# Solving minimax problems with feasible sequential quadratic programming

Qianli Deng (Shally)

E-mail: dqianli@umd.edu

Advisor: Dr. Mark A. Austin

Department of Civil and Environmental Engineering

Institute for Systems Research

University of Maryland, College Park, MD 20742

E-mail: austin@isr.umd.edu

# Abstract:

As one of the SQP-type programming, FSQP involves the solutions of quadratic programs as subproblems. This algorithm is particularly suited to the various classes of engineering applications where the number of variables is not too large but the evaluations of objective or constraint functions and of their gradients are highly time consuming. This project is intended to implement the Feasible Sequential Quadratic Programming (FSQP) algorithm as a tool for solving certain nonlinear constrained optimization problems with feasible iterates.

Keywords: Nonlinear optimization; minimax problem; sequential quadratic programming; feasible iterates

# 1. Introduction

The role of optimization in both engineering analysis and design is continually expanding. As a result, the faster and more powerful optimization algorithms are in constant demand. Motivated by problems from engineering analysis and design, feasible sequential quadratic programming (FSQP) are developed as a dramatic reduction in the amount of computation while still enjoying the same global and fast local convergence properties.

The application of FSQP includes all branches of engineering, medicine, physics, astronomy, economics and finances, which abounds of special interest. In particular, the algorithms are particularly appropriate for problems where the number of variables is not so large, while the function evaluations are expensive and feasibility of iterates is desirable. But for problems with large numbers of variables, FSQP might not be a good fit. The minimax problems with large numbers of objective functions or inequality constraints, such as finely discretized semi-infinite optimization problems, could be handled effectively, for instance, problems involving time or frequency responses of dynamical systems.

The typical constrained minimax problem is in the following format showing in eq.(1).

|  |  |  |
| --- | --- | --- |
|  | minimize |  |

where  is smooth. FSQP generates a sequence  such that  for all  and .  where stands for the number of objective functions . If , .  is a set of points  satisfying the following constraints, as shown in eq.(2).

|  |  |  |
| --- | --- | --- |
|  |  |  |

where  and  are smooth;  stands for the number of nonlinear inequality constraints;  stands for the total number of inequality constraints;  stands for the number of nonlinear equality constraints, and  stands for the total number of inequality constraints.  stands for the constraint of lower boundary and  stands for the constraint of upper boundary. , ,  and  stand for the parameters of linear constraints.

# 2. Algorithm

To solve the constrained minimax problem using FSQP, five steps are taken as a sequence: 1) initialization; 2) computation of a search arc; 3) arch search; 4) updates; and 5) stop.

Parameters setting as: , , , , , , , , , , .

Data: , , and  for .

## 2.1 Initialization

FSQP solves the original problem with nonlinear equality constraints by solving a modified optimization problem with only linear constraints and nonlinear inequality constraints. Set , the identity matrix as the initial Hessian matrix, and an initial guess . If  is infeasible for linear constraints, a point  satisfying these constraints is generated by solving the strictly convex quadratic program. If  or newly generated initial guess is infeasible for the nonlinear inequality constraints, a point  is generated satisfying all constraints by iterating on the problem of minimizing the maximum of the nonlinear inequality constraints.

|  |  |  |
| --- | --- | --- |
|  | , |  |

and the original objective function  is replaced by the modified objective function

|  |  |  |
| --- | --- | --- |
|  |  |  |

where , , which are positive penalty parameters that are iteratively adjusted. For , replace  by  whenever .

Based on a sequential quadratic programming iteration, an Armijo-type line search is used when minimizing the maximum of the nonlinear inequality constraints to generate an initial feasible point . If  is infeasible for some constraint other than a nonlinear equality constraint, substitute a feasible point.

## 2.2 Stop check

If  and , iteration stops and return the value .

## 2.3 Computation of a search arc

From an initial guess , the following steps are repeated as  converges to the solution. The four steps are used to compute a search arc: 1) compute ; 2) compute ; 3) compute ; and 4) compute .  stands for the direction of descent for the objective function and  stands for an arbitrary feasible descent direction.  stands for the feasible descent direction between the directions of  and . And  stands for a second order correction which could be deemed as a “bent” of the search direction. Inner relations among the parameters are displayed in Figure 1.



Figure 1. Calculations of direction *d* in FSQP

### 2.3.1 Compute

Compute , the solution of the quadratic program . At each iteration , the quadratic program  that yields the SQP direction  is defined at  for  symmetric positive definite by

|  |  |  |
| --- | --- | --- |
|  |  |  |

Given , following notation is made.

|  |  |  |
| --- | --- | --- |
|  |  |  |
|  |  |  |

### 2.3.2 Compute

Compute  by solving the strictly convex quadratic program

|  |  |  |
| --- | --- | --- |
|  |  |  |

### 2.3.3 Compute

Set  with , where .

### 2.3.4 Compute

In order to avoid the Maratos effect and guarantee a superlinear rate of convergence, a second order correction  is used to “bend” the search direction. That is an Armijo-type search is performance along the arc . The Maratos correction  is taken as the solution of QP.

|  |  |  |
| --- | --- | --- |
|  |  |  |

Compute  by solving the strictly convex quadratic program

|  |  |  |
| --- | --- | --- |
|  |  |  |

The set of active constraints by

|  |  |  |
| --- | --- | --- |
|  |  |  |

## 2.4 Arc search

If , , while if , . Compute , the first number  in the sequence  satisfying

|  |  |  |
| --- | --- | --- |
|  |  |  |
|  | , . |  |
|  | , & |  |
|  | , |  |

## 2.5 Updates

The updating scheme for the Hessian estimates  is also defined in implementation. Using BFGS formula with Powell’s modification to compute the new approximation  as the Hessian of the Lagrangian, where 

|  |  |  |
| --- | --- | --- |
|  |  |  |
|  |  |  |

where stands for the Lagrange function, and  stands for the Lagrange multiplier.

A scalar  is then defined by

|  |  |  |
| --- | --- | --- |
|  |  |  |

Defining  as

|  |  |  |
| --- | --- | --- |
|  |  |  |

The rank two Hessian update is

|  |  |  |
| --- | --- | --- |
|  |  |  |

Then, the new point is set as

|  |  |  |
| --- | --- | --- |
|  |  |  |

Solve the unconstrained quadratic problem in 

|  |  |
| --- | --- |
|  |  |

For , update the penalty parameters as

|  |  |  |
| --- | --- | --- |
|  |  |  |

Set 

# 3. Implementation

Hardware: the program will be developed and implemented on my personal computer.

Software: the program will be developed by Java.

# 4. Validation

Example. For , the objective function 

The constraints are



Feasible initial guess , 

Global minimizer , 

# 5. Testing

Example. The objective function 



Where 

, 



The feasible initial guess is 

With the corresponding value of the objective function 

Other practical problems such as the wind turbine setting could also be applied if time permits.

# 6. Project Schedule

|  |  |
| --- | --- |
| Time | Tasks |
| October | * Literature review; * Specify the implementation module details; * Structure the implementation; |
| November | * Develop the quadratic programming module; * Unconstrained quadratic program; * Strictly convex quadratic program; * Validate the quadratic programming module; |
| December | * Develop the feasible initial point module; * Validate the feasible initial point module; * Develop the Gradient and Hessian matrix calculation module; * Validate the Gradient and Hessian matrix calculation module; * Midterm project report and presentation; |
| February | * Develop Step II - the computation of a search arc; * Validate Step II; * Develop Step IV - the Hessian matrix and the penalty parameters updating; * Validate Step IV; |
| March | * Develop Step III - the arc search; * Integrate the program; |
| April | * Debug and well document the program; * Validate and test the program; * Develop the user interface if time available; |
| May | * Final project report and presentation; |

# 7. Deliverables

* Project proposal;
* Algorithm description;
* Java code;
* Validation results;
* Test database;
* Test results;
* Project reports;
* Presentations;

# 8. Bibliography

Charalambous, Conn and AR Conn. "An Efficient Method to Solve the Minimax Problem Directly." *SIAM Journal on Numerical Analysis* 15, no. 1 (1978): 162-187.

Goldfarb, Donald and Ashok Idnani. "A Numerically Stable Dual Method for Solving Strictly Convex Quadratic Programs." *Mathematical programming* 27, no. 1 (1983): 1-33.

Lawrence, Craig T and André L Tits. "Nonlinear Equality Constraints in Feasible Sequential Quadratic Programming∗." *Optimization Methods and Software* 6, no. 4 (1996): 265-282.

Lawrence, Craig T and André L Tits. "A Computationally Efficient Feasible Sequential Quadratic Programming Algorithm." *Siam Journal on optimization* 11, no. 4 (2001): 1092-1118.

Li, Wu and John Swetits. "A New Algorithm for Solving Strictly Convex Quadratic Programs." *SIAM Journal on Optimization* 7, no. 3 (1997): 595-619.

Madsen, Kaj and Hans Schjaer-Jacobsen. "Linearly Constrained Minimax Optimization." *Mathematical Programming* 14, no. 1 (1978): 208-223.

Mayne, DQ and E Polak. "Feasible Directions Algorithms for Optimization Problems with Equality and Inequality Constraints." *Mathematical Programming* 11, no. 1 (1976): 67-80.

Watson, GA. "The Minimax Solution of an Overdetermined System of Non-Linear Equations." *IMA Journal of Applied Mathematics* 23, no. 2 (1979): 167-180.