1 Sequential Quadratic Programming Algorithm

Sequential Quadratic Programming (SQP) methods are in general robust and efficient. As suggested be the name, SQP methods are able solve non-linear problems by incorporating second-order information about the objective function. The two broad stages of SQP (as stated in the previous section)—subproblems (1) finding a search direction and (2) calculating the step size—can be solved using various numerical methods. The combination of choices of different numerical methods to use results in different SQP methods. In this section, we introduce the mathematical formulation of SQP that we have adopted for our implementation. Our implementation only considers inequality constraints.

1.1 Descent direction subproblem

In this section, we define the search direction subproblem as a quadratic programming (QP) problem. 2nd order taylor expansion is used to find a quadratic approximation of the objective function of the form:

$$\tilde{f} = f(x^o) + (x_1 - x_2)\left.\frac{\partial f}{\partial x_1}\right|_o + \frac{(x_1 - x_2)^2}{2}\left.\frac{\partial^2 f}{\partial x_1^2}\right|_o \tag{1}$$

Constraints are linearized using first order taylor expansion:

$$\tilde{g} = g(x^o) + (x_1 - x_2)\left.\frac{\partial g}{\partial x_1}\right|_o \tag{2}$$

The problem is then re-written as the QP subproblem as follows:

$$\textbf{Minimize:} \qquad \tilde{f} = \mathbf{c}^T\mathbf{d} + \frac{1}{2}\mathbf{d}^T\mathbf{H}\mathbf{d} \tag{3}$$

$$\text{Subject to the relevant constraints:} \qquad \mathbf{A}^T\mathbf{d} \leq \mathbf{b} \tag{4}$$

where

$$\mathbf{d} = n\text{-vector of search direction} [d_0, ..., d_n]^T \tag{5}$$

$\mathbf{c} = n$-vector $\nabla f(\mathbf{x}^k)$

$\mathbf{H} = n \times n$ Hessian matrix of f

$\mathbf{A} = n \times m$ matrix of constraint coefficients, i.e. for m constraints:$[\nabla g_0(\mathbf{x}^k), ..., \nabla g_m(\mathbf{x}^k)]^T$

d which minimizes the equation 3 gives the search direction for the next point in the iterative search process. To solve this QP problem efficiently, we can transform the problem into a linear programming problem, which can then be solved using linear programming methods such as the simplex method. To do so, we first write the KKT necessary conditions of the QP problem and then present them in a form that can be solved using the simplex method.

Slack variables are introduced to transform the inequality constraints into equalities:

$$\mathbf{A}^T\mathbf{d} + \mathbf{s} = \mathbf{b}; \text{ with } \mathbf{s} \geq 0 \tag{6}$$

The Lagrange function of the QP problem can then be written as

$$L = \mathbf{c}^T\mathbf{d} + 0.5\mathbf{d}^T\mathbf{H}\mathbf{d} + \mathbf{u}^T(\mathbf{A}^T\mathbf{d} + \mathbf{s} - \mathbf{b}) \tag{7}$$

where **u** is the Lagrange multiplier vector for the inequality constraints.

The KKT necessary conditions can be written as

$$\frac{\partial L}{\partial \mathbf{d}} = \mathbf{c} + \mathbf{H}\mathbf{d} + \mathbf{A}\mathbf{u} = 0 \tag{8}$$

$$\mathbf{A}^T\mathbf{d} + \mathbf{s} - \mathbf{b} = 0 \tag{9}$$

$$u_i s_i = 0;\ i = 1 \text{ to } m \tag{10}$$

$$u_i, s_i \geq 0 \text{ for } i = 1 \text{ to } m \tag{11}$$

which can be written compactly in the form of a linear system as

$$\mathbf{BX} = \mathbf{D} \tag{12}$$

where

$$\mathbf{B} = \begin{bmatrix} \mathbf{H} & \mathbf{A} & \mathbf{0} \\ \mathbf{A}^T & \mathbf{0} & \mathbf{I} \end{bmatrix}, \qquad \mathbf{X} = \begin{bmatrix} \mathbf{d} \\ \mathbf{u} \\ \mathbf{s} \end{bmatrix}, \qquad \mathbf{D} = \begin{bmatrix} -\mathbf{c} \\ \mathbf{b} \end{bmatrix} \tag{13}$$

The above matrices have been simplified from their full form, which includes treatment of equality and nonnegativity constraints. For the complete formulation, see Chapter 9.5 of the Arora text [1].

The solution to the above linear system can be solved using the simplex method by introducing artificial variables Υ_i for each equality constraint in the linear system, defining an artificial cost function, and forming the initial simplex tableau.

$$\mathbf{BX} + \boldsymbol{\Upsilon} = \mathbf{D} \tag{14}$$

The artificial cost function can be defined as follows:

$$w = \sum_{i=1}^{n+m} \Upsilon_i \tag{15}$$

$$= w_0 + \sum_{j=1}^{2(n+m)} C_j X_j \tag{16}$$

where

$$C_j = -\sum_{i=1}^{n+m} B_{ij} \text{ and } w_0 = \sum_{i=1}^{n+m} D_i \tag{17}$$

From the above, we can construct the initial simplex tableau and perform the simplex method to find **d**.

1.2 Step-size calculation subproblem

With the search direction d, we next need to find the optimal step length along d to traverse to get to the next best search point along the search direction. This step size α is found such that

$$\mathbf{x}^{k+1} = \mathbf{x}^k + \alpha \mathbf{d}^k \tag{18}$$

minimizes the objective function. Note that the search for the step size is a single variable line search problem than can be easily solved using techniques such as golden section or steepest descent.

In order to incorporate the constraints into the objective function for line search, a descent function is defined similar to the one described by Pshenichny [2]

$$\Phi(\mathbf{x}) = f(\mathbf{x}) + V(\mathbf{x}) \times 10 \tag{19}$$

Where V(x) is the numerical value of the maximum constraint violation and the multiple of 10 is a user defined parameter to prevent the algorithm from leaving the bounded feasible region.

$\mathbf{x}^{k+1}$ expressed in terms of α is then passed into the descent function, resulting in a univarate linear problem which can be solved by line search techniques such as golden section search.

After finding α, the point at the next iteration $\mathbf{x}^{k+1}$ can be calculated and used in the next iteration of SQP until convergence.

2 Implementation details

In our implementation of the algorithm, we have made a few assumptions which slightly restrict the robustness of the algorithm. The assumptions are listed here and subsequently expounded on later in this report. They are:

1. Design Variables are strictly positive
2. Only 2 design variables allowed (2-dimenstional problem)
3. Only inequality ($\leq$) constraints are allowed
4. The inequality constraints have strictly positive right hand sides

The function to be minimized is read as a symbolic object using the sympy [3] package in python. The sympy package allows symbolic differentiation of the passed in function which helps avoid numeric errors that may be observed if numeric differentiation was used instead. After getting the function as input, our implementation of SQP carries out the optimization according to the following steps:

1. Symbolically compute the Hessian, the quadratic approximation of the objective function, and the linearization of the constraints.
2. Substitute the current search point x^k into the expressions.
3. **Solve the descent direction subproblem.** The substituted quadratic approximation of the objective function and linearized constraints now form the descent direction subproblem that needs to be solved for the descent direction **d**. This QP subproblem can be solved using the simplex method.
    - This step is the most crucial problem in the SQP method and also represents the difference between SQP and Sequential Linear Programming (SLP) techniques which are of similar family but do not include the quadratic term in the Taylor expansion.
    - In our implementation, wrote a function to carry out Gauss-Jordan elimination for the simplex method to find the search direction. However, due to the complexities involved in writing a robust simplex solver from scratch, we also considered a solution that was based on extracting the linearized objective function from the simplex tableau as well as the linearized constraints using the simplex method implemented in scipy.optimize.linprog [4] to solve for **d**. We encountered problems when using this approach and decided to abandon this in the interest of time.
4. **Calculate the step size**. We implemented golden section search to solve the step size calculation subproblem. x^{k+1} expressed in terms of α and the computed search direction from step 3 is passed into a golden section function to determine the optimal step size for this search direction.
    - Golden section operates to minimize the descent function equation 19.
    - Golden section was initialized with a convergence threshold of 0.0001 and an initial bracket of $[0.0, 10.0]$.
5. Compute the next search point (design variables) according to equation 18
6. Repeat steps 1-5 until the normed change in search point (design variables) is less than $2\text{e}{-5}$.

References

[1] Arora, J. (2004). *Introduction to optimum design*. Elsevier.

[2] Pshenichny, B. & Danilin, Y. M. (1978). *Numerical methods in extremal problems*. Mir Publishers.

[3] Meurer, A., Smith, C. P., Paprocki, M., Čertík, O., Kirpichev, S. B., Rocklin, M., Kumar, A., Ivanov, S., Moore, J. K., Singh, S., Rathnayake, T., Vig, S., Granger, B. E., Muller, R. P., Bonazzi, F., Gupta, H., Vats, S., Johansson, F., Pedregosa, F., … Scopatz, A. (2017). Sympy: Symbolic computing in python. *PeerJ Computer Science*, *3*, e103. https://doi.org/10.7717/peerj-cs.103

[4] Virtanen, P., Gommers, R., Oliphant, T. E., Haberland, M., Reddy, T., Cournapeau, D., Burovski, E., Peterson, P., Weckesser, W., Bright, J., van der Walt, S. J., Brett, M., Wilson, J., Millman, K. J., Mayorov, N., Nelson, A. R. J., Jones, E., Kern, R., Larson, E., … SciPy 1.0 Contributors. (2020). SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python. *Nature Methods*, *17*, 261–272. https://doi.org/10.1038/s41592-019-0686-2