# 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 as 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, four steps are taken as a sequence: 1) initialization; 2) computation of a search arc; 3) arch search; and 4) updates. Figure 1 below shows a flowchart indicating how FSQP works for solving the constrained minimax problem.



Figure 1. Flowchart for solving the constrained minimax problem using FSQP

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.  and  are initialized in the first step and then updated each time within the loop. For finding the initial feasible point , an initial guess  is generated. If  is infeasible for some constraints other than a nonlinear equality constraint, substitute another feasible point.

Table 1. Methods for finding an initial feasible point

|  |  |
| --- | --- |
| Initial guess | Method |
| Infeasible for linear constraints | Strictly convex quadratic programming |
| Infeasible for the nonlinear inequality constraints | Armijo-type line search |

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. An Armijo-type line search is used when minimizing the maximum of the nonlinear inequality constraints to generate an initial feasible point .

 is an identity matrix as the initial Hessian matrix. Since the constraints , , 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 . Set the loop index .

## 2.2 Computation of a search arc

From an initial guess , the following steps are repeated as  converges to the solution. 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 2.



Figure 2. Calculations of direction *d* in FSQP

### 2.2.1 Compute

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

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

Given , following notation is made.

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

### 2.2.2 Stop check

If  and , iteration stops and return the value .

### 2.2.3 Compute

Compute , solution of  by solving the strictly convex quadratic program as follows

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

### 2.2.4 Compute

Set  with , where .

### 2.2.5 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

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

## 2.3 Arc search

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

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

## 2.4 Updates

 and  are updated within each loop. Firstly, the new point  is calculated based on the following

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

Then, the updating scheme for the Hessian estimates uses 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

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

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 in 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 Gradient and Hessian matrix calculation module; * Validate the Gradient and Hessian matrix calculation module; * Midterm project report and presentation; |
| January | * Develop Armijo line search module; * Validate Armijo line search module; |
| February | * Develop the feasible initial point module; * Validate the feasible initial point module; * Integrate the program; |
| March | * Debug and document the program; * Validate and test the program with case application; |
| April | * Add arch search variable in; * Compare calculation efficiency of line search with arch search methods; |
| May | * Develop the user interface if time available; * 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.

# Appendix A – Nomenclature

|  |  |  |
| --- | --- | --- |
| Symbols |  | Interpretation |
|  |  | Sequence of objective functions |
|  |  | Number of objective functions |
|  |  | Inequality constraints function |
|  |  | Equality constraints function |
|  |  | Number of nonlinear inequality constraints |
|  |  | Total number of inequality constraints |
|  |  | Number of nonlinear equality constraints |
|  |  | Total number of inequality constraints |
|  |  | Lower boundary of the variables |
|  |  | Upper boundary of the variables |
|  |  | Parameters of linear equality constraints |
|  |  | Parameters of linear equality constraints |
|  |  | Parameters of linear inequality constraints |
|  |  | Parameters of linear inequality constraints |
|  |  | Initial guess |
|  |  | Initial feasible point |
|  |  | Initial Hessian matrix as the identity matrix |
|  |  | Feasible point in *k* iteration |
|  |  | Hessian matrix in *k* iteration |
|  |  | Positive penalty parameters in *k* iteration |
|  |  | Iteration index |
|  |  | Descent direction for the objective function in *k* iteration |
|  |  | An arbitrary feasible descent direction in *k* iteration |
|  |  | A feasible descent direction between the directions of  and  in *k* iteration |
|  |  | A second order curve direction in *k* iteration |
|  |  | Weight between directions of  and  in *k* iteration |

|  |  |
| --- | --- |
| Symbols | Values |
|  | 0.1 |
|  | 0.01 |
|  | 0.1 |
|  | 0.5 |
|  | 2.1 |
|  | 2.5 |
|  | 2.5 |
|  | 0.1 |
|  | 1 |
|  | 2 |
|  | 2 |

# Appendix B – Solving strictly convex quadratic programming

Quadratic programming is the problem of optimizing a quadratic function of several variables subject to linear constraints on these variables. Consider a convex quadratic programming problem of the following form, minimizing with respect to the vector *x* of the following function showing in eq.(1).

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

where *n* stands for the number of variables.  is a column of  variables, .  is a  symmetric positive definite matrix that . is a *n* dimension column vector. Subject to the linear inequality constraints as showing in eqs.(2) and (3).

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

where *m* stands for the number of constraints. *A* is an  matrix. is a given column vector of *m* elements. All the variables are constraint to be nonnegative. Combining the objective function as well as the constraints, the Lagrangian function for the quadratic program is formed as the following expression eq.(4).

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

where  is the Lagrangian multipliers as a column vector of *m* dimensional, . Karush-Kuhn-Tucker conditions are first order necessary conditions for a solution in nonlinear programming to be optimal, provided that the following regularity conditions eqs.(5)~(8) are satisfied.

|  |  |  |
| --- | --- | --- |
|  |  |  |
|  | then |  |
|  |  |  |
|  |  |  |

Thinking to solve the problem, slackness variables have been added in to switch the inequality constraints to equality constraints. As a result,  is the  slackness matrix for constraints and  is the  slackness matrix for objective functions. Then, the following eqs.(9)~(12) have been derived.

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

where all the variables are nonnegative, , , , and . Eqs.(13)~(16) are derived based on eqs.(9)~(12), which works as an equivalent problem of the original quadratic programming to be solved.

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

To solve the above problem, artificial variables  are generated for the initial basis composition.  and , as well as  and  are complementary slack variables. As a result, another coefficient vector as the Lagrangian multipliers is given in the objective function. The complementary slackness conditions are satisfied in nature. Then, the problem becomes the following optimization problem with the objective function of eq.(17) and the constraints of eqs.(18)~(20).

|  |  |  |
| --- | --- | --- |
|  | Minimize |  |
|  | Subject to |  |
|  |  |  |
|  |  |  |

where is an *n* dimension column vector. Here we set the coefficient of to be 1. Then,  is also an *n* dimension column vector that has been adjusted as the coefficient for accordingly.

In order to solve the modified quadratic programming problem, the Wolfe’s extended simplex method has been chosen which do not requires the gradient and Hessian computation. Accordingly, three major steps are taken as: 1) set up the initial tableau; 2) repeat the loop until the optimal solution found; and 3) return the final solution. Details are explained as below.

**Step 1. Set up the initial tableau**

According to the constraints eqs.(18) and (19), Table 1 has been set up for further computation. Variables  and are chosen as the initial bases.

Table A1. Step-up the initial tableau for the Simplex method

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Basic variables | Values basic variables |  |  |  |  |  |
|  | *b* | *A* |  |  | *I* |  |
|  | *- d* | *C* | *A'* | *- I* |  | *I* |

Since feasible values of  and  have not been reached yet, the following coefficients are set as a start for calculation simplification.

Table A2. Step-up the initial price value for the Simplex method

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Variables |  |  |  |  |  |
| Values | 0 | 0 | 0 | 0 |  |

**Step 2. Repeat the loop until the optimal solution found**

There are three sub-steps to be conducted within the loop as: 1) search for pivot element; 2) perform pivoting; and 3) update price. When each time a new feasible solution is found through pivoting, the price table for variables is updated, in order to check whether the new combination could help to reduce the value of the objective function. The pivot element is the new variable selected to enter the basis, while the previous one is the one to leave. Two rules are followed by searching for the pivot element in the loop.

* *Rule 1. Search for the pivot column. Find the largest negative indicator in the price table after canonical transformation.*
* *Rule 2. Search for the pivot row. Find the smallest nonnegative indicator for the ratio.*

**Step 3. Return the final solution**

For the loop iteration, the stopping criteria is when the variable changes could no longer help to reduce the value of the objective function , which means the updated price values after canonical transformation are all positive.

In order to demonstrate the feasibility for solving quadratic programming problem, the following test case has been conducted:

|  |  |  |
| --- | --- | --- |
|  | Minimize |  |
|  | Subject to |  |
|  |  |  |
|  |  |  |

Accordingly, the input parameters are set asthe number of constraints *m* = 2; number of variables *n* = 2; and, , , , . Initial set-up tableau and price value have been set as Table 3 and 4 indicating respectively.

Table A3. Application of the simplex method for quadratic programming

|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Basic variables | Values basic variables |  |  |  |  |  |  |  |  |  |  |
|  | 6.0 | 2.0 | 1.0 | 0.0 | 0.0 | 0.0 | 0.0 | 1.0 | 0.0 | 0.0 | 0.0 |
|  | 0.0 | 1.0 | -4.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 1.0 | 0.0 | 0.0 |
|  | 4.0 | 2.0 | -2.0 | 2.0 | 1.0 | -1.0 | 0.0 | 0.0 | 0.0 | 1.0 | 0.0 |
|  | 0.0 | -2.0 | 4.0 | 1.0 | -4.0 | 0.0 | -1.0 | 0.0 | 0.0 | 0.0 | 1.0 |

Table A4. Step-up the initial price value for the Simplex method

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Variables |  |  |  |  |  |  |  |  |  |  |
| Values | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 1.0 | 1.0 |

Switching Table 4 to canonical form, the price value is shown as Table 5 below.

Table A5. Price value after canonical transformation

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Variables |  |  |  |  |  |  |  |  |  |  |
| Value | 0.0 | -2.0 | -3.0 | 3.0 | 1.0 | 1.0 | 0.0 | 0.0 | 0.0 | 0.0 |

As a result, variable  is chosen as the pivot variable indicating the pivot column.

The pivot row is selected from the smallest nonnegative elements of . Then, the pivot element based on the column and row is chosen to be 1.0. Accordingly, pivoting has been conducted and Table 6 is set as follows.

Table A6. Application of the simplex method for quadratic programming

|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Basic variables | Values basic variables |  |  |  |  |  |  |  |  |  |  |
|  | 6.0 | 2.0 | 1.0 | 0.0 | 0.0 | 0.0 | 0.0 | 1.0 | 0.0 | 0.0 | 0.0 |
|  | 0.0 | 1.0 | -4.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 1.0 | 0.0 | 0.0 |
|  | 4.0 | 6.0 | -10.0 | 0.0 | 9.0 | -1.0 | 0.0 | 0.0 | 0.0 | 1.0 | -2.0 |
|  | 0.0 | -2.0 | 4.0 | 1.0 | -4.0 | 0.0 | -1.0 | 0.0 | 0.0 | 0.0 | 1.0 |

From Table 6, , ,  and  built a feasible solution. Then the price values of the corresponding complementary variables are set. For instance, the complementary variable of  is . The updated price value for  is 6.0. Table 7 below indicates the details.

Table A7. Price value after canonical transformation

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Variables |  |  |  |  |  |  |  |  |  |  |
| Value | 0.0 | 0.0 | 6.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 1.0 | 1.0 |

Repeating the process, the following tableau updating is demonstrated in Table 8.

Table A8. Application of the simplex method for quadratic programming

|  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Iteration | Basic variables | Values basic variables |  |  |  |  |  |  |  |  |  |  |
| 3 |  | 6.0 | 2.5 | 0.0 | -0.25 | 1.0 | 0.0 | 0.25 | 1.0 | 0.0 | 0.0 | -0.25 |
|  | 0.0 | -1.0 | 0.0 | 1.0 | -4.0 | 0.0 | -1.0 | 0.0 | 1.0 | 0.0 | 1.0 |
|  | 4.0 | 1.0 | 0.0 | 2.5 | -1.0 | -1.0 | -0.5 | 0.0 | 0.0 | 1.0 | 0.5 |
|  | 0.0 | -0.5 | 1.0 | 0.25 | -1.0 | 0.0 | -0.25 | 0.0 | 0.0 | 0.0 | 0.25 |
| 4 |  | 2.4 | 1.0 | 0.0 | -0.1 | 0.4 | 0.0 | 0.1 | 0.4 | 0.0 | 0.0 | -0.1 |
|  | 2.4 | 0.0 | 0.0 | 0.9 | -3.6 | 0.0 | -0.9 | 0.4 | 1.0 | 0.0 | 0.9 |
|  | 1.6 | 0.0 | 0.0 | 2.6 | -1.4 | -1.0 | -0.6 | -0.4 | 0.0 | 1.0 | 0.6 |
|  | 1.2 | 0.0 | 1.0 | 0.2 | -0.8 | 0.0 | -0.2 | 0.2 | 0.0 | 0.0 | 0.2 |
| 5 |  | 2.46 | 1.0 | 0.0 | 0.0 | 0.35 | -0.04 | 0.08 | 0.38 | 0.0 | 0.04 | -0.08 |
|  | 1.85 | 0.0 | 0.0 | 0.0 | -3.12 | 0.35 | -0.69 | 0.54 | 1.0 | -0.35 | 0.69 |
|  | 0.62 | 0.0 | 0.0 | 1.0 | -0.54 | -0.38 | -0.23 | -0.15 | 0.0 | 0.38 | 0.23 |
|  | 1.08 | 0.0 | 1.0 | 0.0 | -0.69 | 0.08 | -0.15 | 0.23 | 0.0 | -0.08 | 0.15 |

Finally, optimal solutions are reached in the fifth iteration that , , .