The objective of this problem is to solve, via the direct sequential approach, the following scalar optimal control problem:

minimize: $\int_{0}^{2} \frac{1}{2} [x_1(t)]^2 \, dt \quad (1)$
subject to: $\dot{x}_1(t) = x_2(t) + u(t); \quad x_1(0) = 1 \quad (2)$
$\dot{x}_2(t) = -u(t); \quad x_2(0) = 1 \quad (3)$
$x_1(2) = x_2(2) = 0 \quad (4)
-10 \leq u(t) \leq 10, \quad 0 \leq t \leq 2. \quad (5)$

For simplicity, a piecewise control parameterization over $n_s$ stages is considered for approximating,

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

with the time stages being equally spaced, $t_k = \frac{2k}{n_s}.$

Questions:
1. First, reformulate the optimal control problem into the Mayer form (denote the additional state variable by $x_3$).

**Solutions.** Reformulating the problem into the Mayer form proceeds by introducing an extra state variable, $x_3,$ defined by the differential equation,

$\dot{x}_3(t) = \frac{1}{2} [x_1(t)]^2,$

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

minimize: $z(2)$
subject to: $\dot{x}_1(t) = x_2(t) + u(t); \quad x_1(0) = 1$
$\dot{x}_2(t) = -u(t); \quad x_2(0) = 1$
$\dot{x_3}(t) = \frac{1}{2} [x_1(t)]^2; \quad x_3(0) = 0$
$x_1(2) = x_2(2) = 0$

2. In MatLab®, write a m-file calculating the values of the cost (1) and terminal constraints (4), for a given number $n_s$ of stages and given values of the control parameters $\omega^1, \ldots, \omega^{n_s}$.

- Use the function ode15s to integrate the differential equations; set both the absolute and relative integration tolerances to $10^{-8}$.

Application. Calculate the cost and constraint values for the following 2-stage control:

$u^0(t) = \begin{cases} 
10, & 0 \leq t < 1, \\
-10, & 1 \leq t \leq 2.
\end{cases} \quad (6)$

**Solutions.** A possible implementation is as follows:
function [ f ] = fun( x0, ns, ts, ws )

% Options for ODE solver
optODE = odeset( 'RelTol', 1e-8, 'AbsTol', 1e-8 );

% Forward state integration
z0 = [ x0 ];
for ks = 1:ns
    [tspan,zs] = ode15s( @(t,x)state(t,x,ws,ks), [ts(ks),ts(ks+1)], z0, optODE );
    z0 = zs(end,:)';
end

% Functions
f = zs(end,:);'';
end

function [ dx ] = state( t, x, w, ks )

% State system
dx=[ x(2) + w(ks);
     - w(ks);
     x(1)^2 / 2. ];
end

3. Solve the sequentially-discretized NLP problem using the \texttt{fmincon} function in MatLab’s Optimization Toolbox:

- You are to code a main program, as well as two separate m-files called by \texttt{fmincon} that calculate the values of the cost and of the constraints, respectively; these latter m-files should invoke the m-file developed in Question 2.
- For simplicity, let \texttt{fmincon} calculate a finite-difference approximation of the cost and constraint derivatives; make sure that the minimum change in variable for finite differencing is consistent with the tolerances set previously for the ODE solver ($10^{-8}$): here, a value of $10^{-5}$ appears to be a reasonable choice.
- 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 $10^{-9}$; such tight tolerances are needed because the optimal control problem is singular.
- Set all the control coefficients equal to zero as the initial guess.

Application. Solve the optimal control problem for $n_s = 2, 4, 8, 16$ and $32$ stages, then plot the results.

\textit{Solution.} A simple implementation satisfying the aforementioned specifications is given subsequently.
clear all;
clf;
format long;

Optim Options:
OptNLP = optimset('LargeScale', 'off', 'GradObj', 'off', 'GradConstr', 'off',
'DerivativeCheck', 'off', 'Display', 'iter', 'TolX', 1e-9,
'TolFun', 1e-9, 'TolCon', 1e-9, 'MaxFunEval', 300,
'DiffMinChange', 1e-5);

Time Horizon and Initial State

Initial time:
t0 = 0;
Final time:
tf = 2;
Initial state:
x0 = [1; 1; 0];
Number of stages: ns = 1, 2, 4, 8, 16, and 32:
for is = 1:5
    ns = 2*ns;
time stages (equipartition):
ts = [t0:(tf-t0)/ns:tf];
Initial Guess and Bounds for the Parameters

Lower Bounds:
wL = -10*ones(ns,1);
Upper Bounds:
wU = 10*ones(ns,1);

Sequential Approach of Dynamic Optimization

Optimization target:
[wopt] = fmincon(@(ws)obj(x0,ns,ts,ws), w0, [], [], [], [], wL, wU,
    @(ws)ctr(x0,ns,ts,ws), optNLP);

To increase the reliability and execution speed of the direct sequential procedure, you are to compute
the derivatives of the cost (1) and terminal constraints (4) with respect to the control parameters
ω^1, ..., ω^n via the forward sensitivity method.
(a) Write down the state sensitivity equations, as well as the cost and constraint derivatives per the forward sensitivity method.

(b) Duplicate the m-file developed in Question 2 above and modify it so that it calculates the cost and constraint derivatives in addition to their values; run this m-file for the 2-stage control $u^0(t)$ given in (6).

(c) Modify the two m-files passed to `fmincon` for cost/constraint function evaluation:

- In the main program, tell `fmincon` to use the user-supplied cost/constraint derivatives (instead of finite-difference derivatives).
- Use the DerivativeCheck feature of `fmincon` to detect inconsistencies in the computed derivatives!

Application. Recalculate the solution to the optimal control problem for $n_s = 2, 4, 8, 16$ and $32$ stages, and make sure that you get the same results as previously.

Solution. A possible implementation that calculates the cost and constraints derivatives via the forward sensitivity approach is given subsequently. The m-file `disopt.m` plots the optimization results and then saves these plots.

```matlab
main.m
1 clear all;
2 clf;
3 format long;
4
5 % Options for ODE & NLP Solvers
6 optODE = odeset( 'RelTol', 1e-8, 'AbsTol', 1e-8 );
7 optNLP = optimset( 'LargeScale', 'off', 'GradObj', 'on', 'GradConstr', 'on',
8       'DerivativeCheck', 'off', 'Display', 'iter', 'TolX', 1e-9,
9       'TolFun', 1e-9, 'TolCon', 1e-9, 'MaxFunEval', 300 );
10
11 % Time Horizon and Initial State
12 t0 = 0; % Initial time
13 tf = 2; % Final time
14 x0 = [ 1; 1; 0 ]; % Initial state
15
16 ns = 1;
17 for is = 1:5
18    ns = 2*ns; % Number of stages: ns = 2, 4, 8, 16, and 32
19    ts = [t0:(tf-t0)/ns:tf]; % Time stages (equipartition)
20
21    % Initial Guess and Bounds for the Parameters
22    w0 = zeros(ns,1);
23    wL = -10*ones(ns,1);
24    wU = 10*ones(ns,1);
25
26    % Sequential Approach of Dynamic Optimization
27    [ wopt ] = fmincon( @(ws)obj(x0,ns,ts,ws,optODE), w0, [], [], [], [],
28       'TolFun', 1e-9, 'TolCon', 1e-9, 'MaxFunEval', 300 );
29
30    dispopt( x0, ns, ts, wopt, optODE );
31 end

obj.m
1 function [ J, dJ ] = obj( x0, ns, ts, ws, optODE )
```
if nargout == 1
  f = fun( x0, ns, ts, ws, optODE );
  J = f(3);
else
  [f,df] = fun( x0, ns, ts, ws, optODE );
  J = f(3);
  dJ = df(3,:);
end
end

function [ c, ceq, dc, dceq ] = ctr( x0, ns, ts, ws, optODE )
if nargout == 2
  f = fun( x0, ns, ts, ws, optODE );
  ceq = f(1:2);
  c = [];
else
  [f,df] = fun( x0, ns, ts, ws, optODE );
  ceq = f(1:2);
  dceq = df(1:2,:);
  c = [];
  dc = [];
end
end

function [ f, df ] = fun( x0, ns, ts, ws, optODE )
% Calculate function values only
if nargout == 1
  % Forward state integration
  z0 = [ x0 ];
  for ks = 1:ns
    [tspan,zs] = ode15s( @(t,x)state(t,x,ws,ks), [ts(ks),ts(ks+1)], z0, optODE );
    z0 = zs(end,:);
  end
  % Functions
  f = zs(end,:);
% Calculate both function and gradient values
else
  % Forward state & sensitivity integration
  z0 = [ x0; zeros(3*ns,1) ];
  for ks = 1:ns
    [tspan,zs] = ode15s( @(t,x)sens(t,x,ws,ks), [ts(ks),ts(ks+1)], z0, optODE );
    z0 = zs(end,:);
  end
  % Functions & Gradients
  f = zs(end,1:3);
  df = [];
end
for is = 1:ns
    df(1:3,is) = zs(end,is*3+1:is*3+3);
end
end

function [ dx ] = state( t, x, w, ks )
% State system
dx=[ x(2) + w(ks);
    - w(ks);
    x(1)^2 / 2. ];
end

function [ dx ] = sens( t, x, w, ks )
% State system
dx=[ x(2) + w(ks);
    - w(ks);
    x(1)^2 / 2. ];
% Append sensitivity system
for is = 1:length(w)
    if is == ks
        uws = 1.;
    else
        uws = 0.;
    end
    dx = [ dx;
    x(3*is+2) + uws;
    - uws;
    x(1) * x(3*is+1) ];
end
end

function dispopt( x0, ns, ts, ws, optODE )
% Forward state integration; store optimal state & control
z0 = [ x0 ];
topt = [];
xopt = [];
ypo = [];
for ks = 1:ns
    [tspan,zs] = ode15s( @(t,x)state(t,x,ws,ks), [ts(ks),ts(ks+1)], z0, optODE );
z0 = zs(end,:);
topt = [ topt; tspan ];
With the foregoing implementation, one gets the following results:

<table>
<thead>
<tr>
<th>$n_s$</th>
<th>$J^*$</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>5.858335</td>
</tr>
<tr>
<td>4</td>
<td>4.898494</td>
</tr>
<tr>
<td>8</td>
<td>4.608658</td>
</tr>
<tr>
<td>16</td>
<td>4.592631</td>
</tr>
<tr>
<td>32</td>
<td>4.586278</td>
</tr>
</tbody>
</table>

It should be noted that adding more control stages gives very little improvement in terms of the optimal cost, for $n_s$ greater than 8.

The results obtained with $n_s = 32$ are shown in Fig. 1 below. By inspection an optimal control for the problem (1–5) appears to be composed of 3 arcs:

(a) $u^*(t) = 10, \ 0 \leq t \leq t_1^*$;
(b) $u^*(t) = -x_1^*(t) - x_2^*(t), \ t_1^* \leq t \leq t_2^*$;
(c) $u^*(t) = -10, \ t_2^* \leq t \leq 2$.

These numerical results should be compared with the analytical solution given in the Example 4.34 of the class textbook.
$n_s = 32$ stages

Figure 1: Optimal results for CVP with $n_s = 32$. Left plot: optimal control $u^*$ vs. $t$; right plot: optimal responses $x_1^*$ (red curve), $x_2^*$ (blue curve) vs. $t$. 