Development of Algorithm for Non-Linear Optimization Using Sequential Quadratic Programming (SQP) Method

Dr.Suidaa Hassan Bringi Kafi
Department of Mathematics, Faculty of Education, University of Kordofan
Dr. Mohsin H.A. Hashim
Department of Mathematics, Faculty of Mathematical Science, University of Khartoum

مجلة
كلية التربية
جامعة الخرطوم
العدد السادس عشر
السنة الثانية عشرة
سبتمبر 2020م

Development of Algorithm for Non-Linear Optimization Using Sequential Quadratic Programming (SQP) Method

Suidaa Hassan Bringi Kafi
Department of Mathematics, Faculty of Education, University of Kordofan
Dr. Mohsin H.A. Hashim
Department of Mathematics, Faculty of Mathematical Science, University of Khartoum

Abstract.
This paper describes a development of an algorithm for nonlinear optimization using sequential quadratic programming (SQP) method. The computation is performed through a number of steps. The improvement of the performance is achieved and compared to a version with line search.

Key words:
Sequential Quadratic programming (SQP), constraint linearization, BFGS, optimization, approximation, Algorithm, line search, global convergence.

مستخلص

هدفت هذه الورقة لتوضيح خطوات تطور خوارزمية إستناداً على الطريقة التقريبة للمعادلات التربيعية (SQP)حيت تم إنشاء خوارزمية لحساب القيم التفريبية لدالة الهدف وفق قيود محددة. ثم تم تنفيذ هذا البرامج لحل نماذج مختلفة من المعادلات الخطية والغير خطية.

1.Introduction:

Sequential quadratic programming or (SQP) in abbreviated form, is an iterative method for constrained nonlinear optimization. SQP methods used in mathematical problems for which the objective function and the constraints are twice continuously differentiable, are used to solve a sequence of optimization sub problems, each of which optimizes a quadratic model of the objective subject to a linearization of the constraints.

These methods belonging to the most powerful optimization algorithms, for solving differentiable nonlinear problems of the form (1) and (2):

$$minimize\ f(x) = \frac{1}{2}x^T Gx + q^T x\text{q} \quad (1)$$

$$minimize\ \nabla f(x)^T d_k + \frac{1}{2}{d_k}^T B_k d_k \quad (2)$$

The theoretical background is described for example in Stoer [12], and an excellent review is given by Boggs and Tolle [2]. SQP methods are also introduced in the books of Papalambros and Wilde [7] and Edgar and Himmelblau [3], among many others. Their excellent numerical performance has been tested and compared with other methods, and for many years they belong to the set of most frequently used algorithms for solving practical optimization problems.

A first idea of sequential quadratic programming or (SQP),

has been investigated in the Ph.D. thesis of Wilson [13]. Sequential quadratic programming methods became popular during the late seventies due to papers of Han [5,6] and Powell [9,10]. Their superiority over other optimization methods known at that time, was shown by Schittkowski [12]. Since then many modifications and extensions have been published on SQP methods. Nice review papers are given by Boggs and Tolle [2] and Gould and Toint [4]. As the presentation of even a selected overview is impossible due to the limited space here, we concentrate on a few important facts and highlights from a personal view without any attempt to be complete.

In this paper we describe mathematical models used for an Algorithm development, and some application of our Algorithm.

2. Methods

2.1 Mathematical Models Used for Algorithm Development

With respect to the general QP problems, the main model problem to be solve is:

$$minimize\ f(x) = \tfrac{1}{2} x^T G x + q^T x q \tag{3}$$

The basic idea of our project is to formulate a Matlab algorithm to solve a quadratic programming problem in each iteration which is obtained by

linearization of the constraints and approximating the lagrangian function $\mathrm{L(x,u)}$ quadratically. Starting from any $x_0 \in R^n$, suppose that $x_k \in R^m$ is an actual approximation of the solution y_k an approximation of the multipliers, and $B_k \in R^{n\times n}$ an approximation of the Hessian of the $\mathcal{L}$agrangian function, $k = 0,1,2,\dots,n$ then, a quadratic program(QP) of the forms

$$minimize\ \nabla f(x)^T d_k + \tfrac{1}{2} {d_k}^T B_k d_k,$$

$d_k \in R^n: \nabla g(x_k)^T d_k + g(x_k) \geq 0$, is formulated and solved in each iteration.

It is considered the nonlinear, constrained optimization problem to minimize an objective function *f* under *m* nonlinear inequality constraints,

$$minimize\ f(x), over\ x \in R^n\ , g(x) \geq 0,$$

(4)

where *x* is an *n*-dimensional parameter vector and $g(x) = (g_1(x), g_2(x), \dots g_m(x))^T$.

We assumed that the objective function $f(x)$and *m* constraint functions $g_1(x), g_2(x), \dots g_m(x)$, are continuously differentiable on the whole R^n.

In general, the unconstrained optimization problems are described as follows

$$min_{x \in R^n} f(x)$$

(5)

whereR^n is an n −dimensional Euclidean space and $f: R^n \to R$ is assumed to be continuously twice differentiable. The gradient and Hessian for (5) are denoted as g and G, respectively. In order to display the updated formula of BFGS, the step-vector, and y_k are defined as:

as:

$$d_k = x_{k+1} - x_k$$

(6)

$$y_k = g(x_{k+1}) - g(x_k)$$

$$= g_{k+1} - g_k$$

The search direction $\mathrm{p_k}$ at stage k is given by the solution of the analogue of the Newton equation

$$B_k d_k = -\nabla f(x_k)$$

where B_k is an approximation to the Hessian matrix, which is updated iteratively at each stage, and $\nabla f(x_k)$ is the gradient of the function evaluated at x_k . A line search d_k in the direction p_k is then used to find the next point x_{k+1} by minimizing $f(x_k + \alpha p_k)$ over the scalar.

The quasi-Newton condition imposed on the update of B_k is:

$$B_{k+1}(x_{k+1} - x_k) = \nabla f(x_{k+1}) - \nabla f(x_k)$$

(7)

Let

$$y_k = \nabla f(x_{k+1}) - \nabla f(x_k)$$

and

$$d_k = x_{k+1} - x_k$$

then B_{k+1} satisfies

$$B_{k+1} d_k = y_k$$

which is the secant equation. The curvature condition ${d_k}^T y_k > 0$ should be satisfied. If the function is not strongly convex, then the condition has to be enforced explicitly.

Instead of requiring the full Hessian matrix at the point x_{k+1} to be computed as B_{k+1}, the approximate Hessian at stage k is updated by the addition of two matrices:

$$B_{k+1} = B_k + H_k + T_k$$

(8)

Both H_k and T_k are symmetric rank-one matrices, but their sum is a rank-two update matrix. In order to maintain the symmetry and positive definitiveness of B_{k+1}, the update form can be chosen as:

$$B_{k+1} = B_k + \alpha h h^T + \beta t t^T$$

(9)

Imposing the secant condition,

$$B_{k+1}d_k = y_k$$

Choosing $\mathrm{h} = \mathrm{y_k}$ and $\mathrm{t} = \mathrm{d_k}$, we can obtain:

$$\alpha = \frac{1}{y_k^T d_k}$$

(10)

$$\beta = -\frac{1}{d_k^T B_k d_k}$$

(11)

Finally, we substitute α and β into $\mathrm{B_{k+1}} = \mathrm{B_k} + \alpha \mathrm{hh^T} + \beta \mathrm{tt^T}$ and get the update equation of B_{k+1} :

$$B_{k+1} = B_k + \frac{y_k y_k^T}{y_k^T d_k} - \frac{B_k\, d_k d_k^T B_k^T}{d_k^T B_k d_k}$$

(12)

We have the following optimization algorithm using SQP method to build a MATLAB function, for solving the equation in (3), the computing procedure of the SQP method is descript,

From an initial guess $\mathrm{x_0}$ and an approximate Hessian matrix B_0 the following steps are repeated as x_k converges to the solution:

Obtain a direction $\mathrm{d_k}$ by solving

$$\mathrm{B_k d_k} = -\nabla \mathrm{f(x_k)}$$

So the linear search $\mathrm{d_k}$ is define by:

$$d_k = -\mathrm{B_k^T} \nabla \mathrm{f}(x_k)$$

(13)

Perform a one-dimensional optimization line search to find an acceptable step size α_k in the direction found in the first step, so

$$\alpha_k = argminf(x_k + \alpha d_k)$$

Set $d_k = \alpha_k p_k$ and update the personal best

$$x_{k+1} = x_k$$

And update the global best as:

$$x_{k+1} = x_k d_k^T$$

The Hessian approximation B using (BFGS) update,

$$x_{k+1} = x_k d_k^T$$

$$y_k = \nabla f(x_{k+1}) - \nabla f(x_k)$$

$$B_{k+1} = B_k + \frac{y_k y_k^T}{d_k^T y_k} - \frac{B_k \ d_k d_k^T B_k^T}{d_k^T B_k d_k}$$

(14)

2.2 Algorithm

Let$x^{(0)} = 0, B^{(0)} = 0, y^{(0)} = 0, d^{(0)} = 0,$ be given.

Start: $\alpha_0 = 0$

For $i = 1,2,\dots,n\ population\ do:$

Compute:

1)$g(x_k) - g'(x_k)$

2) $\mathrm{d_k} = -\mathrm{B_k^T}\nabla \mathrm{f}(\mathrm{x_k})$;

3)$minf(x) = gf(x_k)'d + 0.5d_k' B_k;$

If$(x_{k+1}) = (x_k); y_k = g(x_{k+1}) - g(x_k),$

1) then update;

i) $x_{k+1} = x_k + \alpha d'$;

ii)$B_{k+1} = B_k + \frac{y_k y_k^T}{d_k^T y_k} - \frac{B_k d_k d_k^T B_k^T}{d_k^T B_k d_k}$

2) Compute:

$$minf(x) = gf(x_k)'d + 0.5d_k' B_k \quad .$$

3) Update personal best

4) For it=1:MaxIt and for i=1:nPop,

Compute:

i)$x_{k+1} := x_k + \alpha d'$;

ii) $y_k = g(x_{k+1}) - g(x_k)$;

iii) $B_{k+1} = B_k + \frac{y_k y_k^T}{d_k^T y_k} - \frac{B_k d_k d_k^T B_k^T}{d_k^T B_k d_k}$;

$\alpha = \frac{1}{y_k' d_k}$;

$$g(x_k) = gf'^{(x_k)};$$

$dx_{k+1} := x_k$

Compute

i) $g(x_{k+1}) = g(x_k)$;

ii) $d_k = -B_k' gf(x_k)$

iii) $minf(x_k) = gf(x_k)'d + 0.5 d_k' B_k$

Update personal best

$x_{k+1} := x_k$

$$minf(x_{k+1}) = minf(x_k)$$

Update Glob Best

If $x_{k+1} := x_k d'$

Compute: $minf(x_{k+1})$;

If $minf(x_{k+1}) < minf(x_k)/(\nabla g(x_{k+1}) + \nabla g(x_k)', d > 0)$;

then stop.

2.3 Algorithm Validation and Discussion

We formulated function with name of (SQP,m) using matlab and applied it in some chosen problems with deferent Iterations, using the sup Algorithm blow:

SQPProblem(i) .m

```
clc;
clear;
close all;
```

```
%% Problem Definition
problem.GradCostFunction = @(x) GCFProb(i)(x);    % Gradient
of Cost Function
problem.ConstFunction    = @(x) CFProb(i)(x);     %
Constrained Function
problem.nVar=;              % Number of Unknown (Decision)
Variables
problem.VarMin =;               % Lower Bound of Decision
Variables
problem.VarMax =;               % Upper Bound of Decision
Variables
%% Parameters of SQP
params.MaxIt = ;       % Maximum Number of Iterations
params.nPop = ;         % Population Size (SQP Size)
params.ShowIterInfo = true;  % Flag for Showing Iteration
information
%% Calling SQP
out=SQP(problem, params);
BestSol=out.BestSol;
BestCosts=out.BestCosts;
%% Results
figure;
plot(BestCosts, 'LineWidth',2);
xlabel('Iteration');
ylabel('Best Cost');
grid on;
figure;
semilogy(BestCosts, 'LineWidth',2);
xlabel('Iteration');
ylabel('Best Cost');
```

```
grid on;
```

2.4:The Chosen Problems:

In this sub section we applied our algorithm in ten chosen problems with deferent Iterations.

Problem1

Consider using SQP method to solve the problem

$$min - x_1 x_2 x_3 \quad \backslash \textbf{(1)}$$

Subject to

$$72 - x_1 - 2x_2 - 2x_3 \quad \textbf{(2)}$$

(R. Fletcher**)**

[11]

Solution:

The unknown decisions are 3, the lower bound is -10 and the upper one is 10, the maximum number of iterations is 100, and the SQP size is 50.

The gradient of constrained Function:

```
function g=GCFProb1(x)
g(1)=-x(2)*x(3);
g(2)=-x(1)*x(3);
g(3)=-x(1)*x(2);
g=g';
end
```

The Constrained Function:

```
function f=CFProb1(x)
f=72-x(1)-2*x(2)-2*x(3);
end
```

SQPProblem1.m:

```
clc;
clear;
```

```
close all;
%% Problem Definition
problem.GradCostFunction = @(x) GCFProb1(x);
  % Gradient of Cost Function
problem.ConstFunction    = @(x) CFProb1(x);
 % Constrained Function
problem.nVar=3;                 % Number of Unknown
(Decision) Variables
problem.VarMin =-10;              % Lower Bound of Decision
Variables
problem. VarMax =10;              % Upper Bound of Decision
Variables
%% Parameters of SQP
params.MaxIt = 100;       % Maximum Number of Iterations
params.nPop =50;          % Population Size (SQP Size)
params.ShowIterInfo = true;  % Flag for Showing Iteration
information
%% Calling SQP
out=SQP(problem, params);
BestSol=out.BestSol;
BestCosts=out.BestCosts;
%% Results
figure;
plot(BestCosts, 'LineWidth',2);
xlabel('Iteration');
ylabel('Best Cost');
grid on;
figure;
semilogy(BestCosts, 'LineWidth',2);
xlabel('Iteration');
```

```
ylabel('Best Cost');
grid on;
```

Problem 2

Consider using SQP method to solve the problem

$$minimize \quad exp(x_1x_2x_3x_4x_5) \quad \textbf{(3)}$$

Subject to

$$x_1^2 + x_2^2 + x_3^2 + x_4^2 + x_5^2 = 10 \quad \textbf{(4)}$$

$$x_2x_3 = 5x_4x_5 \quad \textbf{(5)}$$

$$x_1^3 + x_2^3 = -1 \quad \textbf{(6)}$$

(R. Fletcher**) [11]**

The Solution:

The unknown decisions are 5, the lower bound is -10 and the upper one is 10, the maximum number of iterations is 100, and the SQP size is 10.

The gradient of constrained Function

```
function f=GCFPob2(x)
g=[(x(2)*x(3)*x(4)*x(5))*exp(x(1)*x(2)*x(3)*x(4)*x(5))
(x(1)*x(3)*x(4)*x(5))*exp(x(1)*x(2)*x(3)*x(4)*x(5))
(x(1)*x(2)*x(4)*x(5))*exp(x(1)*x(2)*x(3)*x(4)*x(5))
(x(1)*x(2)*x(3)*x(5))*exp(x(1)*x(2)*x(3)*x(4)*x(5))
(x(1)*x(2)*x(3)*x(4))*exp(x(1)*x(2)*x(3)*x(4)*x(5))];
f=g;
end
```

The Constrained Function

```
function f=CFProb2(x)
g(1)=x(1)^2+x(2)^2+x(3)^2+x(4)^2+x(5)^2-10;
g(2)=(x(1)*x(2))-(5*(x(4)*x(5)));
  g(3)=(x(1)^2+x(2)^2)+1;
  f=[g(1);g(2);g(3)];
end
```

SQP Problem 2 .m:

```
clc;
clear;
close all;
%% Problem Definition
problem.GradCostFunction = @(x) GCFProb2(x);    % Gradient
of Cost Function
problem.ConstFunction = @(x) CFProb2(x);   % Constrained
Function
problem.nVar=5;                 % Number of Unknown
(Decision) Variables
problem.VarMin =-10;                % Lower Bound of Decision
Variables
problem. VarMax =10;                % Upper Bound of Decision
Variables
%% Parameters of SQP
params.MaxIt = 100;       % Maximum Number of Iterations
params.nPop =10;          % Population Size (SQP Size)
params.ShowIterInfo = true;  % Flag for Showing Iteration
information
%% Calling SQP
out=SQP(problem, params);
```

```
BestSol=out.BestSol;
BestCosts=out.BestCosts;
%% Results
figure;
plot(BestCosts, 'LineWidth',2);
xlabel('Iteration');
ylabel('Best Cost');
grid on;
figure;
semilogy(BestCosts, 'LineWidth',2);
xlabel('Iteration');
ylabel('Best Cost');
grid on;
```

Problem 3

Consider using SQP method to solve the problem

$$minimize\ x_1 x_2 \quad x \in R^4 \qquad (7)$$

Subject to

$$\frac{(x_1 x_3 + x_4 x_5)^2}{(x_1^2 + x_2^2)} - x_3^2 - x_4^2 \qquad (8)$$

$$x_1 \geq x_3 + 1 \qquad (9)$$

$$x_2 \geq x_4 + 1 \qquad (10)$$

$$x_3 \geq x_4 \qquad (11)$$

$$x_4 \geq 1 \qquad (12)$$

(R. Fletcher**) [11]**

The Solution

The unknown decisions are 4, the lower bound is -10 and the upper one is 10, the maximum number of iterations is 100, and the SQP size is 50.

The gradient of constrained Function

```
function g=GCFProb3(x)
g=[x(2)
   x(1)
   0
   0];
End
```

The Constrained Function

```
function f=CFProb3(x)
g1=((x(1)*x(3)+x(2)*x(4))^2/(x(1)^2+x(2)^2))-x(3)^2-x(4)^2+1;
g2=x(1)-x(3)-1;
g3=x(2)-x(4)-1;
g4=x(3)-x(4);
g5=x(4)-1;
f=[g1;g2;g3;g4;g5];
end
```

SQPProblem3 :

```
clc;
clear;
close all;
```

```
%% Problem Definition
problem.GradCostFunction = @(x) GCFProb3(x);    % Gradent
of Cost Function
problem.ConstFunction    = @(x) CFProb3(x);     % Constrained
Function
problem.nVar=4;                  % Number of Unknown
(Decision) Variables
problem.VarMin =-10;               % Lower Bound of Decision
Variables
problem.VarMax =10;                % Upper Bound of Decision
Variables
%% Parameters of SQP
params.MaxIt = 100;       % Maximum Number of Itreations
params.nPop =50;          % Population Size (SQP Size)
params.ShowIterInfo = true;  % Flag for Showing Iteration
information
%% Calling SQP
out=SQP(problem, params);
BestSol=out.BestSol;
BestCosts=out.BestCosts;
%% Results
figure;
plot(BestCosts, 'LineWidth',2);
xlabel('Iteration');
ylabel('Best Cost');
grid on;
figure;
semilogy(BestCosts, 'LineWidth',2);
xlabel('Iteration');
ylabel('Best Cost');
```

grid on;

Problem 4

Consider the geometric programming using SQP method to solve the problem

$$minimize\ x_1^{-1}x_2^{-1}x_3^{-1} \quad (13)$$

$$x_1 + 2x_2 - 2x_3 \le 72 \quad (14)$$

$$x > 1 \quad (15)$$

(R. Fetcher) **[11]**

The Solution

The unknown decisions are 3, the lower bound is -10 and the upper one is 10, the maximum number of iterations is 100, and the SQP size is 10.

The gradient of constrained Function

```
function f=GCFProb4(x)
g=[-x(1)^-2*x(2)^-1*x(3)^-1
   -x(1)^-1*x(2)^-2*x(3)^-1
   -x(1)^-1*x(2)^-1*x(3)^-2];
f=g;
end
```

The Constrained Function

```
function f=CFProb4(x)
f=-x(1)-2*x(2)+2*x(3)+72;
end
```

SQPProblem4 .m

```
clc;
clear;
```

```
close all;
%% Problem Definition
problem.GradCostFunction = @(x) GCFProb4(x);    % Gradent
of Cost Function
problem.ConstFunction    = @(x) CFProb4(x);     % Constrained
Function
problem.nVar=3;                  % Number of Unknown
(Decision) Variables
problem.VarMin =-10;               % Lower Bound of Decision
Variables
problem.VarMax =10;                % Upper Bound of Decision
Variables
%% Parameters of SQP
params.MaxIt = 100;       % Maximum Number of Itreations
params.nPop =10;          % Population Size (SQP Size)
params.ShowIterInfo = true;  % Flag for Showing Iteration
information
%% Calling SQP
out=SQP(problem, params);
BestSol=out.BestSol;
BestCosts=out.BestCosts;
%% Results
figure;
plot(BestCosts, 'LineWidth',2);
xlabel('Iteration');
ylabel('Best Cost');
grid on;
figure;
semilogy(BestCosts, 'LineWidth',2);
xlabel('Iteration');
```

```
ylabel('Best Cost');
grid on;
```

Problem 5

Consider the geometric programming using SQP method to solve the problem

$$minimize \quad f(x) = 40x_1x_2 + 20x_2x_3 \quad \textbf{(16)}$$

Subject to

$$\frac{1}{5}x_1^{-1}x_2^{-\frac{1}{2}} + \frac{3}{5}x_2^{-1}x_3^{-\frac{2}{3}} \leq 1 \quad , x > 0 \quad \textbf{(17)}$$

(R. Fletcher**) [11]**

The Solution:

The unknown decisions are 3, the lower bound is -1 and the upper one is 1, the maximum number of iterations is 100, and the SQP size is 50.

The gradient of constrained Function

```
function g=GCFProb5(x)
g=[40*x(2);40*x(1)+20*x(3);20*x(2)];
end
```

Constrained Function

```
function f=CFProb5(x)
f=1/(5*x(1)*x(2)^(0.5))+3/(5*x(2)*x(3)^(2/3));
end
```

SQPProblem5 .m

```
clc;
clear;
close all;
%% Problem Definition
```

```
problem.GradCostFunction = @(x) GCFProb5(x);    % Gradent
of Cost Function
problem.ConstFunction    = @(x) CFProb5(x);     % Constrained
Function
problem.nVar=3;                  % Number of Unknown
(Decision) Variables
problem.VarMin =-1;               % Lower Bound of Decision
Variables
problem.VarMax =1;                % Upper Bound of Decision
Variables
%% Parameters of SQP
params.MaxIt = 100;       % Maximum Number of Itreations
params.nPop =50;          % Population Size (SQP Size)
params.ShowIterInfo = true;  % Flag for Showing Iteration
information
%% Calling SQP
out=SQP(problem, params);
BestSol=out.BestSol;
BestCosts=out.BestCosts;
%% Results
figure;
plot(BestCosts, 'LineWidth',2);
xlabel('Iteration');
ylabel('Best Cost');
grid on;
figure;
semilogy(BestCosts, 'LineWidth',2);
xlabel('Iteration');
ylabel('Best Cost');
grid on;
```

Problem 6 : Find the optimum to the using problem

$$min\ f(x) = x_1^4 - 2x_2x_1^{2\cdot} + x_2^2 + x_1^2x_2 + 9 = 0 \quad \textbf{(18)}$$

Subject to

$$g(x) = -([\![x_1 + 0.25)]\!]^2 + 0.75x_2 \geq 0 \quad \textbf{(19)}$$

(Optimization)

[1]

The Solution:

The unknown decisions are 3, the lower bound is -10 and the upper one is 10, the maximum number of iterations is 100, and the SQP size is 50.

The gradient of constrained Function

```
function f=GCFProb6(x)
g=[4*x(1)^3-4*(x(2)*x(1))+2*x(1)
   -2*x(1)^2+2*x(2)];
f=g;
end
```

Constrained Function

```
function f=CFProb6(x)
f=-(x(1)+0.25)^2+0.75*x(2);
end
```

SQP Problem6 .m

```
clc;
clear;
close all;
%% Problem Definition
 problem.GradCostFunction = @(x) GCFProb6(x);    % Gradent
of Cost Function
problem.ConstFunction    = @(x) CFProb6(x);     % Constrained
```

```
Function
problem.nVar=2;          .  % Number of Unknown (Decision)
Variables
problem.VarMin =-10;              % Lower Bound of Decision
Variables
problem.VarMax =10;               % Upper Bound of Decision
Variables
%% Parameters of SQP
params.MaxIt = 50;       % Maximum Number of Itreations
params.nPop =100;          % Population Size (SQP Size)
params.ShowIterInfo = true;  % Flag for Showing Iteration
information
%% Calling SQP
out=SQP(problem, params);
BestSol=out.BestSol;
BestCosts=out.BestCosts;
%% Results
figure;
plot(BestCosts, 'LineWidth',2);
xlabel('Iteration');
ylabel('Best Cost');
grid on;
figure;
semilogy(BestCosts, 'LineWidth',2);
xlabel('Iteration');
ylabel('Best Cost');
grid on;
```

Problem 7

Find the optimum to the using problem

$$min\ f(x) = x_1^{-1} x_2^{-1} x_3^{-1} \quad \textbf{(20)}$$

Subject to

$$g(x) = x_1 + 2x_2 - 2x_2 \leq 75 \quad \textbf{(21)}$$

(R. Fletcher) [11]

The Solution

The unknown decisions are 3, the lower bound is -10 and the upper one is 10, the maximum number of iterations is 50, and the SQP size is 100.

The gradient of constrained Function

```
function f=GCFProb7(x)
g=[-x(1)^-2*x(2)^-1*x(3)^-1
   -x(1)^-1*x(2)^-2*x(3)^-1
   -x(1)^-1*x(2)^-1*x(3)^-2];
f=g;
end
```

Constrained Function

```
function f=CFProb7(x)
f=-x(1)-2*x(2)+2*x(3)+72;
end
```

SQPProblem7.m

```
clc;
clear;
close all;
%% Problem Definition
 problem.GradCostFunction = @(x) GCFProb7(x);    % Gradent of Cost Function
problem.ConstFunction    = @(x) CFProb7(x);     % Constrained Function
problem.nVar=3;                % Number of Unknown (Decision) Variables
problem.VarMin =-10;           % Lower Bound of Decision
```

```
Variables
problem.VarMax =10;            % Upper Bound of Decision
Variables
%% Parameters of SQP
params.MaxIt = 100;       % Maximum Number of Itreations
params.nPop =10;          % Population Size (SQP Size)
params.ShowIterInfo = true;  % Flag for Showing Iteration
information
%% Calling SQP
out=SQP(problem, params);
BestSol=out.BestSol;
BestCosts=out.BestCosts;
%% Results
figure;
plot(BestCosts, 'LineWidth',2);
xlabel('Iteration');
ylabel('Best Cost');
grid on;
figure;
semilogy(BestCosts, 'LineWidth',2);
xlabel('Iteration');
ylabel('Best Cost');
grid on;
```

Problem 8

Find the optimum to the using problem

$$min\ f(x) = 2x_1^2 + 4x_2^2 \quad \textbf{(22)}$$

S.t to

$$g(x) = 3x_1 + 2x_2 \geq 12 \quad \textbf{(23)}$$

(Optimization**)** [1]

The Solution

The unknown decisions are 2, the lower bound is -1 and the upper one is 1, the maximum number of iterations is 10, and the SQP size is 5.

The gradient of constrained Function

```
function g=GCFProb8(x)
g=[4*x(1)
   8*x(2)];
end
```

Constrained Function

```
function f=CFProb8(x)
f=3*x(1)+2*x(2)-12;
end
```

SQPProblem8 .m

```
clc;
clear;
close all;
%% Problem Definition
problem.GradCostFunction = @(x) GCFProb8(x);    % Gradient
of Cost Function
problem.ConstFunction    = @(x) CFProb8(x);     % Constrained
Function
problem.nVar=2;           % Number of Unknown (Decision)
Variables
problem.VarMin =-1;              % Lower Bound of Decision
Variables
problem.VarMax =1;               % Upper Bound of Decision
Variables
```

```
%% Parameters of SQP
params.MaxIt = 10;       % Maximum Number of Iterations
params.nPop =5;          % Population Size (SQP Size)
params.ShowIterInfo = true;  % Flag for Showing Iteration
information
%% Calling SQP
out=SQP(problem, params);
BestSol=out.BestSol;
BestCosts=out.BestCosts;
%% Results
figure;
plot(BestCosts, 'LineWidth',2);
xlabel('Iteration');
ylabel('Best Cost');
grid on;
figure;
semilogy(BestCosts, 'LineWidth',2);
xlabel('Iteration');
ylabel('Best Cost');
grid on;
```

Problem 9

Find the optimum to the using problem

$$min\, f(x) = x_1^2 + x_2^{2\cdot} \quad \textbf{(24)}$$

S.t to

$$g_1(x) = x_1^2 + x_2^2 - 9 \le 0 \quad \textbf{(25)}$$

$$g_2(x) = x_1^{\square} + x_2^{\square} - 1 \le 0 \quad \textbf{(26)}$$

(Optimization) [1]

The Solution

The unknown decisions are 2, the lower bound is -10 and the upper one is 10, the maximum number of iterations is 10, and the

SQP size is 50.
The gradient of constrained Function

```
function g=GCFProb9(x)
g=[2*x(1)
   2*x(2)];
end
```

Constrained Function

```
function f=CFProb9(x)
g1=-x(1)^2-x(2)^2+9;
g2=-x(1)-x(2)+1;
f=[g1;g2];
end
```

SQPProblem9 .m

```
clc;
clear;
close all;
%% Problem Definition
problem.GradCostFunction = @(x) GCFProb9(x);    % Gradient of Cost Function
problem.ConstFunction    = @(x) CFProb9(x);     % Constrained Function
problem.nVar=2;              % Number of Unknown (Decision) Variables
problem.VarMin =-10;                % Lower Bound of Decision Variables
problem.VarMax =10;                 % Upper Bound of Decision Variables
%% Parameters of SQP
```

```
paams.MaxIt = 10;       % Maximum Number of Iterations
params.nPop =50;          % Population Size (SQP Size)
params.ShowIterInfo = true;  % Flag for Showing Iteration
information
%% Calling SQP
out=SQP(problem, params);
BestSol=out.BestSol;
BestCosts=out.BestCosts;
%% Results
figure;
plot(BestCosts, 'LineWidth',2);
xlabel('Iteration');
ylabel('Best Cost');
grid on;
figure;
semilogy(BestCosts, 'LineWidth',2);
xlabel('Iteration');
ylabel('Best Cost');
grid on;
```

Problem 10

Find the optimum to the using problem

$$min\ f(x) = -x_1^{\square} - x_2^{\square \cdot} \quad \textbf{(27)}$$

S.t to

$$g_1(x) = x_2^{\square} - x_1^{\ 2} \geq 0, \quad \textbf{(28)}$$

$$g_2(x) = 1 - x_1^{\ 2} - x_2^{\ 2} \geq 0 \quad \textbf{(29)}$$

(R. Fetcher)[11]

The Solution

The unknown decisions are 3, the lower bound is -1 and the

upper one is 1, the maximum number of iterations is 5, and the SQP size is 5.

The gradient of constrained Function

```
function g=GCFProb10(x)
g=[-1
-1];
End
```

Constrained Function

```
function f=CFProb10(x)
g1=-x(2) -x(1)^2;
g2=1-x(1)^2-x(2)^2;
f=[g1;g2];
end
```

SQPProblem10 .m

```
clc;
clear;
close all;
%% Problem Definition
problem.GradCostFunction = @(x) GCFProb10(x);    % Gradient of Cost Function
problem.ConstFunction    = @(x) CFProb10(x);     % Constrained Function
problem.nVar=2;              % Number of Unknown (Decision) Variables
problem.VarMin =-1;              % Lower Bound of Decision Variables
problem.VarMax =1;               % Upper Bound of Decision Variables
%% Parameters of SQP
paams.MaxIt = 5;       % Maximum Number of Iterations
```

```
params.nPop =5;           % Population Size (SQP Size)
params.ShowIterInfo = true;  % Flag for Showing Iteration
information
%% Calling SQP
out=SQP(problem, params);
BestSol=out.BestSol;
BestCosts=out.BestCosts;
%% Results
figure;
plot(BestCosts, 'LineWidth',2);
xlabel('Iteration');
ylabel('Best Cost');
grid on;
figure;
semilogy(BestCosts, 'LineWidth',2);
xlabel('Iteration');
ylabel('Best Cost');
grid on;
```

3.Conclusion

An a Algorithm was developed using the *SQP* method to build a MATLAB function, for solving (3), Computation procedure of the *SQP* method is described and carried out by evaluating the gradient of constrained function, defining the constrained function of the problem, mentioning the parameters of choosing alpha acceptable value in the direction, initializing empty particle of the old and new position. Then a flag for showing iteration information was drown, a array to hold best cost on each iteration and main loop of SQP were built, the best cost value was stored, and iteration information was displayed.Algorithm was applied

to solve selected problems.

4.References

Alan R. Parkinson, Richard j. Balling, John D. Hedengren. (2013),Optimization methods for Engineering Design Application and Theory Brigham Young University.

Boggs P.T, Tolle J.W. (1995), *Sequential quadratic programming,* Acta Numerica, Vol. 4, 1 – 51

Edgar T.F. Himmelblau D.M. (1988): *Optimization of Chemical Processes,* McGraw Hill

Gould N.I.M, Toint Ph.L. (1999): *SQP methods for large-scale nonlinear programming,,* Proceedings of the 19th IFIP TC7 Conference on System Modelling and Optimization: Methods, Theory and Applications, pp. 149–178.

Han S.-P. (1976): *Superlinearly convergent variable metric algorithms for general nonlinear programming problems* Mathematical Programming, Vol. 11, 263-282

Han S.-P. (1977): *A globally convergent method for nonlinear programming* Journal of Optimization Theory and Applications, Vol. 22, 297–309 17

Omojokun E.O. (1989): *Trust Region Algorithms for Optimization with Nonlinear Equality and Inequality Constraints,,* Ph.D. Thesis, University of Colorado at Boulder, USA

Papalambros P.Y., Wilde D.J. (1988): *Principles of Optimal Design,* Cambridge University Press

Powell M.J.D. (1978): *A fast algorithm for nonlinearly constraint optimization calculations,* in: Numerical Analysis, G.A. Watson ed., Lecture Notes in Mathematics, Vol. 630, Springer

Powell M.J.D. (1978): *The convergence of variable metric methods for nonlinearly constrained optimization calculations,* in: Nonlinear Programming 3, O.L. Mangasarian, R.R. Meyer, S.M. Robinson eds., Academic Press

Fletcher R. (1986) Practical Methods of Optimization, Second edition.

Schittkowski K. (1980): *Nonlinear Programming Codes,* Lecture Notes in Economics and Mathematical Systems, Vol. 183 Springer

Wilson R.B. (1963): *A simplicial algorithm for concave programming,* Ph.D. Thesis, Harvard University