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

Dr. Suida 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
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.
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):

\[
\text{minimize } f(x) = \frac{1}{2} x^T G x + q^T x q
\]

(1)

\[
\text{minimize } \nabla f(x)^T d_k + \frac{1}{2} d_k^T B_k d_k
\]

(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:

\[
\text{minimize } f(x) = \frac{1}{2} x^T G x + q^T x q
\]

(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 \( L(x, u) \) 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 Lagrangian function, \( k = 0, 1, 2, \ldots, n \) then, a quadratic program(QP) of the forms
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,

\[
\text{minimize } f(x), \text{over } x \in \mathbb{R}^n, g(x) \geq 0,
\]

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

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

In general, the unconstrained optimization problems are described as follows

\[
\min_{x \in \mathbb{R}^n} f(x)
\]

where \( \mathbb{R}^n \) is an \( n \) —dimensional Euclidean space and \( f: \mathbb{R}^n \to \mathbb{R} \) is assumed to be continuously twice differentiable. The gradient and Hessian for (5) are denoted as \( \nabla f \) and \( \nabla^2 f \), respectively. In order to display the updated formula of BFGS, the step-vector, and \( y_k \) are defined as:

\[
d_k = x_{k+1} - x_k
\]

\[
y_k = g(x_{k+1}) - g(x_k)
\]

\[
y_k = g_{k+1} - g_k
\]

The search direction \( p_k \) at stage \( k \) is given by the solution of the analogue of the Newton equation
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 in the direction \( d_k \) 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,
Choosing \( h = y_k \) and \( t = d_k \), we can obtain:

\[
\alpha = \frac{1}{y_k^T d_k}
\]

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

Finally, we substitute \( \alpha \) and \( \beta \) into \( B_{k+1} = B_k + \alpha hh^T + \beta tt^T \) 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 \( x_0 \) and an approximate Hessian matrix \( B_0 \) the following steps are repeated as \( x_k \) converges to the solution:

Obtain a direction \( d_k \) by solving

\[
B_k d_k = -\nabla f(x_k)
\]

So the linear search \( d_k \) is define by:

\[
d_k = -B_k^T \nabla f(x_k)
\]

(13)

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

\[
\alpha_k = \arg \min f(x_k + \alpha d_k)
\]

Set \( d_k = \alpha_k p_k \) and update the personal best
And update the global best as:

\[
x_{k+1} = x_k
\]

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 \( i = 1, 2, \ldots, n \) population do:

1) Compute:
   1) \( g(x_k) - g'(x_k) \)
   2) \( d_k = -B_k^T \nabla f(x_k) \);
   3) \( \min f(x) = g f(x_k)'d + 0.5 d_k' B_k \);

2) Compute:

\[
\min f(x) = g f(x_k)'d + 0.5 d_k' B_k
\]

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

1) then update;

   i) \( x_{k+1} = x_k + \alpha d_k \);
   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:

3) Update personal best

4) For \( i = 1: \text{MaxIt} \) and for \( i = 1: \text{nPop} \),

Compute:

   i) \( x_{k+1} = x_k + \alpha d_k \);
   ii) \( y_k = g(x_{k+1}) - g(x_k) \);
iii) \( B_{k+1} = B_k + \frac{y_k y_k^T}{d_k y_k} d_k d_k^T B_k d_k^T B_k^T \)

\[
\alpha = \frac{1}{y_k ' d_k} ;
\]

\[
g(x_k) = g^{f}(x_k) ;
\]

\[
dx_{k+1} : = x_k
\]

Compute

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

ii) \( d_k = -B_k^T g f(x_k) \)

iii) \( \min f(x_k) = g f(x_k)' d + 0.5 d_k^T B_k \)

Update personal best

\[
x_{k+1} : = x_k \quad \min f(x_{k+1}) = \min f(x_k)
\]

Update Glob Best

If \( x_{k+1} : = x_k d' \)

Compute: \( \min f(x_{k+1}) ; \)

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

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:

\textbf{SQPProblem(i) .m}

clc;
clear;
close all;

\{266\}
%% 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');
2.4: The Chosen Problems:

In this sub section we applied our algorithm in ten chosen problems with different iterations.

**Problem 1**
Consider using SQP method to solve the problem

\[
\min -x_1 x_2 x_3 \quad (1)
\]

Subject to

\[
72 - x_1 - 2x_2 - 2x_3 \quad (2)
\]

(R. Fletcher)

**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:

```matlab
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:

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

SQPProblem1.m:

```matlab
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');

{269}
ylabel('Best Cost');
grid on;

**Problem 2**
Consider using SQP method to solve the problem

\[
\text{minimize } \exp(x_1x_2x_3x_4x_5) \tag{3}
\]

Subject to

\[
x_1^2 + x_2^2 + x_3^2 + x_4^2 + x_5^2 = 10 \tag{4}
\]

\[
x_2x_3 = 5x_4x_5 \tag{5}
\]

\[
x_1^3 + x_2^3 = -1 \tag{6}
\]

(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

```matlab
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

```matlab
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:
```matlab
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

\[
\text{minimize} \quad x_1 x_2 \quad x \in \mathbb{R}^4
\]  \hspace{1cm} (7)

Subject to

\[
\frac{(x_1 x_3 + x_4 x_5)^2}{(x_1^2 + x_2^2)} - x_3^2 - x_4^2
\]  \hspace{1cm} (8)

\[
x_1 \geq x_3 + 1 \quad (9)
\]

\[
x_2 \geq x_4 + 1 \quad (10)
\]

\[
x_3 \geq x_4 \quad (11)
\]

\[
x_4 \geq 1 \quad (12)
\]
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

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

The Constrained Function

```matlab
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 :

```matlab
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 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 4
Consider the geometric programming using SQP method to solve the problem

\[
\begin{align*}
\text{minimize } & \quad x_1^{-1}x_2^{-1}x_3^{-1} \\
\text{subject to } & \quad x_1 + 2x_2 - 2x_3 \leq 72 \\
& \quad x > 1
\end{align*}
\]

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

\[
\begin{align*}
\text{function } f &= \text{GCFProb4}(x) \\
g &= [-x(1)^{-2}x(2)^{-1}x(3)^{-1} \\
&\quad -x(1)^{-1}x(2)^{-2}x(3)^{-1} \\
&\quad -x(1)^{-1}x(2)^{-1}x(3)^{-2}];
\end{align*}
\]

\[f = g;\]

end

The Constrained Function

\[
\begin{align*}
\text{function } f &= \text{CFProb4}(x) \\
f &= -x(1) - 2x(2) + 2x(3) + 72;
\end{align*}
\]

end

SQPProblem4 .m
clc;
clear;
close all;

%% Problem Definition
problem.GradCostFunction = @(x) GCFProb4(x); % Gradient 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 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 5**
Consider the geometric programming using SQP method to solve the problem

\[
\begin{align*}
\text{minimize} & \quad f(x) = 40x_1x_2 + 20x_2x_3 \\
\text{Subject to} & \quad \frac{1}{5}x_1^{-1}x_2^{\frac{1}{2}} + \frac{3}{5}x_2^{-1}x_3^{\frac{2}{3}} \leq 1, \quad x > 0
\end{align*}
\]

(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

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

Constrained Function

```matlab
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
params.MaxIt = 100;                    % Maximum Number of Iterations
params.nPop =50;                       % Population Size (SQP Size)
params.ShowIterInfo = true;           % Flag for Showing Iteration information
out=SQP(problem, params);             % Calling SQP
BestSol=out.BestSol;
BestCosts=out.BestCosts;
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 problem using problem

\[ \begin{align*}
\min f(x) &= x_1^4 - 2x_2x_1^2 + x_2^2 + x_1^2x_2 + 9 = 0 \\
\text{Subject to} \quad g(x) &= -\left(\|x_1 + 0.25\|\right)^2 + 0.75x_2 \geq 0
\end{align*} \] (18) (19)

(Optimization)

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 \( f = \text{GCFProb6}(x) \)
\[ g = [4*x(1)^3 - 4*(x(2)*x(1)) + 2*x(1)
- 2*x(1)^2 + 2*x(2)]; \]
\( f = g; \)
end

Constrained function

function \( f = \text{CFProb6}(x) \)
\( f = -(x(1) + 0.25)^2 + 0.75*x(2); \)
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

\{ 279 \}
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 Iterations
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

\[ \min f(x) = x_1^{-1} x_2^{-1} x_3^{-1} \]  \hspace{1cm} (20)
Subject to

\[ g(x) = x_1 + 2x_2 - 2x_2 \leq 75 \]  \hspace{1cm} (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 \( f = \text{GCFProb7}(x) \)

\[
g = [-x(1)^2 - 2x(2)^2 - 1x(3)^2 - 1 \]
\[
- x(1)^2 - 1x(2)^2 - 2x(3)^2 - 1 \]
\[
- x(1)^2 - 1x(2)^2 - 1x(3)^2 - 2 \];
\]

\( f = g; \)

end

Constrained Function

function \( f = \text{CFProb7}(x) \)

\( f = -x(1) - 2x(2) + 2x(3) + 72; \)

end

SQPProblem7.m

clc;
clear;
close all;

%% Problem Definition

problem.GradCostFunction = @(x) GCFProb7(x); \hspace{1cm} \% Gradient of Cost Function

problem.ConstFunction = @(x) CFProb7(x); \hspace{1cm} \% Constrained Function

problem.nVar = 3; \hspace{1cm} \% Number of Unknown (Decision) Variables

problem.VarMin = -10; \hspace{1cm} \% Lower Bound of Decision
Variables
problem.VarMax = 10; \hspace{1cm} % Upper Bound of Decision Variables

%%% Parameters of SQP
params.MaxIt = 100; \hspace{1cm} % Maximum Number of Iterations
params.nPop = 10; \hspace{1cm} % Population Size (SQP Size)
params.ShowIterInfo = true; \hspace{1cm} % 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

\[
\min f(x) = 2x_1^2 + 4x_2^2 \hspace{1cm} (22)
\]

S.t to

\[
g(x) = 3x_1 + 2x_2 \geq 12 \hspace{1cm} (23)
\]

(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

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

Constrained Function

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

SQPProblem8.m

clc;
clear;
close all;

```matlab
%% 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
\[ \min \ f(\alpha) = x_1^2 + x_2^2 \] (24)
S.t to
\[ g_1(\alpha) = x_1^2 + x_2^2 - 9 \leq 0 \] (25)
\[ g_2(\alpha) = x_1^4 + x_2^4 - 1 \leq 0 \] (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 \( g = GCFProb9(x) \)
\[
\begin{align*}
g &= [2 \cdot x(1) \\
    & \quad 2 \cdot x(2)];
\end{align*}
\]
end

Constrained Function
function \( f = CFProb9(x) \)
\[
\begin{align*}
g1 &= -x(1)^2 - x(2)^2 + 9; \\
g2 &= -x(1) - x(2) + 1; \\
f &= [g1; g2];
\end{align*}
\]
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

\[
\min f(x) = -x_1^2 - x_2^2
\]  \hspace{1cm} (27)

S.t to

\[
g_1(x) = x_2^2 - x_1^2 \geq 0, \hspace{1cm} (28)
g_2(x) = 1 - x_1^2 - x_2^2 \geq 0 \hspace{1cm} (29)
\]

(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

{287}
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 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


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


