rabbit(r)** example.

$$\begin{bmatrix} \dot{r} \\ \dot{F} \end{bmatrix} = \begin{bmatrix} \alpha_1 r - \beta_1 r F \\ -\alpha_2 F + \beta_2 r F \end{bmatrix} - \begin{bmatrix} u_r \\ u_f \end{bmatrix} \qquad \begin{bmatrix} r \\ F \end{bmatrix}_0 = \begin{bmatrix} \underline{r}_0 \\ \underline{F}_0 \end{bmatrix}$$



```
%----------------------------------------------------------------------
function dx=dfoxc(t,x,u,a1,a2,b1,b2)
%----------------------------------------------------------------------
% Dynamic function for the continuous Lotka-Volterra system.
% It determine the state derivative as function of the time, state vector
% and system parameters.

r=x(1);                 % # of Rabbits is the first state
F=x(2);                 % # of Foxes is the second state

dx=[ a1*r-b1*r*F;       % dx: derivative of x
    -a2*F+b2*r*F ]-u;
```

$$\begin{bmatrix} \dot{r} \\ \dot{F} \end{bmatrix} = \begin{bmatrix} \alpha_1 r - \beta_1 r F \\ -\alpha_2 F + \beta_2 r F \end{bmatrix} - \begin{bmatrix} u_r \\ u_f \end{bmatrix} \qquad \begin{bmatrix} r \\ F \end{bmatrix}_0 = \begin{bmatrix} \underline{r}_0 \\ \underline{F}_0 \end{bmatrix}$$

DTU

```
%------------------------------------------------------------------
function foxc
%------------------------------------------------------------------
% This program simulates the trajectories for the Lotka-Volterra system.
%------------------------------------------------------------------

a1=0.03; a2=0.03;       % System parameters (enters in dfoxc function)
b1=0.03/100; b2=b1;

r=100;                  % Initial # of rabbits
f=50;                   % Initial # of foxes
x0=[r;f];               % Initial state value

Tstp=500;               % Stop time
dT=0.1;                 % Step size in output data
Tspan=0:dT:Tstp;        % Time span of which the solution is to be found
%Tspan=[0 Tstp];        % Alternative time span

u=[0.0;0.0];            % Shootings of rabbits and foxes

[time,xt]=ode45(@dfoxc,Tspan,x0,[],u,a1,a2,b1,b2);      % ODE solver
                                                        % See dfox for dynamic
                                                        % function
```

```
%---------------------------------------------------------------------
% The rest of this program (until next function declaration) is just
% plot and plot related commands

plot(time,xt); grid;
title('Lotka-Volterra');
ylabel('# fox, rabbit');
xlabel('period');
text(50,80,'fox');
text(220,140,'rabbit');
% print foxc.pps       % Just a printing command

%
% end of main program
%---------------------------------------------------------------------
```

Analytical solutions

Discrete time

$$x_{i+1} = x_i \qquad x_i = C$$

$$x_{i+1} = x_i + \alpha \qquad x_i = C + \alpha i$$

$$x_{i+1} = a x_i \qquad x_i = C a^i$$

$$x_{i+1} = A x_i + B u_i \qquad x_0 = \underline{x}_0$$

$$x_i = A^i \underline{x}_0 + \sum_{j=0}^{i} A^{n-j-1} B u_j$$

Continuous time

$$\dot{x} = 0 \qquad x = C$$

$$\dot{x} = \alpha \qquad x_t = C + \alpha t$$

$$\dot{x} = a x \qquad x_t = C exp(at)$$

$$\dot{x} = A x + B u \qquad x_0 = \underline{x}_0$$

$$x_t = e^{At} \underline{x}_0 + \int_0^t e^{A(t-s)} B u_s ds$$

Constant as C can be determined from boundary conditions. Examples are

$x_0 = \underline{x}_0$ or $x_N = \underline{x}_N$

The Performance Index

In discrete time we search for a sequence of decisions ($u_i \quad i = 0,\ 1,\ \dots\ N-1$) such that the performance index

$$J = \phi(x_N) + \sum_{i=0}^{N-1} L_i(x_i, u_i)\ h$$

is optimized s.t. the dynamics (and other possible constraints).

In continuous time we search for a decision function ($u_t \quad 0 \le t \le T$) such that the performance index

$$J = \phi_T(x_T) + \int_0^T L_t(x_t, u_t)\ dt$$

is optimized s.t. the dynamics (and other possible constraints).

Free dynamic optimization: Minimize J (ie. determine the function u_t, $0 \leq t \leq T$) where:

$$J = \phi_T(x_T) + \int_0^T L_t(x_t, u_t)\, dt \qquad \text{Objective}$$

subject to

$$\dot{x} = f_t(x_t, u_t) \qquad x_0 = \underline{x}_0 \qquad \text{Dynamics}$$

- T is given
- $\underline{x}_0$ is given
- $f_t(x_t, u_t)$ dynamics
- $L_t(x_t, u_t)$ kernel or running cost
- $\phi_T(x_T)$ terminal loss and x_T is free.

$$\boxed{\begin{aligned}
\dot{x}_t &= f_t(x_t, u_t) \\
-\dot{\lambda}_t^T &= \frac{\partial}{\partial x_t} L_t(x_t, u_t) + \lambda_t^T \frac{\partial}{\partial x_t} f_t(x_t, u_t) \\
0^T &= \frac{\partial}{\partial u_t} L_t(x_t, u_t) + \lambda_t^T \frac{\partial}{\partial u_t} f_t(x_t, u_t)
\end{aligned}}$$

with boundary conditions:

$$x_0 = \underline{x}_0 \qquad \lambda_T^T = \frac{\partial}{\partial x} \phi_T(x_T)$$

Hamilton function

Define the Hamilton function as:

$$H(x, u, \lambda, t) = L_t(x_t, u_t) + \lambda_t^T f_t(x_t, u_t)$$

Then the Euler-Lagrange equations (KKT conditions) for this problem can be written as:

$$\boxed{\dot{x}^T = \frac{\partial}{\partial \lambda} H_t \qquad -\dot{\lambda}^T = \frac{\partial}{\partial x} H_t \qquad 0^T = \frac{\partial}{\partial u} H_t}$$

$\frac{\partial}{\partial u} H$ is the gradient of J wrt. u.

λ_0^T is the gradient of J wrt. x_0.

The first equation is just the state equation

$$\dot{x} = f_t(x_t, u_t)$$

Properties of the Hamiltonian

$$H_t(x_t, u_t) = L_t(x_t, u_t) + \lambda_t^T f_t(x_t, u_t)$$

$$\begin{aligned}
\dot{H} &= \frac{\partial}{\partial t} H + \frac{\partial}{\partial u} H \ \ \dot{u} + \frac{\partial}{\partial x} H \ \ \dot{x} + \frac{\partial}{\partial \lambda} H \ \ \dot{\lambda} \\
&= \frac{\partial}{\partial t} H + \frac{\partial}{\partial u} H \ \ \dot{u} + \frac{\partial}{\partial x} H \ \ f + f^T \dot{\lambda} \\
&= \frac{\partial}{\partial t} H + \frac{\partial}{\partial u} H \ \ \dot{u} + \left[\frac{\partial}{\partial x} H + \dot{\lambda}^T \right] f \\
&= \frac{\partial}{\partial t} H \qquad = 0 \qquad \text{for time invariant problems}
\end{aligned}$$

along the optimal trajectories for x, u and λ.

▸ Motion control

Proof The Lagrange function for the problem is:

$$J_L = \phi_T(x_T) + \int_0^T L_t(x_t, u_t)dt + \int_0^T \lambda_t^T \left[f_t(x_t, u_t) - \dot{x}_t\right] dt$$

If partial integration

$$\int_0^T \lambda^T \dot{x} dt + \int_0^T \dot{\lambda}^T x dt = \lambda_T^T x_T - \lambda_0^T x_0$$

is introduced the Lagrange function can be written as:

$$J_L = \phi_T(x_T) + \lambda_0^T x_0 - \lambda_T^T x_T + \int_0^T \left(L_t(x_t, u_t) + \lambda_t^T f_t(x_t, u_t) + \dot{\lambda}_t^T x_t\right) dt$$

and the Euler-Lagrange equations emerge from the stationarity of the Lagrange function.

$$\begin{aligned} dJ_L &= \left(\frac{\partial}{\partial x_T}\phi_T - \lambda_t^T\right) dx_T \\ &+ \int_0^T \left(\frac{\partial}{\partial x}L + \lambda^T \frac{\partial}{\partial x} f + \dot{\lambda}^T\right) \delta x dt \\ &+ \int_0^T \left(\frac{\partial}{\partial u}L + \lambda^T \frac{\partial}{\partial u} f\right) \delta u dt \end{aligned}$$

The Fundamental Lemma of the calculus of variation

The Fundamental Lemma: Let f_t be a continuous real-values function defined on $a \leq t \leq b$ and suppose that:

$$\int_a^b f_t \delta_t \; dt = 0$$

for any $\delta_t \in C^2[a, b]$ satisfying $\delta_a = \delta_b = 0$. Then

$$\boxed{f_t \equiv 0 \qquad t \in [a, b]}$$

Motion control

Optimal stepping (in continuous time) but in one dimension. Consider the problem of bringing the system

$$\dot{x} = u \qquad x_0 = \underline{x}_0$$

from the initial state along a trajectory such the cost

$$J = \frac{1}{2} p x_T^2 + \int_0^T \frac{1}{2} u_t^2$$

is minimized. The Hamiltonian function is

$$H_t = \frac{1}{2} u_t^2 + \lambda_t u_t$$

and the Euler-Lagrange equations are:

$$\begin{aligned} \dot{x} &= u \\ -\dot{\lambda} &= 0 \\ 0 &= u_t + \lambda_t \end{aligned}$$

with boundary conditions:

$$x_0 = \underline{x}_0 \qquad \lambda_T = p x_T$$

The last two are easily solved:

$$\lambda_t = p x_T \qquad\qquad u_t = -\lambda_t = -p x_T$$

The state equation (with the solution to u_t) gives us

$$x_t = \underline{x}_0 - p x_T \ t \qquad\qquad x_T = \underline{x}_0 - p x_T \ T$$

from which we can find

$$x_T = \frac{1}{1+pT}\underline{x}_0 \qquad \to 0 \qquad \text{for} \qquad p \to \infty$$

and then

$$x_t = \left(1 - \frac{p}{1+pT} \ t \right) \underline{x}_0$$

$$\lambda_t = \frac{p}{1+pT}\underline{x}_0 \qquad u_t = -\frac{p}{1+pT}\underline{x}_0$$

and the Hamilton function is constant:

$$H = -\frac{1}{2}\left[\frac{p}{1+pT}\underline{x}_0\right]$$

Consider the linear dynamic system

$$\dot{x} = Ax_t + Bu_t \qquad x_0 = \underline{x}_0$$

and the cost function

$$J = \frac{1}{2}x_T^T P x_T + \frac{1}{2}\int_0^T x_t^T Q x_t + u_t^T R u_t \; dt$$

The problem has the Hamiltonian:

$$H = \frac{1}{2}x_t^T Q x_t + \frac{1}{2}u_t^T R u_t + \lambda^T (Ax_t + Bu_t)$$

and the Euler-Lagrange equations:

$$\begin{aligned} \dot{x} &= Ax_t + Bu_t & x_0 &= \underline{x}_0 \\ -\dot{\lambda}_t^T &= x_t^T Q + \lambda_t^T A & \lambda_T^T &= x_T^T P \\ 0 &= u_t^T R + \lambda_t^T B \end{aligned}$$

or

$$\boxed{\begin{aligned} \dot{x} &= Ax_t + Bu_t & x_0 &= \underline{x}_0 \\ -\dot{\lambda}_t &= Qx_t + A^T \lambda_t & \lambda_T &= Px_T \\ u_t &= -R^{-1}B^T \lambda_t \end{aligned}}$$

We will try the candidate function:

$$\lambda_t = S_t x_t$$

Then

$$\dot{\lambda}_t = \dot{S}_t x_t + S_t \dot{x}_t = \dot{S}_t x_t + S_t \left(A x_t - B R^{-1} B^T S_t x_t \right)$$

If inserted in the costate equation

$$-\dot{\lambda}_t = Q x_t + A^T \lambda_t$$

$$-\dot{S}_t x_t - S_t \left(A x_t - B R^{-1} B^T S_t x_t \right) = Q x_t + A^T S_t x_t$$

then for every x_t:

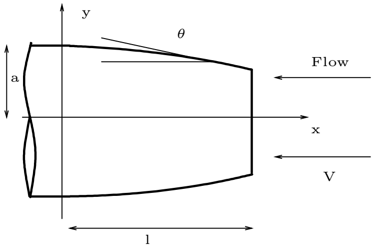
$$-\dot{S}_t x_t = S_t A x_t + A^T S_t x_t + Q x_t - S_t B R^{-1} B^T S_t x_t$$

which fulfilled if (the Riccati equation):

$$\boxed{-\dot{S}_t = S_t A + A^T S_t + Q - S_t B R^{-1} B^T S_t \qquad S_T = P}$$

$$\boxed{u_t = -R^{-1} B^T S_t x_t}$$

Find the shape i.e. $r(x)$ of a axial symmetric nose, such that the drag is minimized.



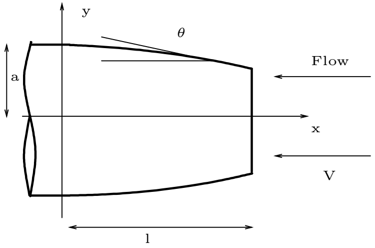
The decision $u(x)$ is the slope of the profile:

$$\frac{\partial r}{\partial x} = -u = -tan(\theta) \qquad r(0) = a$$

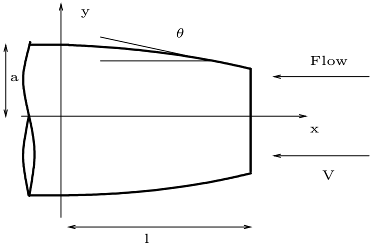
Find the shape i.e. $r(x)$ of a axial symmetric nose, such that the drag is minimized.

$$D = q \int_0^a C_p(\theta) 2\pi r dr$$

$q = \frac{1}{2}\rho V^2$ (Dynamic pressure) $\qquad C_p(\theta) = 2sin(\theta)^2$ for $\theta \geq 0$



Minimum drag nose shape (Newton)



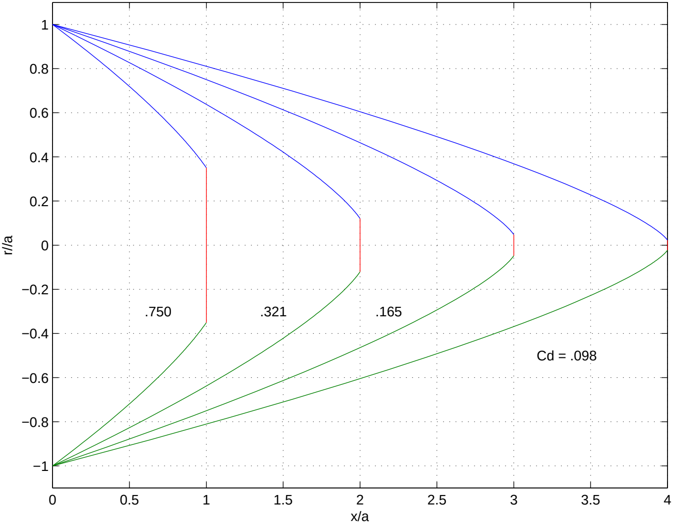
Dynamic:

$$\frac{\partial r}{\partial x} = -u \qquad r_0 = a \qquad\qquad tan(\theta) = u$$

Cost function (drag coefficient, including a blunt nose):

$$C_d = \frac{D}{q\pi a^2} = 2r_l^2 + 4\int_0^l \frac{ru^3}{1+u^2}dx \qquad\qquad \leq 1$$

Minimum drag nose shape (Newton)



Reading guidance

DO: 11-14, 27-34