Otimização de Sistemas

Prof. Sandro Jerônimo de Almeida, PhD.

Programação Não Linear
Otimização Restrita

Programação Não Linear

Minimização Restrita - Lagrange

$$\begin{aligned} &minimizar \quad f(x) \\ &sujeito \ \ a \quad h_i(x) = 0 \quad i = 1, \ldots, m \end{aligned}$$

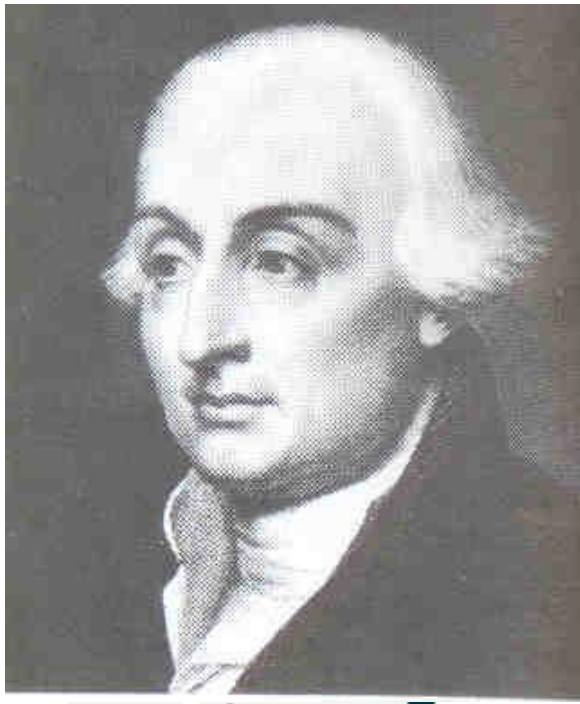
Lagrange mostrou que se x é uma solução do problema, então existem multiplicadores

$$\lambda_0, \lambda_1, \ldots \lambda_m$$

tais que

$$\lambda_0 \nabla f(x) + \sum_{i=1}^{m} \lambda_i \nabla h_i(x) = 0$$

Lagrange



1736-1813

Programação Não Linear

Minimização Restrita - Desigualdades

$$\begin{aligned} &minimizar && f(x) \\ &sujeito\ a && g_i(x) \le 0 \quad i = 1, 2, \ldots, m \end{aligned}$$

Fritz John mostrou em 1948 que se x é uma solução do problema, então existem multiplicadores

$$\lambda_0, \lambda_1, \ldots \lambda_m \ge 0$$

tais que

$$\lambda_0 \nabla f(x) + \sum_{i=1}^{m} \lambda_i \nabla h_i(x) = 0$$

Fritz John

Dificuldade: como garantir $\lambda_0 \neq 0$?

Programação Não Linear

Minimização Restrita - Karush-Kuhn-Tucker

$$\begin{aligned} &minimizar && f(x) \\ &sujeito\ \ a && g_i(x) \le 0 \quad i = 1, 2, \ldots, m \end{aligned}$$

Kuhn e Tucker mostraram em 1951 que se x é uma solução do problema, então existem multiplicadores

$$\lambda_1, \lambda_2, \ldots \lambda_m \ge 0$$

tais que

$$\nabla f(x) + \sum_{i=1}^{m} \lambda_i \nabla h_i(x) = 0$$

desde que seja satisfeita uma certa condição de qualificação.

Kuhn

Programação Não Linear

Otimização Restrita – Exemplos e Referências

- *Método de Lagrange* – Exemplos Numéricos
- https://en.wikipedia.org/wiki/Lagrange_multiplier

- *Condições de Karush-Kuhn-Tucker (KKT)*
- https://en.wikipedia.org/wiki/Karush%E2%80%93Kuhn%E2%80%93Tucker_conditions

Programação Não Linear

Otimização Restrita – Um estudo de caso

- CASE STUDIES IN TRAJECTORY OPTIMIZATION: TRAINS, PLANES, AND OTHER PASTIMES - ROBERT J. VANDERBEI

- https://vanderbei.princeton.edu/tex/trajopt/trajopt.pdf

AMPL: *Train.mod*

```
param N := 201;
param time := 4.8;
param length := 6.0;
param ns := 3;
param z{1..ns-1};
param s{1..ns};
param h := time/N;
param uamax := 10.0;
param ubmax :=  2.0;
param aa:= 0.3;
param bb := 0.14;
param cc := 0.16;
param eps := 0.05;
param pi := 4*atan(1);

var x{0..N};
var v{i in 0..N-1} = (x[i+1]-x[i])/h;
var v_avg{i in 1..N-1}
    = (v[i]+v[i-1])/2;
var a{i in 1..N-1} = (v[i]-v[i-1])/h;
var ua{1..N-1} >=0.0, <=uamax, :=0.0;
var ub{1..N-1} >=0.0, <=ubmax, :=0.0;
var u {i in 1..N-1} = ua[i]-ub[i];

minimize energy:
  sum {i in 1..N-1} ua[i]*v_avg[i]*h;
```

```
s.t. newton {i in 1..N-1}:
  h*a[i] =
  h*
  (
    - sum {j in 1..ns-1}
      (s[j+1]-s[j])*
       atan((x[i]-z[j])/eps)/pi
    - aa - bb*v_avg[i] - cc*v_avg[i]^2
    + u[i]
  );

s.t. x_init: x[0] = 0;
s.t. x_finl: x[N] = length;
s.t. v_init: v[0] = 0;
s.t. v_finl: v[N-1] = 0;

data;
param z := 1 2.0 2 4.0;
param s := 1 2.0 2 0.0 3 -2.0;

solve;

printf {i in 0..N}: "%10f %10f \n",
    i*h, x[i] > train_x;
printf {i in 1..N-1}: "%10f %10f \n",
    i*h, u[i] > train_a;
printf {i in 0..N-1}: "%10f %10f \n",
    i*h, v[i] > train_v;
```