IC-32 **Optimal Control** **Winter 2006**

Benoît Chachuat
ME C2 401, Ph: 33844, benoit.chachuat@epfl.ch

Problem Set #6 (With Corrections)

1. Consider the problem of finding the smooth curve $y(x), x_A \leq x \leq x_B$, in the vertical plane (x, y), joining given points $A = (x_A, y_A)$ and $B = (x_B, y_B)$, $x_A < x_B$, and such that a material point sliding along $y(x)$ without friction from A to B, under gravity and with initial speed $v_A \geq 0$, reaches B in a minimal time (see Fig. 1).

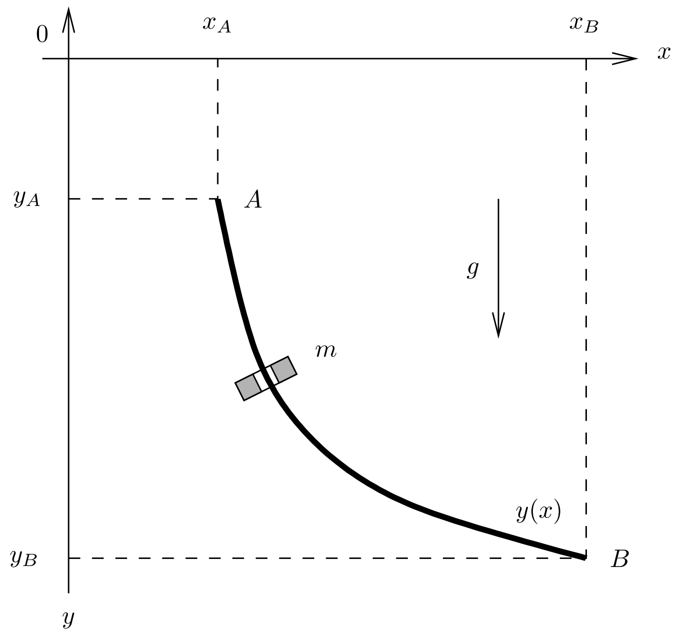

Figure 1: Brachistochrone problem.

We saw in a previous problem (Problem 4) that this formulation yields the following problem of the calculus of variations:

$$\text{minimize:} \quad \mathcal{J}(y) = \int_{x_A}^{x_B} \sqrt{\frac{1 + \dot{y}(x)^2}{\frac{v_A^2}{2g} - y(x)}} \, \mathrm{d}x \tag{1}$$

$$\text{subject to:} \quad y \in \mathcal{D} := \{y \in \mathcal{C}^1[x_A, x_B] : y(x_A) = y_A, y(x_B) = y_B\},$$

where g denotes the gravity acceleration.

(a) Show that the stationary functions $\bar{y}$ of the Brachistochrone problem are those satisfying the system of differential equations

$$\dot{y}(x) = z(x) \tag{2}$$

$$\dot{z}(x) = \frac{1+z(x)^2}{2\left(\frac{v_A^2}{2g} - y(x)\right)}, \tag{3}$$

subject to the split boundary conditions

$$y(x_A) = y_A, \quad \text{and} \quad y(x_B) = y_B. \tag{4}$$

Solution. The Lagrangian function for the Brachistochrone problem is

$$\ell(y(x), \dot{y}(x)) = \sqrt{\frac{1+\dot{y}(x)^2}{\frac{v_A^2}{2g} - y(x)}}.$$

After simplifications, the derivatives ℓ_y, $\ell_{\dot{y}}$, and $\frac{\mathrm{d}}{\mathrm{d}t}\ell_{\dot{y}}$, read

$$\ell_y(y(x), \dot{y}(x)) = \frac{\sqrt{1+\dot{y}(x)^2}}{2\left(\frac{v_A^2}{2g} - y(x)\right)^{\frac{3}{2}}}$$

$$\ell_{\dot{y}}(y(x), \dot{y}(x)) = \frac{\dot{y}(x)}{\sqrt{1+\dot{y}(x)^2}\sqrt{\frac{v_A^2}{2g} - y(x)}}$$

$$\frac{\mathrm{d}}{\mathrm{d}t}\ell_{\dot{y}}(y(x), \dot{y}(x)) = \frac{1}{\sqrt{1+\dot{y}(x)^2}\sqrt{\frac{v_A^2}{2g} - y(x)}}\left[\ddot{y}(x) + \frac{\dot{y}(x)}{2\left(\frac{v_A^2}{2g} - y(x)\right)} - \frac{\dot{y}(x)^2\ddot{y}(x)}{1+\dot{y}(x)^2}\right].$$

Stationary functions for the Lagrangian ℓ are those satisfying the Euler equation $\frac{\mathrm{d}}{\mathrm{d}t}\ell_{\dot{y}}(y(x), \dot{y}(x)) = \ell_y(y(x), \dot{y}(x))$. After simplifications, we get the second-order differential equation

$$\ddot{y}(x) = \frac{1+\dot{y}(x)^2}{2\left(\frac{v_A^2}{2g} - y(x)\right)},$$

which is readily transformed into two first-order differential equations by introducing the auxiliary variable $z(t)$ as

$$\dot{y}(x) = z(x)$$

$$\dot{z}(x) = \frac{1+z(x)^2}{2\left(\frac{v_A^2}{2g} - y(x)\right)}.$$

(b) Stationary functions to the Brachistochrone problem can be obtained by computing the solutions to the differential equations (2,3) which satisfy the boundary conditions (4). However, the boundary conditions being split (some are specified at $x = x_A$ and others at $x = x_B$), a solution cannot be obtained based on classical solvers for initial value problems. Instead, the so-called *shooting* approach is considered here, which consists of guessing the missing initial condition $z(x_A) = z_A$ so that the corresponding curve $y(x)$ verifies the end-point condition $y(x_B) = y_B$.

i. In MATLAB®, write a program plotting $y(x_B)$ vs. z_A, for z_A in the range $[-10 : 0]$; in particular:

- Use the function ode15s to integrate the differential equations (2,3), from the initial conditions (y_A, z_A);
- Set both relative and absolute tolerances to 10^{-6} in ode15s;
- Take the values $x_A = y_A = 0$, $x_B = 1$, $y_B = -0.5$ and $v_A = 1$ for the parameters (g can be set to $10\,\mathrm{m/s^2}$, for simplicity).

Estimate *visually* the value of the missing initial condition z_A giving the desired end-point condition $y(x_B) \approx y_B$.

Solution. A possible implementation is as follows:

brachistMain.m

```
clear all;

% Parameters & Specifications
xA = 0.;
yA = 0.;
xB = 1.;
yB = -0.5;
vA = 1.;
g  = 10.;

% Calculate value of y(xB) for given values of dy/dx(xA) in [-10:0]
%zA(1) = -3.19;
optODE = odeset( 'RelTol', 1e-6, 'AbsTol', [1e-6 1e-6] );
zA = [-10:0.1:0];
for k = 1:length(zA)
  [x,y] = ode15s( @(x,y)brachistRHS(x,y,vA,g), [xA xB], [yA zA(k)], optODE );
  yBres(k) = y(length(y),1);
end

%Display shoothing results
figure(1);
plot(zA,yBres,'-r');
```

brachistRHS.m

```
function [ dy ] = brachistRHS( x, y, vA, g )
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% brachistRHS calculates the right hand side of the differential
% equations obtained from Euler's first equation, relative to the
% Brachistochrone problem
% Arguments:
%   x   current x-coordinate
%   y   current value of [y,dy/dt]
%   vA  initial velocity at (xA,yA)
%   g   gravity acceleration
% Outputs:
%   dy  time derivatives of [y,dy/dt]
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
  dy = zeros(2,1);
  dy(1) = y(2);
  dy(2) = (1+y(2)^2)/(2*(vA^2/(2*g)-y(1)));
end
```

A plot of the curve $y(x_B)$ vs. z_A, for $z_A \in [-10 : 0]$ is shown in Fig. 2 below. Based on these results, a rough estimate of the (stationary) initial condition $\bar{z}_A$ such that $y(x_B) = y_B$ is obtained as:

$$\bar{z}_A \approx 3.2.$$

Note also that the numerical integration fails for values of z_A greater than -2.3, as the program attempts to calculate the square root of a negative number.

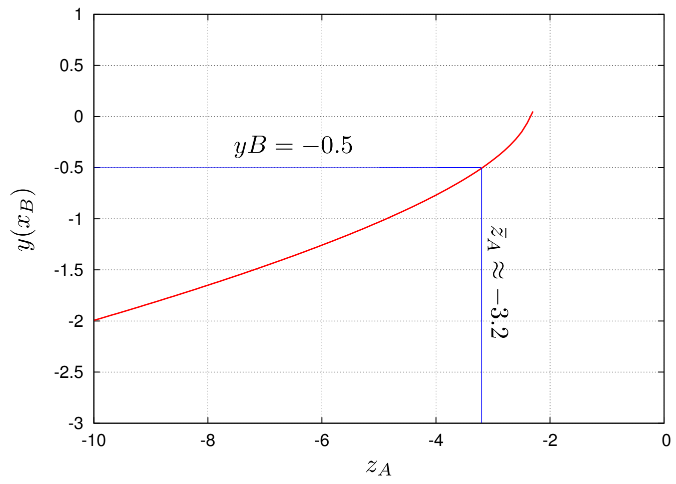

Figure 2: $y(x_B)$ vs. z_A for the Brachistochrone problem, with $x_A = y_A = 0$, $x_B = 1$, $y_B = -0.5$, $v_A = 1$ and $g = 10$.

ii. In order to obtain a more accurate estimation of $\hat{z}_A$, we now want to devise an automatic optimization procedure. In MATLAB®, write a program solving the following optimization problem with simple bound constraints:

$$\min_{z_A^L \le z_A \le z_A^U} \quad (y(x_B) - y_B)^2 ,$$

where $y(x_B)$ is obtained from the solution to the differential equations (2,3), with initial conditions $y(x_A) = y_A$ and $z(x_A) = z_A$; in particular:

- Use the function `fmincon` to solve the optimization problem;
- Consider the range $[z_A^L, z_A^U] := [-4, -3]$ for the optimization parameter z_A;
- Set the solution point tolerance, function tolerance and constraint tolerance to 10^{-6} in `fmincon`;
- For simplicity, let `fmincon` calculate a **finite-difference approximation** of the gradients of the objective function; in particular, set the minimum change in variables for finite differencing to 10^{-5};
- Take the same values as before for the parameters x_A, y_A, x_B, y_B and v_A.

Plot the resulting stationary curve $\bar{y}(x)$, and compare the results with those obtained for Problem 4.

Solution. A possible implementation is given subsequently. Note that the M-file `brachistRHS.m` has been modified for computing the value of the cost functional (1) as well.

brachistMain.m

```
clear all;

```

```
% Parameters & Specifications
xA = 0.;
yA = 0.;
xB = 1.;
yB = -0.5;
vA = 1.;
g  = 10.;

%Optimization parameters
zAL = -4;
zAU = -3;
zA0 = -3.5;

% Optimize value of zA=dy/dt(xA) so that y(xB)=yB
optNLP = optimset( 'GradObj', 'off', 'GradConstr', 'off', 'Display', 'iter',...
                    'LargeScale', 'off', 'HessUpdate', 'bfgs', 'Diagnostic', 'on',...
                    'TolX', 1e-6, 'TolFun', 1e-6, 'TolCon', 1e-6, 'MaxIter', 100,...
                    'MaxFunEval', 500, 'DiffMinChange', 1e-5 );
[zAopt, erropt, iout] = fmincon( @(zA)brachistFun(xA,yA,xB,yB,zA,vA,g,optODE),...
                                 [zA0], [], [], [], [], [zAL], [zAU], [], optNLP );

%Display optimal trajectory y(x) and results
[x,yopt] = ode15s( @(x,y)brachistRHS(x,y,vA,g), [xA xB], [yA zAopt 0.], optODE );
figure(2);
plot(x,yopt(:,1),'-r');
disp(sprintf('Minimum value of J:          %d', yopt(length(yopt),3)));
disp(sprintf('Initial condition dy/dx(xA): %d', zAopt));
disp(sprintf('Error (y(xB)-yB)^2:          %d', erropt));
```

brachistFun.m

```
function [ err ] = brachistFun( xA, yA, xB, yB, zA, vA, g, optODE )
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%brachistSta calculates the squared error relative to end-point
%specification in the Brachistochrone problem
% Arguments:
%   xA      x-coordinate of point A
%   yA      y-coordinate of point A
%   xB      x-coordinate of point B
%   yB      y-coordinate of point B
%   zA      guess for the initial value of dy/dx at point A
%   vA      initial velocity at (xA,yA)
%   g       gravity acceleration
%   optODE  options to be passed to numerical integrator
% Outputs:
%   err     squared error relative to end-point specification: (y(xB)-yB)^2
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
  [x,y] = ode15s( @(x,y)brachistRHS( x, y, vA, g ), [xA xB], [yA zA 0.], optODE );
  err = ( yB - y(length(y),1) )^2;
end
```

brachistRHS.m

```
function [ dy ] = brachistRHS( x, y, vA, g )
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

```
% brachistRHS calculates the right hand side of the differential
% equations obtained from Euler's first equation, relative to the
% Brachistochrone problem
% Arguments:
%   x   current x-coordinate
%   y   current value of [y,dy/dt,int_{xA}^x(l)]
%   vA  initial velocity at (xA,yA)
%   g   gravity acceleration
% Outputs:
%   dy  time derivatives of [y,dy/dt,l]
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
  dy = zeros(2,1);
  dy(1) = y(2);
  dy(2) = (1+y(2)^2)/(2*(vA^2/(2*g)-y(1)));
  dy(3) = sqrt((1+y(2)^2)/(vA^2/(2*g)-y(1)));
end
```

An accurate estimate of the (stationary) initial condition $\bar{z}_A$ such that $y(x_B) = y_B$ is obtained as:

$$\bar{z}_A \approx -3.1879.$$

The corresponding stationary function $\bar{y}$ is shown in Fig. 3 below. (Check that both end-point conditions are verified.) The value of the cost functional (1) is $\mathcal{J}(\bar{y}) \approx 2.0725$.

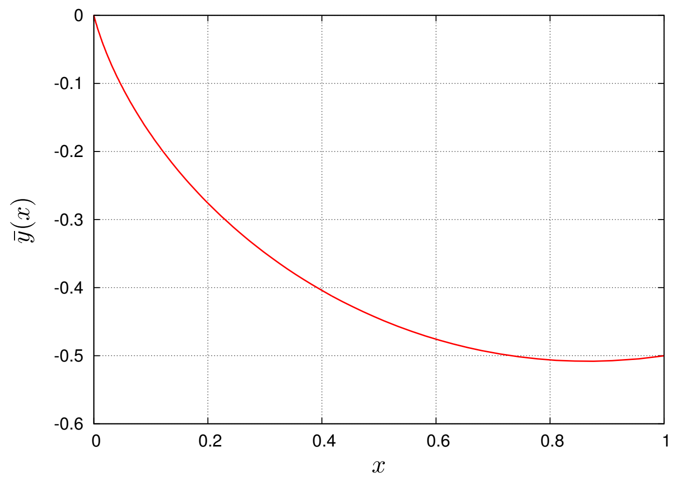

Figure 3: Stationary function $\bar{y}$ for the Brachistochrone problem, with $x_A = y_A = 0$, $x_B = 1$, $y_B = -0.5$, $v_A = 1$ and $g = 10$.