POLITECNICO
MILANO 1863

EXERCISE 4 – NONLINEAR OPTIMAL CONTROL (part 2)

EMANUELE Riva
Course: Mechatronic systems and laboratory

Department of Mechanical Engineering, Politecnico di Milano, Italy

AGENDA

- *THEORY: direct transcription (direct method)*
- *EXAMPLE 1*

THEORY REVIEW: problem statement

- *Consider a "simple" optimal control problem (This is a* **fixed end-time free end-point** *optimal control problem):*

$$
\begin{aligned}
\min_{\mathbf{u} \in \mathcal{C}^1[t_0, t_f]} \quad & J[\mathbf{u}] := \phi(t_f, \mathbf{x}(t_f)) + \int_{t_0}^{t_f} L(\mathbf{x}(t), \mathbf{u}(t))\, \mathrm{d}t \\
\text{s.t.} \quad & \dot{\mathbf{x}}(t) = \mathbf{f}(\mathbf{x}(t), \mathbf{u}(t)) \\
& \mathbf{x}\left(t_0\right) = \mathbf{x}_0
\end{aligned}
$$

- *Note that constraints on the final end-point, path constraints along the trajectory and control constraints can be accommodated as well. It is also possible to solve free end-time problems (refer to lecture notes, slides and reference textbooks).*

THEORY REVIEW: problem transcription

- *Discretized dynamics (explicit Euler integration scheme):*

$$\mathbf{x}_{i+1} = \mathbf{x}_i + h_i\,\mathbf{f}(\mathbf{x}_i, \mathbf{u}_i) \quad i = 0, \ldots, N-1$$

- *The same numerical discretization is applied to the* **cost functional***, that is transcribed into the* **objective function***:*

$$J \approx \phi(t_N, \mathbf{x}_N) + \sum_{i=0}^{N-1} L(\mathbf{x}_i, \mathbf{u}_i)\, h_i$$

- *We keep the state and control variables as* **independent** *optimization variables coupled by a series of equality constraints (i.e. the dynamics).*

THEORY REVIEW: problem transcription

- *The resulting NLP problem is therefore a* **large-scale** *equality constrained problem:*

$$\begin{aligned} \min_{\mathbf{x}_i,\mathbf{u}_i} \quad & \phi(t_N,\mathbf{x}_N) + \sum_{i=0}^{N-1} L(\mathbf{x}_i,\mathbf{u}_i)\,h_i \\ \text{s.t.} \quad & \mathbf{x}_{i+1} = \mathbf{x}_i + h_i\,\mathbf{f}(\mathbf{x}_i,\mathbf{u}_i) \quad i = 0,\ldots,N-1 \\ & \mathbf{x}_0 = \bar{\mathbf{x}}_0 \end{aligned}$$

$$dim(x) = n_x$$
$$dim(u) = n_u$$
$$(n_x + n_u) \times N + n_x$$

- *A Lagrange function is built by adjoining the dynamic constraint to the objective function via Lagrange multipliers:*

$$\mathcal{L} = \phi(t_N,\mathbf{x}_N) + \boldsymbol{\lambda}_0^\top\left(\mathbf{x}_0 - \bar{\mathbf{x}}_0\right) + \sum_{i=0}^{N-1}\left[L(\mathbf{x}_i,\mathbf{u}_i)\,h_i + \boldsymbol{\lambda}_{i+1}^\top\left(\mathbf{x}_i + h_i\,\mathbf{f}(\mathbf{x}_i,\mathbf{u}_i) - \mathbf{x}_{i+1}\right)\right]$$

THEORY REVIEW: problem transcription

$$\mathcal{L} = \phi(t_N, \mathbf{x}_N) + \boldsymbol{\lambda}_0^\top \Big(\mathbf{x}_0 - \bar{\mathbf{x}}_0\Big) + \sum_{i=0}^{N-1} \Big[L(\mathbf{x}_i, \mathbf{u}_i)\, h_i + \boldsymbol{\lambda}_{i+1}^\top \Big(\mathbf{x}_i + h_i\, \mathbf{f}(\mathbf{x}_i, \mathbf{u}_i) - \mathbf{x}_{i+1}\Big)\Big]$$

THEORY REVIEW: problem transcription

- *The KKT conditions for this problem for* $i \neq 0, N$ *are:*

$$\nabla \mathcal{L}_{\boldsymbol{\lambda}_{i+1}} = \mathbf{x}_i + h_i \mathbf{f}(\mathbf{x}_i, \mathbf{u}_i) - \mathbf{x}_{i+1} = \mathbf{0}$$

$$\nabla \mathcal{L}_{\mathbf{x}_i} = h_i \Big(\frac{\partial L}{\partial \mathbf{x}}^\top \Big|_{\mathbf{x}_i, \mathbf{u}_i} + \frac{\partial \mathbf{f}}{\partial \mathbf{x}}^\top \Big|_{\mathbf{x}_i, \mathbf{u}_i} \boldsymbol{\lambda}_{i+1} \Big) + \boldsymbol{\lambda}_{i+1} - \boldsymbol{\lambda}_i = \mathbf{0}$$

$$\nabla \mathcal{L}_{\mathbf{u}_i} = h_i \frac{\partial \mathbf{f}}{\partial \mathbf{u}}^\top \Big|_{\mathbf{x}_i, \mathbf{u}_i} \boldsymbol{\lambda}_{i+1} + \frac{\partial L}{\partial \mathbf{u}}^\top \Big|_{\mathbf{x}_i, \mathbf{u}_i} h_i = \mathbf{0}$$

- *While for initial and final instant we have:*

$$\nabla \mathcal{L}_{\mathbf{x}_N} = \frac{\partial \phi}{\partial \mathbf{x}} \Big|_{\mathbf{x}_N, \mathbf{u}_N}^\top - \boldsymbol{\lambda}_N = \mathbf{0}$$

$$\nabla \mathcal{L}_{\boldsymbol{\lambda}_0} = \mathbf{x}_0 - \bar{\mathbf{x}}_0 = \mathbf{0}$$

THEORY REVIEW: problem transcription

- *As the time discretization goes to zero we also have that* $t_{i+1} \to t_i$ *and thus we can recover the continuous time formulation. Rearranging the first equation and passing to the limit we get:*

$$\frac{\mathbf{x}_{i+1} - \mathbf{x}_i}{h_i} = \mathbf{f}(\mathbf{x}_i, \mathbf{u}_i) \quad \longrightarrow \quad \dot{\mathbf{x}}(t) = \mathbf{f}(\mathbf{x}(t), \mathbf{u}(t))$$

- *The second KKT condition gives:*

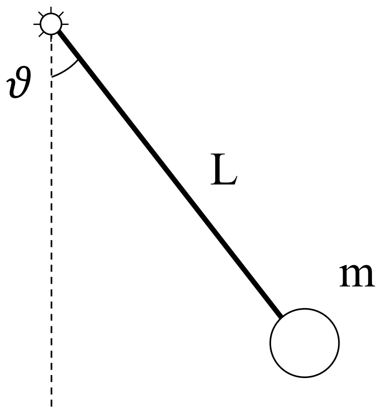
$$\frac{\boldsymbol{\lambda}_{i+1} - \boldsymbol{\lambda}_i}{h_i} = -\frac{\partial \mathbf{f}}{\partial \mathbf{x}}^\top \bigg|_{\mathbf{x}_i, \mathbf{u}_i} \boldsymbol{\lambda}_{i+1} - \frac{\partial L}{\partial \mathbf{x}}^\top \bigg|_{\mathbf{x}_i, \mathbf{u}_i} \quad \longrightarrow \quad \dot{\boldsymbol{\lambda}}(t) = -\frac{\partial \mathbf{f}}{\partial \mathbf{x}}^\top \boldsymbol{\lambda}(t) - \frac{\partial L}{\partial \mathbf{x}}^\top$$

- *While dividing by* h_i *the third KKT condition gives:*

$$\frac{\partial \mathbf{f}}{\partial \mathbf{u}}^\top \bigg|_{\mathbf{x}_i, \mathbf{u}_i} \boldsymbol{\lambda}_{i+1} + \frac{\partial L}{\partial \mathbf{u}}^\top \bigg|_{\mathbf{x}_i, \mathbf{u}_i} = \mathbf{0} \quad \longrightarrow \quad \frac{\partial \mathbf{f}}{\partial \mathbf{u}}^\top \boldsymbol{\lambda}(t) + \frac{\partial L}{\partial \mathbf{u}}^\top = \mathbf{0}$$

- *The first and last instant gives the boundary conditions:* $\boldsymbol{\lambda}(t_f) = \frac{\partial \phi}{\partial \mathbf{x}}\Big|_{t_f, \mathbf{x}(t_f)}^\top \qquad \mathbf{x}(t_0) = \bar{\mathbf{x}}_0$

EXAMPLE 1



Equation of motion:

$$\begin{cases} \dot{x}_2 = -2\zeta\omega_0 x_2 - \omega_0^2 \sin(x_1) + \dfrac{c(t)}{mL^2} \\ \dot{x}_1 = x_2 \end{cases}$$

State-space formulation:

$$\dot{\boldsymbol{x}} = f(\boldsymbol{x}, u) \qquad \text{with:} \qquad \boldsymbol{x} = [x_2, x_1]^T = \left[\dot{\theta}, \theta\right]^T$$

Initial conditions: $\boldsymbol{x}_i = [0, 0]^T$

Evaluation of the stability and control-input matrices:

$$A = \frac{\partial f}{\partial \boldsymbol{x}} = \begin{bmatrix} -2\zeta\omega_0 & -\omega_0^2 \cos(x_1) \\ 1 & 0 \end{bmatrix} \qquad B = \frac{\partial f}{\partial u} = \begin{bmatrix} \dfrac{1}{mL^2} \\ 0 \end{bmatrix}$$

EXAMPLE 1

$$
\begin{aligned}
\min_{\mathbf{x}_i,\mathbf{u}_i} J = \quad & \phi(t_N, \mathbf{x}_N) + \sum_{i=0}^{N-1} L(\mathbf{x}_i, \mathbf{u}_i)\, h_i \\
\text{s.t.} \quad & c = \mathbf{x}_i + h_i\, \mathbf{f}(\mathbf{x}_i, \mathbf{u}_i) - \mathbf{x}_{i+1} = 0 \quad i = 0, \dots, N-1 \\
& \bar{\mathbf{x}}_0 - \mathbf{x}_0 = 0
\end{aligned}
$$

We will need (if the constraint is expressed in the form $c = 0$): $J(\boldsymbol{z}),\ c(\boldsymbol{z}),\ \nabla J(\boldsymbol{z}),\ \nabla c(\boldsymbol{z})$

with: $\boldsymbol{z} = [\boldsymbol{x_0}, u_0, \dots, \boldsymbol{x_i}, u_i, \dots, \boldsymbol{x_N}]^T$

$$
\nabla J = \frac{\partial \phi(x_N)}{\partial \boldsymbol{z}} + \sum_{i=0}^{N-1} h_i \frac{\partial L(\boldsymbol{x_i}, u_i)}{\partial \boldsymbol{z}} \qquad \nabla c = \frac{\partial \boldsymbol{x_i}}{\partial \boldsymbol{z}} + h_i \frac{\partial f(\boldsymbol{x_i}, u_i)}{\partial \boldsymbol{z}} - \frac{\partial \boldsymbol{x_{i+1}}}{\partial \boldsymbol{z}}
$$

EXAMPLE 2

```
X = fmincon(FUN,X0,A,B,Aeq,Beq,LB,UB,NONLCON)

fmincon attempts to solve problems of the form:
     min F(X)  subject to:  A*X  <= B, Aeq*X  = Beq (linear constraints)
      X                     C(X) <= 0, Ceq(X) = 0   (nonlinear constraints)
                               LB <= X <= UB        (bounds)

fmincon subjects the minimization to the constraints defined in NONLCON.

The function NONLCON accepts X and returns the vectors C and Ceq, representing
the nonlinear inequalities and equalities respectively.

fmincon minimizes FUN such that C(X) <= 0 and Ceq(X) = 0. (Set LB = [] and/or
UB = [] if no bounds exist.)
```

EXAMPLE 2

```
% Minimize ObjFun s.t. NLcon
[z,fval] = fmincon(ObjFun,z0,A,b,Aeq,beq,lb,ub,NLcon,options);

%___ Define objective function and constraints ___________________________%

ObjFun = @(z) cost_and_grad(z,param);
NLcon = @(z) con_and_grad(z,param);

%___ store the necessary stuff in a structure ____________________________%

param.N = N;                param.L = L;
param.nu = nu;              param.Lx = Lx;
param.nx = nx;              param.Lu = Lu;
param.dx = dx;              param.p = p;
param.x_i = x_i;            param.px = px;
param.fx = fx;              param.h = h;
param.fu = fu;
```

EXAMPLE 2

```
function [cost,grad] = cost_and_grad(z,param)

% extract states and control from z
x = zeros(nx,N+1); u = zeros(nu,N);
for ii = 0:N
    x(:,ii+1) = z((1 + ii*(nu + nx)):(nx + ii*(nu + nx)));
end
for ii = 0:N-1
    u(:,ii+1) = z((1 + nx + ii*(nu + nx)):(nx + nu + ii*(nu + nx)));
end

% Cost Function
cost = 0;
for ii = 1:N
    cost = cost + h*L(x(:,ii),u(:,ii));
end
cost = cost + p(x(:,end));
```

EXAMPLE 2

```
% Gradient of the objective function

grad = zeros(size(z));
for ii = 0:N-1
grad((1 + ii*(nu + nx)):(nx + ii*(nu + nx)),1) = h*Lx(x(:,ii + 1),u(:,ii + 1));
grad((1 + nx + ii*(nu + nx)):(nx + nu + ii*(nu + nx)),1) = h*Lu(x(:,ii + 1),u(:,ii + 1));
end
grad(end - nx + 1:end,1) = px(x(:,end));
```

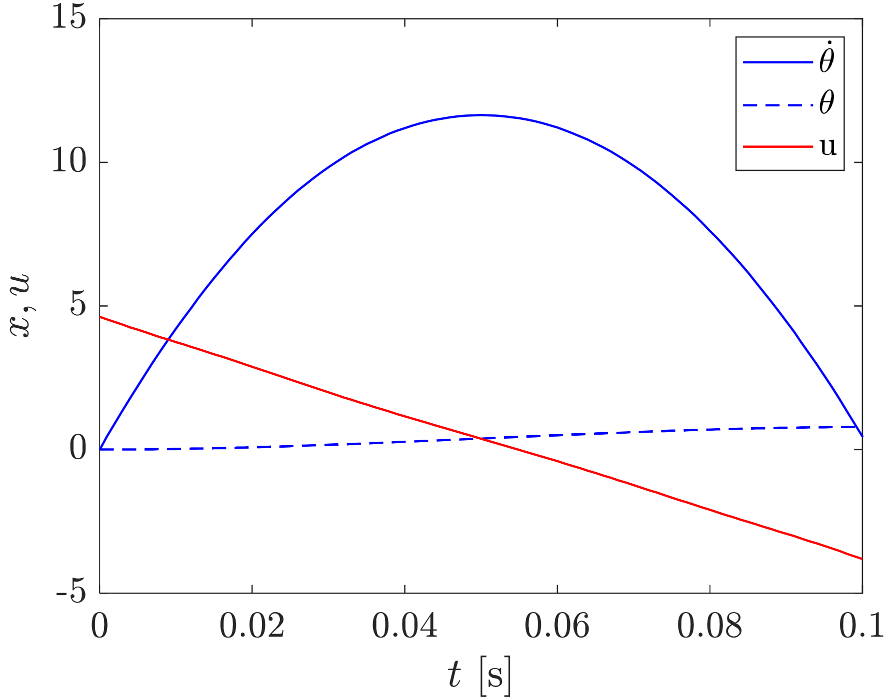
EXAMPLE 2

```
function [c,con,g,grad] = con_and_grad(z,param)

% extract states and control from z
x = zeros(nx,N+1); u = zeros(nu,N);
for ii = 0:N
    x(:,ii+1) = z((1 + ii*(nu + nx)):(nx + ii*(nu + nx)));
end
for ii = 0:N-1
    u(:,ii+1) = z((1 + nx + ii*(nu + nx)):(nx + nu + ii*(nu + nx)));
end

% Constraint function
c = []; % here is room for inequality constraint
con = zeros(N*nx,1);
con(1:nx) = x_i - x(:,1) ; % initial condition constraint

for ii = 1:N
    con((1:nx) + ii*nx) = x(:,ii) + h*dx(x(:,ii),u(:,ii)) - x(:,ii+1);
end
```

EXAMPLE 2

```
% Gradient of the constraint

if nargout > 2
g = []; % here is room for inequality constraint
grad = zeros(nx,N*(nu+nx) + nx); % gradient of a vector
grad(1:nx,1:nx) = - eye(nx);
for ii = 1:N
grad((1 + nx +(ii - 1)*nx):((nx + ii*nx)),(1 +(ii - 1)*(nx+nu)):((ii - 1)*(nx+nu) + nx)) = ...
         eye(nx) + h*fx(x(:,ii),u(:,ii));
grad((1 + nx +(ii - 1)*nx):((nx + ii*nx)),(1 +(ii)*(nx+nu)):((ii)*(nx+nu) + nx)) = ...
         - eye(nx);
grad((1 + nx +(ii - 1)*nx):((nx + ii*nx)),(1 +(ii - 1)*(nx+nu) + nx):((ii - 1)*(nx+nu) + nx +
nu)) = ...
         + h*fu(x(:,ii),u(:,ii));
end
grad = grad.';
end
```

EXAMPLE 2



Weights:

$$R = 0.01$$

$$P = \begin{bmatrix} 1 & 0 \\ 0 & 100 \end{bmatrix}$$

HANDS-ON

- *Try to implement direct transcription method to the example N° 3 of the third training session.*