IC-32 **Optimal Control** **Spring 2009**

Benoît Chachuat
ME C2 426, Ph: 35148, benoit@mcmaster.ca

Problem Set #8 — With Solutions

1. The objective of this problem is to solve, via the **direct simultaneous approach**, the following scalar optimal control problem:

$$\text{minimize:} \quad \int_0^1 \tfrac{1}{2}[u(t)]^2 \, \mathrm{d}t \tag{1}$$

$$\text{subject to:} \quad \dot{x}(t) = -x(t) + u(t); \qquad x(0) = 1 \tag{2}$$

$$x(1) = 0. \tag{3}$$

For simplicity, a **piecewise-constant control parameterization** over n_s stages is considered for approximating the control profile,

$$u(t) = \omega^k, \quad t_{k-1} \leq t \leq t_k, \quad k = 1, \ldots, n_s,$$

with the time stages being equally spaced, $t_k = \frac{k}{n_s}$.

On the other hand, the response is to be approximated, on each stage, by **Lagrange polynomials of degree** $N = 3$,

$$x(t) = \sum_{i=0}^{3} \xi_i^k \phi_i^{(3)} \left(\tfrac{t - t_{k-1}}{t_k - t_{k-1}} \right), \quad t_{k-1} \leq t \leq t_k, \quad k = 1, \ldots, n_s,$$

with

$$\phi_i^{(3)}(\tau) = \prod_{\substack{q=0 \\ q \neq i}}^{3} \frac{\tau - \tau_q}{\tau_i - \tau_q},$$

and the collocation points $\tau_0, \ldots, \tau_3$ as given in the following table:

τ_0	τ_1	τ_2	τ_3
0	0.25	0.5	0.75

Questions:

(a) Reformulate the optimal control problem into the Mayer form, then discretize this problem as per the direct simultaneous approach; give the expressions of the cost and all the constraints in the resulting NLP problem.

Solutions. Reformulating the problem into the Mayer form proceeds by introducing an extra state variable, z, defined by the differential equation,

$$\dot{z}(t) = \frac{1}{2}[u(t)]^2,$$

with initial condition $z(0) = 0$. An equivalent optimal control formulation is then obtained as:

$$\begin{aligned} \text{minimize:} \quad & z(1) \\ \text{subject to:} \quad & \dot{x}(t) = -x(t) + u(t); \qquad x(0) = 1 \\ & \dot{z}(t) = \frac{1}{2}[u(t)]^2; \qquad z(0) = 0 \\ & x(1) = 0. \end{aligned}$$

The next step consists in discretizing the state and control profiles as per the indicated parameterization (piecewise constant control and piecewise cubic state parameterizations). The decision variables in the resulting NLP problem correspond to the control coefficients ω^k and the state coefficients $\boldsymbol{\xi}^k = [\xi_0^k, \ldots, \xi_3^k]$ and $\boldsymbol{\zeta}^k = [\zeta_0^k, \ldots, \zeta_3^k]$:

$$\begin{aligned}
\text{minimize:} \quad & \sum_{i=0}^{3} \zeta_i^{n_s} \phi_i^{(3)}(1) \\
\text{subject to:} \quad & \sum_{i=0}^{3} \xi_i^k \Big[\frac{1}{t_k - t_{k-1}} \dot{\phi}_i^{(3)}(\tau_q) + \underbrace{\phi_i^{(3)}(\tau_q)}_{=\delta_{i,q}} \Big] - w^k = 0, \quad \forall k = 1 \ldots n_s,\ \forall q = 1 \ldots 3 \\
& \sum_{i=0}^{3} \xi_i^1 \phi_i^{(3)}(0) = 1, \qquad \sum_{i=0}^{3} \xi_i^k \phi_i^{(3)}(0) = \sum_{i=0}^{3} \xi_i^{k-1} \phi_i^{(3)}(1), \quad \forall k = 2 \ldots n_s \\
& \sum_{i=0}^{3} \zeta_i^k \dot{\phi}_i^{(3)}(\tau_q) - \frac{1}{2} \left(w^k\right)^2 = 0, \qquad \forall k = 1 \ldots n_s,\ \forall q = 1 \ldots 3 \\
& \sum_{i=0}^{3} \zeta_i^1 \phi_i^{(3)}(0) = 0, \qquad \sum_{i=0}^{3} \zeta_i^k \phi_i^{(3)}(0) = \sum_{i=0}^{3} \zeta_i^{k-1} \phi_i^{(3)}(1), \quad \forall k = 2 \ldots n_s \\
& \sum_{i=0}^{3} \xi_i^{n_s} \phi_i^{(3)}(1) = 0.
\end{aligned}$$

Observe that the discretization of each differential equation along with its initial condition yields a total of $4 \times n_s$ equality constraints, i.e., it exactly matches the number of state collocation coefficients. Overall, the effective number of degrees of freedom in the foregoing NLP problem thus corresponds to the number of control coefficients minus the number of terminal constraints, $n_s - 1$.

(b) In MatLab®, write two m-files that calculate the cost value and the constraint values in the foregoing NLP problem, respectively, for a variable number n_s of stages and corresponding values of the control coefficients $\omega^1, \ldots, \omega^{n_s}$ and the state coefficients $\xi^1, \ldots, \xi^{n_s}, \zeta^1, \ldots, \zeta^{n_s}$.

- Use the m-files `lagrange.m` and `dlagrange.m` to calculate the Lagrange polynomials and their time derivatives; these functions can be retrieved from the class website

Solutions. For simplicity, the following m-file breaks down the vector of decision variables, $\mathbf{p}^\mathsf{T} \triangleq [\omega^1, \ldots, \omega^{n_s}, \boldsymbol{\xi}^{1\mathsf{T}}, \ldots, \boldsymbol{\xi}^{n_s\mathsf{T}}, \mathbf{zeta}^{1\mathsf{T}}, \ldots, \boldsymbol{\zeta}^{n_s\mathsf{T}}]$ into vector/matrix components $\boldsymbol{\omega}$, $\boldsymbol{\xi}$ and $\boldsymbol{\zeta}$.

var2coef.m

```
function [ w, x, z ] = var2coef( p, ns, N )

  w = p(1:ns);

  for k=1:ns
    x(k,1:N+1) = p(ns+(k-1)*(N+1)+1:ns+k*(N+1));
    z(k,1:N+1) = p(ns+(ns+k-1)*(N+1)+1:ns+(ns+k)*(N+1));
  end

end
```

The following function calculates the cost value in the discretized NLP problem, for a given number n_s of stages and given values of the state/control coefficients $\mathbf{p}$.

obj.m

```
function [ J, dJ ] = obj( p, ns, ts, N, tau )

```

```
  [ w, x, z ] = var2coef( p, ns, N );

  if nargout == 1

    % Terminal cost
    J = lagrange(z(ns,:),tau,1);

  else

    disp(sprintf('Error: Analytical cost derivatives not yet implemented!'));

  end
end
```

Likewise, the following function calculates the constraint values in the discretized NLP problem, for a given number n_s of stages and given values of the state/control coefficients $\mathbf{p}$.

ctr.m

```
function [ c, ceq, dc, dceq ] = ctr( p, ns, ts, N, tau )

  [ w, x, z ] = var2coef( p, ns, N );
  ic = 1;

  if nargout == 2

    for k=1:ns

      % ODEs residuals at collocation points
      for q=2:N+1
        ceq(ic)   = dlagrange(x(k,:),tau,tau(q))/(ts(k+1)-ts(k)) ...
                    +lagrange(x(k,:),tau,tau(q))-w(k);
        ceq(ic+1) = dlagrange(z(k,:),tau,tau(q))/(ts(k+1)-ts(k)) ...
                    -0.5*w(k)^2;
        ic = ic+2;
      end

      % Initial conditions
      if( k == 1 )
        ceq(ic)   = lagrange(x(k,:),tau,0.) - 1.;
        ceq(ic+1) = lagrange(z(k,:),tau,0.) - 0.;

      % Continuity conditions at time stages
      else
        ceq(ic)   = lagrange(x(k,:),tau,0.) - lagrange(x(k-1,:),tau,1.);
        ceq(ic+1) = lagrange(z(k,:),tau,0.) - lagrange(z(k-1,:),tau,1.);
      end
      ic = ic+2;
    end

    % Terminal constraint
    ceq(ic) = lagrange(x(ns,:),tau,1.) - 0.;

    c = [];

  else

    disp(sprintf('Error: Analytical constraint derivatives not yet implemented!'));

  end
```

```
end
```

The input arguments in the above functions are as follows:

- `p` vector of decision variables, $\mathbf{p}$
- `ns` number of time stages, n_s
- `ts` stage times, $t_0, t_1, \ldots, t_{n_s}$
- `N` degree of state interpolating polynomial, N
- `tau` vector of collocation points for the state interpolating polynomial, $\tau_0, \ldots, \tau_N$

The m-files `lagrange.m` and `dlagrange.m` calculating the Lagrange interpolating polynomial at a given (scaled) time and its time derivative, respectively, are repeated below for the sake of completeness.

lagrange.m

```
% y = lagrange(wk,xk,x)
% arguments:
%   wk : row vector of size N+1 containing the interpolation coefficients
%   xk : row vector of size N+1 containing the interpolation points,
%        xk(1)<...<xk(N+1)
%   x  : points at which the Lagrange polynomial is to be evaluated
% output:
%   y  : value of the Lagrange polynomial at x

function y = lagrange(wk,xk,x)

    N = length(wk);
    y = zeros(size(x));
    for k = 1:N
        prod = wk(k)*ones(size(x));
        for j = 1:N
            if j ~= k
                prod = prod .* (x-xk(j)) / (xk(k)-xk(j));
            end
        end
        y = y + prod;
    end

end
```

dlagrange.m

```
% dy = dlagrange(wk,xk,x)
% arguments:
%   wk : row vector of size N+1 containing the interpolation coefficients
%   xk : row vector of size N+1 containing the interpolation points,
%        xk(1)<...<xk(N+1)
%   x  : points at which the Lagrange polynomial is to be evaluated
% output:
%   dy : derivative of the Lagrange polynomial at x

function dy = dlagrange(wk,xk,x)

    N = length(wk);
    y = zeros(size(x));
    for k = 1:N
        sum = zeros(size(x));
        for i = 1:N
```

```
            if i ~= k
                prod = ones(size(x));
                for j = 1:N
                    if j ~= k && j ~= i
                        prod = prod .* (x-xk(j)) / (xk(k)-xk(j));
                    end
                end
                sum = sum + prod / (xk(k)-xk(i));
            end
        end
        dy = dy + wk(k) * sum;
    end

end
```

(c) Solve the fully-discretized NLP problem using the `fmincon` function in MatLab's Optimization Toolbox:

- Set all the control and state coefficients equal to zero as the initial guess
- For simplicity, let `fmincon` calculate a finite-difference approximation of the cost and constraint derivatives; in particular, set the minimum change in variable for finite differencing to 10^{-10}
- Make sure to select the medium-scale SQP algorithm with quasi-Newton update and line-search, and set the solution point tolerance, the function tolerance and the constraint tolerance all to 10^{-6}

Plot the results for $n_s = 2$, 5, 10 and 20 stages.

Solutions. The following m-file invokes the `fmincon` function in MatLab's Optimization Toolbox to solve the discretized NLP. In particular, the m-files `obj.m` and `ctr.m` derived in the previous questions are called by `fmincon` to calculate the cost and constraints values. On the other hand, the derivatives of the cost and constraints are calculated by `fmincon` via finite differences for simplicity.

main.m

```
clear all;
format long;

% Options for NLP Solvers
optNLP = optimset( 'Algorithm', 'active-set', 'LargeScale', 'off',...
    'GradObj', 'off',  'GradConstr', 'off', 'DerivativeCheck', 'off', ...
    'Display', 'iter', 'TolX', 1e-6, 'TolFun', 1e-6, 'TolCon', 1e-6, ...
    'MaxFunEval', 10000, 'MaxIter', 1000, 'DiffMinChange', 1e-10 );

% Time Horizon and Initial State
t0 = 0;             % Initial time
tf = 1;             % Final time

% Direct simultaneous approach
ns = 20;                      % Number of time stages
ts = [t0:(tf-t0)/ns:tf];      % Time stages (equipartition)
N  = 3;                       % Degree of state interpolating polynomials
tau = [ 0, 0.25, 0.5, 0.75 ]; % Position of interpolation points

% Initial guess (all coefficients at zero)
w0 = zeros(ns,1);
x0 = zeros((N+1)*ns,1);
z0 = zeros((N+1)*ns,1);
p0 = [ w0; x0; z0 ];
```

```
25
26  % Solve fully-discretized optimal control problem
27  [popt, Jopt, erropt, iout, nuopt] = fmincon( @(p)obj(p,ns,ts,N,tau), p0, ...
28    [], [], [], [], [], [], @(p)ctr(p,ns,ts,N,tau), optNLP);
29
30  % Display results (10 points for each time stage)
31  dispopt( popt, ns, ts, N, tau, 10 );
```

Finally, the following m-file displays the optimization results.

main.m

```
1   function dispopt( p, ns, ts, N, tau, Nsamples )
2
3     [ w, x, z ] = var2coef( p, ns, N )
4
5     tspan = [];
6     wspan = [];
7     xspan = [];
8     zspan = [];
9
10    wana = [];
11    xana = [];
12
13    for k=1:ns
14      dt = [ts(k):(ts(k+1)-ts(k))/Nsamples:ts(k+1)]';
15      tspan = [ tspan; dt ];
16
17      % Results from full-discretization approach
18      wspan = [ wspan; w(k)*ones(size(dt)) ];
19      xspan = [ xspan; lagrange(x(k,:),tau,(dt-ts(k))/(ts(k+1)-ts(k))) ];
20      zspan = [ zspan; lagrange(z(k,:),tau,(dt-ts(k))/(ts(k+1)-ts(k))) ];
21
22      % Analytical solution (see textbook Example 4.23)
23      wana = [ wana; -exp(dt-1)/sinh(1) ];
24      xana = [ xana; exp(-dt)-sinh(dt)/(exp(1)*sinh(1)) ];
25    end
26
27    % Display Optimal Cost
28    disp(sprintf('Optimal cost for ns=%d:  %d', ns, zspan(end)));
29
30    % Plot results
31    figure(1)
32    plot(tspan,xspan,'r',tspan,wspan,'b')
33    xlabel('t')
34    ylabel('u(t), x(t)')
35    axis([0 1 -1 1]);
36    title(['ns=',int2str(ns)])
37    saveas(gcf,['u(t)_x(t)_',int2str(ns),'stg.eps'], 'psc2')
38
39  end
```

The results obtained for $n_s = 2$, 5, 10 and 20 stages are displayed in Figure 1 below. The corresponding optimal cost value is given in Table 1. Interestingly, it is seen that even a coarse parameterization (e.g., 5 control stages) yields a good approximation of the actual optimal control.

(d) Compare these results with the analytical solution, as given in Example 4.23 (pp.133-134) of the class textbook.

Solutions. The analytical solution for this problem can be shown to be as follows:

$$u^*(t) = -\nu^* \mathrm{e}^{t-1}$$
$$x^*(t) = \mathrm{e}^{-t} - \frac{\nu}{\mathrm{e}} \sinh(t)$$

with $\nu^* = \frac{1}{\sinh(1)}$. The full-discretization optimal control solution is found in very good agreement with the analytical solution. This is illustrated in Figure 2 for $n_s = 20$ stages.

(e) Modify the m-files so that the cost and constraint derivatives with respect to the control/state coefficients are also calculated analytically.

- Needless to say, always cross-check the analytical derivatives by comparison with forward finite differences

Resolve the fully-discretized NLP problem for $n_s = 2$, 5, 10 and 20 stages.

Solutions. To be completed...

Table 1: Optimal performance for full-discretization approach with $n_s = 2$, 5, 10 and 20 stages.

n_s	$z^*(1)$
2	0.1597
5	0.1570
10	0.1567
20	0.1566

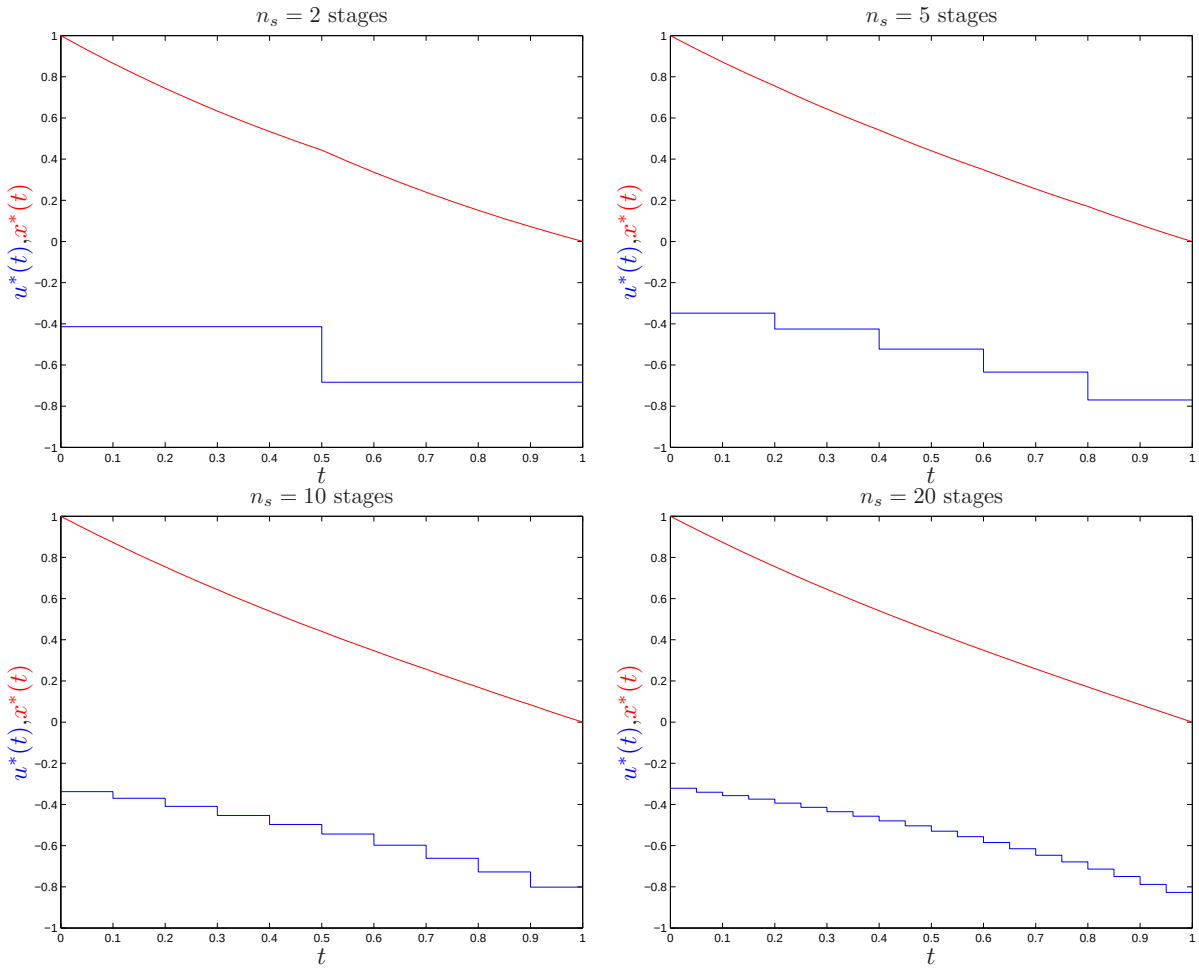

Figure 1: Optimal results for full-discretization approach with $n_s = 2$, 5, 10 and 20 stages. Blue line: control, $u(t)$; red line: state, $x(t)$.

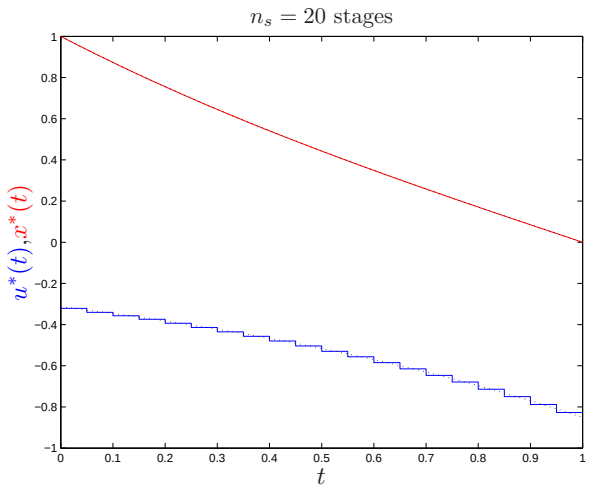

Figure 2: Comparison between the optimal solution for full-discretization approach with $n_s = 20$ stages (solid lines) and the analytical solution (dotted lines).