Constrained Optimization

Lecture 01B
fmincon and the Himmelblau Optimization Problem

John Bagterp Jørgensen

Department of Applied Mathematics and Computer Science
Technical University of Denmark

02612 Constrained Optimization

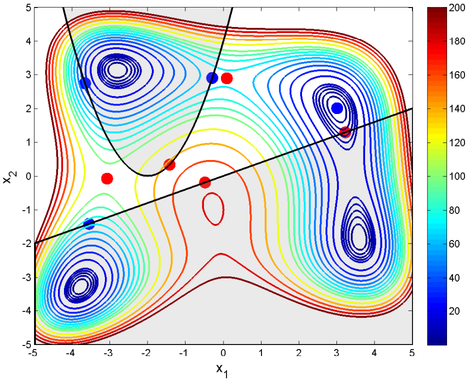
The Himmelblau Optimization Problem

The Himmelblau Optimization Problem

$$\min_{x} \quad f(x) = (x_1^2 + x_2 - 11)^2 + (x_1 + x_2^2 - 7)^2 \tag{1a}$$

$$c_1(x) = (x_1 + 2)^2 - x_2 \geq 0 \tag{1b}$$

$$c_2(x) = -4x_1 + 10x_2 \geq 0 \tag{1c}$$



The Himmelblau Optimization Problem - Gradients

$$\min_{x} \quad f(x) = (x_1^2 + x_2 - 11)^2 + (x_1 + x_2^2 - 7)^2 \tag{2a}$$

$$c_1(x) = (x_1 + 2)^2 - x_2 \geq 0 \tag{2b}$$

$$c_2(x) = -4x_1 + 10x_2 \geq 0 \tag{2c}$$

- Gradient of the objective function

$$\nabla f(x) = \begin{bmatrix} \frac{\partial f}{\partial x_1} \\ \frac{\partial f}{\partial x_2} \end{bmatrix} = \begin{bmatrix} 4(x_1^2 + x_2 - 11)x_1 + 2(x_1 + x_2^2 - 7) \\ 2(x_1^2 + x_2 - 11) + 4(x_1 + x_2^2 - 7)x_2 \end{bmatrix} \tag{3}$$

- Gradients of the constraints

$$\nabla c_1(x) = \begin{bmatrix} \frac{\partial c_1}{\partial x_1} \\ \frac{\partial c_1}{\partial x_2} \end{bmatrix} = \begin{bmatrix} 2(x_1 + 2) \\ -1 \end{bmatrix} \tag{4a}$$

$$\nabla c_2(x) = \begin{bmatrix} \frac{\partial c_2}{\partial x_1} \\ \frac{\partial c_2}{\partial x_2} \end{bmatrix} = \begin{bmatrix} -4 \\ 10 \end{bmatrix} \tag{4b}$$

fmincon

`fmincon` in the Matlab Optimization toolbox

$$
\begin{aligned}
\min_{x\in\mathbb{R}^n} \quad & f(x) && \text{(5a)}\\
s.t. \quad & c(x) \leq 0 && \text{(5b)}\\
& c_{eq}(x) = 0 && \text{(5c)}\\
& Ax \leq b && \text{(5d)}\\
& A_{eq}x = b_{eq} && \text{(5e)}\\
& x_l \leq x \leq x_u && \text{(5f)}
\end{aligned}
$$

- Convert the mathematical optimization problem into this form (note the inequalities)
- Syntax
  `[x,fval,exitflag,output,lambda,grad,hessian] = fmincon(fun,x0,A,b,Aeq,beq,lb,ub,nonlcon,options)`
- doc fmincon

Examples

Version 1
Specification of objective functions and constraints

- Objective function, $f(x)$.

```
function f = objfunHimmelblau(x,p)

tmp1 = x(1)*x(1)+x(2)-11;
tmp2 = x(1)+x(2)*x(2)-7;
f = tmp1*tmp1 + tmp2*tmp2;
```

- Constraint functions, $c(x) \leq 0$, $c_{eq}(x) = 0$.

```
function [c,ceq] = confunHimmelblau(x,p)

c   = zeros(2,1);
ceq = zeros(0,1);

% Inequality constraints c(x) <= 0
tmp = x(1)+2;
c(1,1) = -(tmp*tmp - x(2));
c(2,1) = -(-4*x(1) + 10*x(2));
```

Version 1
Driver for `fmincon`

```
x0 = [0;0];      % initial point
xl = [-5;-5];    % lower bounds
xu = [5;5];      % upper bounds

% First we pretend that there is no linear equality/inequality constraints
A   = zeros(0,2);
b   = zeros(0,1);
Aeq = zeros(0,2);
beq = zeros(0,1);

% Parameters (in this case we do not use parameters)
p = [];

% Call fmincon
options = optimoptions( 'fmincon',...
                        'Display','none',...
                        'Algorithm','interior-point');

[x,fval,exitflag,output]=fmincon(...
                            @objfunHimmelblau, x0, ...
                            A, b, Aeq, beq, ...
                            xl, xu, ...
                            @confunHimmelblau, ...
                            options, ...
                            p);

x,fval,output
```

Version 2 - Specification of Gradients
Specification of objective function

- Objective function, $f(x)$, and its gradient, $\nabla f(x)$

```
function [f,dfdx] = objfungradHimmelblau(x,p)

tmp1 = x(1)*x(1)+x(2)-11;
tmp2 = x(1)+x(2)*x(2)-7;
f = tmp1*tmp1 + tmp2*tmp2;

% compute the gradient of f
if nargout > 1
    dfdx = zeros(2,1);
    dfdx(1,1) = 4*tmp1*x(1) + 2*tmp2;
    dfdx(2,1) = 2*tmp1 + 4*tmp2*x(2);
end
```

Version 2 - Specification of Gradients
Specification of constraint function

- Constraints, $c(x) \leq 0$ and $c_{eq}(x) = 0$, and their gradients, $\nabla c(x)$ and $\nabla c_{eq}(x)$

```
function [c,ceq,dcdx,dceqdx] = confungradHimmelblau(x,p)

c   = zeros(2,1);
ceq = zeros(0,1);

% Inequality constraints c(x) <= 0
tmp = x(1)+2;
c(1,1) = -(tmp*tmp - x(2));
c(2,1) = -(-4*x(1) + 10*x(2));

% Compute constraint gradients
if nargout > 2
    dcdx = zeros(2,2);
    dceqdx   = zeros(2,0);

    dcdx(1,1) = -2*tmp; % dc1dx1
    dcdx(2,1) = 1.0;    % dc1dx2
    dcdx(1,2) = 4.0;    % dc2dx1
    dcdx(2,2) = -10;    % dc2dx2
end
```

Version 2 - Specification of Gradients
Driver for `fmincon`

```
x0 = [0;0];      % initial point
xl = [-5;-5];    % lower bounds
xu = [5;5];      % upper bounds

% First we pretend that there is no linear equality/inequality constraints
A   = zeros(0,2);
b   = zeros(0,1);
Aeq = zeros(0,2);
beq = zeros(0,1);

% Parameters (in this case we do not use parameters)
p = [];

% Call fmincon
options = optimoptions( 'fmincon',...
                        'SpecifyObjectiveGradient',true,...
                        'SpecifyConstraintGradient',true,...
                        'Display','none',...
                        'Algorithm','interior-point');

[x,fval,exitflag,output]=fmincon(...
                            @objfungradHimmelblau, x0, ...
                            A, b, Aeq, beq, ...
                            xl, xu, ...
                            @confungradHimmelblau, ...
                            options, ...
                            p);

x,fval,output
```

Version 3 - Nonlinear / linear constraints
Objective function

Objective function

$$f(x) = (x_1^2 + x_2 - 11)^2 + (x_1 + x_2^2 - 7)^2 \tag{6}$$

Matlab specification

```
function [f,dfdx] = objfungradHimmelblau(x,p)

tmp1 = x(1)*x(1)+x(2)-11;
tmp2 = x(1)+x(2)*x(2)-7;
f = tmp1*tmp1 + tmp2*tmp2;

% compute the gradient of f
if nargout > 1
    dfdx = zeros(2,1);
    dfdx(1,1) = 4*tmp1*x(1) + 2*tmp2;
    dfdx(2,1) = 2*tmp1 + 4*tmp2*x(2);
end
```

Version 3 - Nonlinear / linear constraints
Nonlinear constraint function

- Nonlinear constraint function

$$c_1(x) = (x_1 + 2)^2 - x_2 \geq 0 \quad (-c_1(x) \leq 0) \tag{7}$$

- Linear constraint function

$$c_2(x) = -4x_1 + 10x_2 \geq 0 \quad (-c_2(x) \leq 0) \tag{8}$$

Specification of the nonlinear constraint function in Matlab

```
function [c,ceq,dcdx,dceqdx] = confungradHimmelblau1(x,p)

c   = zeros(1,1);
ceq = zeros(0,1);

% Inequality constraints c(x) <= 0
tmp = x(1)+2;
c(1,1) = -(tmp*tmp - x(2));

% Compute constraint gradients
if nargout > 2
    dcdx   = zeros(2,1);
    dceqdx = zeros(2,0);

    dcdx(1,1) = -2*tmp; % dc1dx1
    dcdx(2,1) = 1.0;    % dc1dx2
end
```

Version 3 - Nonlinear / linear constraints
Driver for `fmincon`

```
x0 = [0;0];      % initial point
xl = [-5;-5];    % lower bounds
xu = [5;5];      % upper bounds

% -4x1 + 10 x2 >= 0 represented as A x <= b
A   = [4 -10];
b   = 0;
Aeq = zeros(0,2);
beq = zeros(0,1);

% Parameters (in this case we do not use parameters)
p = [];

% Call fmincon
options = optimoptions( 'fmincon',...
                        'SpecifyObjectiveGradient',true,...
                        'SpecifyConstraintGradient',true,...
                        'Display','none',...
                        'Algorithm','interior-point');

[x,fval,exitflag,output]=fmincon(...
                            @objfungradHimmelblau, x0, ...
                            A, b, Aeq, beq, ...
                            xl, xu, ...
                            @confungradHimmelblau1, ...
                            options, ...
                            p);

x,fval,output
```