Spacetime Constraints
(Lagrange Multiplier Methods)

Byeong-Chun SHIN

November 24, 1999

< Problem Formulation >

Find a force function $f(t)$, defined on the interval $(0, 1)$, such that the objective function R,

$$R = \int_0^1 |f(t)|^2 \, dt$$

is a minimum, and such that the position function $x(t)$ satisfies

$$mx''(t) - f(t) - mg = 0 \quad t \in (0, 1) \tag{1}$$

$$x(0) = x_a, \quad x'(0) = x_{pa}$$

$$x(1) = x_b.$$

< Numerical Solution >

Given an integer $n > 0$, let $\{t_i\}$ be a sequence such that

$$t_i = (i-1)h \quad \text{with} \quad h = \frac{1}{n-1}$$

and denote by $x_i = x(t_i)$.

Finite difference formulas :

$$x_i' = \frac{x_{i+1} - x_i}{h},$$

$$x_i'' = \frac{x_{i-1} - 2x_i + x_{i+1}}{h^2}.$$

Then we have, for $1 < i < n$,

$$p_i = m\frac{x_{i-1} - 2x_i + x_{i+1}}{h^2} - f_i - mg = 0,$$

From the initial conditions, we have

$$x_1 - x_a = 0,$$

$$x_2 - x_1 - h\, x_{pa} = 0.$$

Combining these, we have

$$A\,\mathbf{x} - \mathbf{f} - \mathbf{g} = 0, \tag{2}$$

$$A = \alpha \begin{bmatrix} 1 & & & & & \\ -1 & 1 & & & & \\ 1 & -2 & 1 & & & \\ & 1 & -2 & 1 & & \\ & & \ddots & \ddots & \ddots & \\ & & & 1 & -2 & 1 \end{bmatrix}, \quad \alpha = \frac{m}{h^2},$$

$$\mathbf{x} = \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ \vdots \\ x_n \end{bmatrix}, \quad \mathbf{f} = \begin{bmatrix} 0 \\ 0 \\ f_2 \\ \vdots \\ f_{n-1} \end{bmatrix}, \quad \mathbf{g} = \begin{bmatrix} \alpha\, x_a \\ \alpha\, h\, x_{pa} \\ mg \\ \vdots \\ mg \end{bmatrix}.$$

Assuming that $f(t)$ is constant between samples, the objective function R becomes a sum

$$R = \frac{h}{2} \sum_{i=2}^{n-1} |f_i|^2 + \lambda(x_n - x_b)^2.$$

Using the constraints $x(1) - x_b = 0$ and Eq. (2), define Lagrange function J :

$$J = \frac{h}{2} \sum_{i=2}^{n-1} |f_i|^2 + \lambda(x_n - x_b)^2$$
$$+ < A\,\mathbf{x} - \mathbf{f} - \mathbf{g} \mid \mathbf{y} >,$$

where $\mathbf{y}$ is a Lagrange multiplier and λ is a positive constant.

Necessary Conditions :

$$\frac{DJ}{D\mathbf{y}} = 0 \;\Rightarrow\; A\,\mathbf{x} - \mathbf{f} - \mathbf{g} = 0, \tag{3}$$

$$\frac{DJ}{D\mathbf{x}} = 0 \;\Rightarrow\; B\,\mathbf{x} + A^t\,\mathbf{y} - \mathbf{b} = 0, \tag{4}$$

$$\frac{DJ}{D\mathbf{f}} = 0 \;\Rightarrow\; h\,\mathbf{f} - \tilde{\mathbf{y}} = 0, \tag{5}$$

where

$$B = \begin{bmatrix} \mathbf{0} & \vdots & \mathbf{0} \\ \ldots & & \ldots \\ \mathbf{0} & \vdots & 2\,\lambda \end{bmatrix}, \quad \tilde{\mathbf{y}} = \begin{bmatrix} 0 \\ 0 \\ y_3 \\ \vdots \\ y_n \end{bmatrix}.$$

Substituting (5) into (3),

$$A\,\mathbf{x} - \frac{1}{h}\tilde{\mathbf{y}} - \mathbf{g} = 0,$$
$$B\,\mathbf{x} + A^t\,\mathbf{y} - \mathbf{b} = 0.$$

So, we have the optimality system

$$\begin{bmatrix} A & -\frac{1}{h}\tilde{I}_n \\ B & A^t \end{bmatrix} \cdot \begin{bmatrix} \mathbf{x} \\ \mathbf{y} \end{bmatrix} = \begin{bmatrix} \mathbf{g} \\ \mathbf{b} \end{bmatrix},$$

where

$$\tilde{I}_n = \begin{bmatrix} \mathbf{0} & \mathbf{0} \\ \mathbf{0} & I_{n-2} \end{bmatrix}, \quad I_k = k\text{th unit matrix}.$$

< A Sample for Matlab Program >

File Name : Luxo1.m

```
disp('This program solves an optimal control problem in [0, 1].')



for kkk=1:50

  xa  = input(' The initial state      x(0) = ');
  xpa = input(' The initial velocity  x`(0) = ');
  xb  = input(' The terminal state     x(1) = ');
  n   = input(' The number of grid-points:  n  = ');

  if n >= 300,
    fprintf(' n must be less than or equal to 300'); n=300;
  end
```

```
l1 = 1000;  l2 = 1; % The weights
h = 1/(n-1);
m  = 1;            % The mass of a given particle
g  = 0; % .98;          % The gravitational acceleration

alp = m/h^2;

A = diag(ones(n,1))+diag(-2*ones(n-1,1),-1)+diag(ones(n-2,1),-2);
A(2,1) = -1;  A = alp*A;

B = -(l2/h)*eye(n); B(1,1) = 0; B(2,2) = 0;
C = zeros(n,n); C(n,n) = 2*l1;

M = [A B; C l2*A'];
b = m*g*ones(n,1); b(1) = alp*xa; b(2) = alp*h*xpa;
F = [b; zeros(n-1,1); 2*l1*xb];
```

```
X = M\F;

x = X(1:n);
f = X(n+3:2*n); f = (l2/h)*f;  % f(1)=0; f(2)=0;
plot(1:n,x,'o',1:n,[f;0;0],'*')
xlabel(' position(o) and fuel(*) ')


fprintf(' The mass of the particle       m   = %g \n', m)
fprintf(' The gravitation acceleration   g   = %g \n', g)
fprintf( ' ************  \n')
fprintf(' The disired terminal state    x(1) = %g \n', xb)
fprintf(' The computing terminal state  x(1) = %g \n', x(n))
fprintf(' The error of terminal state   err  = %g \n', abs(x(n)-xb))
fprintf(' The total fuel consumption   sum(f) = %g \n', h*norm(f)^2)
fprintf(' The maximal fuel consumption max(|f|) = %g \n', max(abs(f)))
```

```
  fprintf( ' ************  \n ')

  lll = input('Do you want to test another example? Yes(1), No(0) ');
  if lll == 0, break; end

end
```

< A Sample for Matlab Program >

File Name : Luxo-grad.m

```
xa  = input(' The initial state      x(0) = ');
xpa = input(' The initial velocity  x`(0) = ');
xb  = input(' The terminal state     x(1) = ');
n   = input(' The number of grid-points:  n  = ');

if n >= 300, fprintf(' n must be less than or equal to 300');
   n=300; end

l1 = 1000;  l2 = 1; % The weights
h = 1/(n-1);
m  = 1;           % The mass of a given particle
g  = 0; % .98;         % The gravitational acceleration

alp = m/h^2;
```

```
A = diag(ones(n,1))+diag(-2*ones(n-1,1),-1)+diag(ones(n-2,1),-2);
A(2,1) = -1;  A = alp*A;

b = m*g*ones(n,1); b(1) = alp*xa; b(2) = alp*h*xpa;
f = ones(n,1); f(1)=0; f(2) = 0;   % initial control

tol = 0.001;
itr = 0;
jerr = 100;


while ( jerr >= tol)
   x = A\(f+b);
   b2 = zeros(n,1);  b2(n) = ( 2*l1*xb-2*l1*x(n) )/l2;
   y = A'\b2;  y(1)=0; y(2) = 0;
   d = -(h*f-l2*y);
```

```
   xd = A\(d+b);
   yd = (1/l2)*A'\[zeros(n-1,1);2*l1*(xb-xd(n))]; yd(1)=0; yd(2)=0;
   rho = dot(d,d)/( h*dot(d,d)-l2*dot(yd,d) );
   f = f + rho*d; % (1/h)*d;

%   f = f + 2*h*d; % (1/h)*d;
   jerr = h*norm(d); %[rho jerr]
   itr = itr + 1;
   if itr == n, break; end
end

f = f(3:n);
plot(1:n,x,'o',1:n,[f;0;0],'*')
xlabel(' position(o) and fuel(*) ')

fprintf(' The mass of the particle       m   = %g \n', m)
```

```
fprintf(' The gravitation acceleration   g   = %g \n', g)
fprintf(' The iteration numbers         itr  = %g \n', itr)
fprintf(' The norm of gradient         grad  = %g \n', jerr)
fprintf( ' ************  \n')
fprintf(' The disired terminal state    x(1) = %g \n', xb)
fprintf(' The computing terminal state  x(1) = %g \n', x(n))
fprintf(' The error of terminal state   err  = %g \n', abs(x(n)-xb))
fprintf(' The total fuel consumption   sum(f) = %g \n', h*norm(f)^2)
fprintf(' The maximal fuel consumption max(|f|) = %g \n', max(abs(f)))
fprintf( ' ************  \n ')
```