SEQUENTIAL QUADRATIC PROGRAMMING

Youssef Sherif Mansour

MAY 27, 2018

Introduction:

Sequential quadratic programming (SQP) is a one of the most recently developed optimization algorithms, it is also considered one of the best. It is very powerful because it can handle any degree of non-linearity in both the objective function and the constraints. Its main idea is to simplify the solution of a hard-non-linear problem by solving a sequence of easier quadratic problems which can be solved efficiently by a variety of techniques.

The method has theoretical basis related to two concepts – the first is the derivation of simultaneous non-linear equations that satisfies the KKT condition at the optimal point from the Lagrangian of the hard problem. The second concept is the solution of these equations using Newton's method. The second concept is the quadratic problem simplification in each iteration and it can be solved by many methods ranging from ones that are exact to others that are approximations but very fast. The main disadvantage in this method is that it needs several derivatives that needs to be calculated analytically before iterating over the sequence of quadratic problems. This makes SQP hard to apply to some large problems with many variables and constraints - and impossible to apply to problems that doesn't have a closed form of the objective function and constraints such as black box systems and some practical applications in control and reinforcement learning.

Theoretical derivation:

We consider a non-linear problem with only equality constraints to simplify the derivation. This is without any loss of generality as inequality constraints can be stated as equality constraints by adding slack or surplus variables.

So, we try to solve

$$\min f(\boldsymbol{X})$$

Subject to

$$h_k(\boldsymbol{X}) = 0 \quad k = 1,2, \dots \dots, p$$

The lagrangian is

$$L = f(\boldsymbol{X}) + \boldsymbol{\lambda}^T \boldsymbol{h}(\boldsymbol{X})$$

The KKT states that

$$\nabla L(\boldsymbol{X}^*) = 0 \; and \; \boldsymbol{h}(\boldsymbol{X}^*) = 0$$

Which leads to

$$\nabla f(\boldsymbol{X}^*) + \boldsymbol{\nabla h}(\boldsymbol{X}^*)^T \boldsymbol{\lambda} = 0$$

$$for\ simplicity\ let\ \boldsymbol{\nabla h}(\boldsymbol{X}^*) = A \qquad , where\ A\ size\ is\ (n \times p)$$

By combining the two conditions we get $n + p$ equations in $n + p$ unknowns. Since these equations are non-linear, they can't be solved easily. Thus, Newton's method to solve them. For convenience, the two equations are grouped as one equation

$$\boldsymbol{F}(\boldsymbol{Y}) = \boldsymbol{0}$$

Where

$$\boldsymbol{F} = \begin{Bmatrix} \nabla \boldsymbol{L} \\ \boldsymbol{h} \end{Bmatrix} \ , \boldsymbol{Y} = \begin{Bmatrix} \boldsymbol{X} \\ \boldsymbol{\lambda} \end{Bmatrix}$$

By applying Newton's method, we get

$$[\nabla \boldsymbol{F}]_j^T * \Delta \boldsymbol{Y}_j = -\boldsymbol{F}(\boldsymbol{Y}_j)$$

Where $j\ is\ the\ jth\ iteration$

By substituting F in the above equation, we get

$$\begin{bmatrix} \nabla^2 \boldsymbol{L} & \boldsymbol{H} \\ \boldsymbol{H}^T & \boldsymbol{0} \end{bmatrix}_j * \begin{bmatrix} \Delta \boldsymbol{X} \\ \Delta \boldsymbol{\lambda} \end{bmatrix}_j = - \begin{bmatrix} \nabla \boldsymbol{L} \\ \boldsymbol{h} \end{bmatrix}_j$$

Where H is a n x p matrix where the ith column is the gradient of the ith constraint.

By manipulating the first equation

$$\nabla^2 \boldsymbol{L}_j * \Delta \boldsymbol{X}_j + \boldsymbol{H}_j * \Delta \boldsymbol{\lambda}_j = -\nabla \boldsymbol{L}_j$$

$$\nabla^2 \boldsymbol{L}_j * \Delta \boldsymbol{X}_j + \boldsymbol{H}_j * (\boldsymbol{\lambda}_{j+1} - \boldsymbol{\lambda}_j) = -\nabla \boldsymbol{f}_j - \boldsymbol{H}_j \boldsymbol{\lambda}_j$$

Which leads to the system

$$\begin{bmatrix} \nabla^2 \boldsymbol{L} & \boldsymbol{H} \\ \boldsymbol{H}^T & \boldsymbol{0} \end{bmatrix}_j * \begin{bmatrix} \Delta \boldsymbol{X}_j \\ \boldsymbol{\lambda}_{j+1} \end{bmatrix} = - \begin{bmatrix} \nabla \boldsymbol{f} \\ \boldsymbol{h} \end{bmatrix}_j$$

This is now a linear system that can be solved to get the next point and the next lagrangian multipliers.

By taking a closer look to the above system we notice it is the same as solving the following quadratic function

$$\min_{\Delta X} Q = \nabla \boldsymbol{f}^T * \Delta \boldsymbol{X} + \frac{\boldsymbol{1}}{\boldsymbol{2}} * \Delta \boldsymbol{X}^T * \nabla^2 \boldsymbol{L} * \Delta \boldsymbol{X}$$

Subject to

$$\boldsymbol{h} + H^T \Delta \boldsymbol{X} = \boldsymbol{0}$$

By adding the inequality constraints

$$\boldsymbol{g} + G^T \Delta \boldsymbol{X} \leq \boldsymbol{0}$$

We now solve this subproblem for ΔX and then utilize the following derivative of the lagrangian function to get lambda from KKT condition

$$\boldsymbol{\nabla f} + \boldsymbol{\nabla^2 L} * \Delta \boldsymbol{X} + H\boldsymbol{\lambda} = \boldsymbol{0}$$

I am using two convergence measures one on the function value and the other on the point value. They are defined as

$$\|x_{i+1} - x_i\| < e1$$

$$\left|\frac{f(x_{i+1}) - f(x_i)}{f(x_i)}\right| < e_2$$

Chosen problem:

It is obvious that SQP requires a lot of analytical work before solving the problem. Thus, I chose the Rosen Brock problem constrained by a unit circle as the test problem. This problem I very challenging for many algorithms because of its shallow minimum found in a deep valley which makes many optimization algorithms miss the minimum and diverge.

The problem is defined as

$$\min_X f(x) = 100(x_2 - x_1^2)^2 + (1 - x)^2$$

Subject to

$$x_1^2 + x_2^2 \leq 1$$

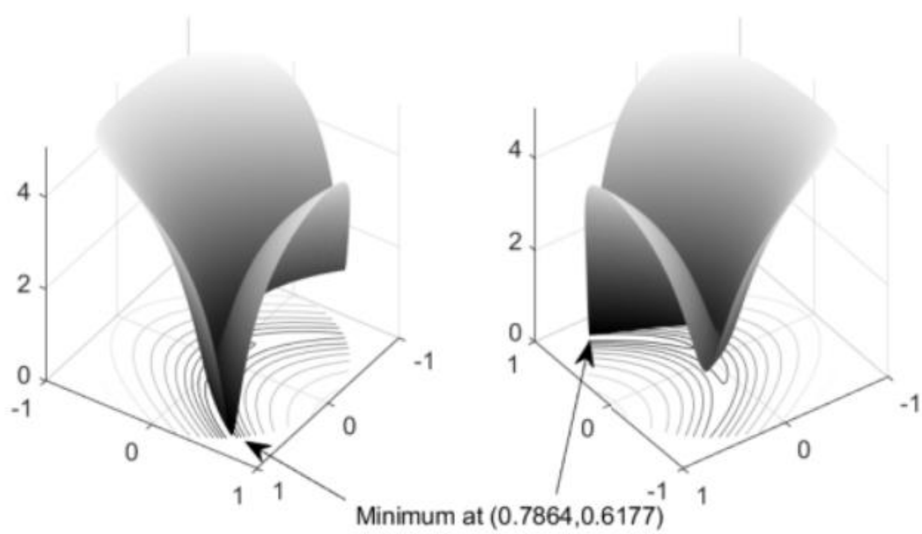


Figure 1 Rosen Brock function

To solve the problem, I first analytically solved for

1) ∇f which is the gradient of the function
2) ∇L^2 which is the hessian of the lagrangian of the function
3) ∇g which is the gradient of the inequality constraints

It is worth to note that this problem has no equality constraints but nonetheless my implementation accounts for it.

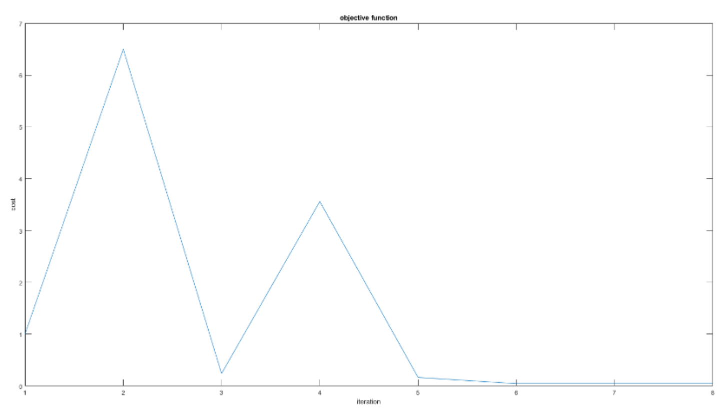
Results:

The algorithm is very fast it converges in 0.13 seconds taking 7 iterations to achieve a very high accuracy of $e_1 = 0.00001$ and $e_2 = 0.00001$

It gives an optimal point at $[0.7864 \quad 0.6177]^T$

With a function value $= 0.0457$

Objective function vs. Iterations



The path to the minimum