DOI: 10.5281/zenodo.14484115

Quasi-Newton Approximation of Non-Linear Programming Problem with Inequality Constraints

Idowu, Gbolahan A. & Osewa, Omodele M.

ABSTRACT

The study of non-linear programming (NLP) problems has wide applications in industry, science, agriculture, and engineering. Such applications include engineering design, control, data fitting, crop, and economic planning. However, most existing studies relating to NLP have been carried out without considering the effect of large-scale inequality constraints on the NLP problem despite their importance in real-life situations. In this paper, we consider NLP with large-scale inequality constraints where the system of equations obtained by the Karush-Kuhn Tucker (KKT) technique has no analytical solutions. Several methods in the literature were computationally expensive. A Quasi-Newton Approximation Scheme (QNAS) was adopted and implemented to obtain an approximate solution to NLP of this type. The QNAS performed better and converged faster than other approximation schemes in the literature. Real-life engineering problems resulting in NLP were solved, and the result obtained was compared with Newton's approximation scheme.

Keywords: *non-linear programming, karush-kuhn-tucker, quasi-newton, inequality constraints, aprproximation schemes*

INTRODUCTION

Optimization is an essential tool for finding the best posible solution or maximizing or minimizing a certain objective within a given set of constraints of linear and non-linear formulations to solve large problems accurately and study the solutions. Mathematical optimization or mathematical programming is the selection of a best element, with regard to some criterion, from some set of available alternatives (Liu And Wang,2022). It is generally divided into two subfields: discrete optimization and continuous optimization. Optimization problems of sorts arise in all quantitative disciplines from computer science and engineering to operations research and economics, and the development of solution methods has been of interest in mathematics for centuries (Liuzzi, Lucidi, Rinaldi & Vicente, 2019; Martin & Ning, 2021).

DOI: 10.5281/zenodo.14484115

DOI: 10.5281/zenodo.14484115

It assists in finding the answer that gives the best result; reaches the highest profit, output, or happiness; or obtains the lowest cost, waste, or discomfort. Usually, these problems entail making the most efficient use of available resources, including capital, time, machinery, personnel, inventory, and land and water. These problems have been solved by using different optimization methods from researchers worldwide. The theory and application of optimization is sometimes referred to as mathematical programming. Here the term programming does not refer to computer programming. The optimization theory provides a powerful framework for formulating the general optimization problem.

As shown in Figure 1, optimization problems can first be classified by their continuous and discrete variables (see, Biegher And Grossmann, 2004). The major problems for continuous optimization include linear programming (LP) and NLP. An important subclass of LP is the linear complementary problem (LCP), while the NLP includes quadratic programming (QP) and semidefinite programming (SP). An important distinction for the latter is whether the NLP problem is convex or nonconvex since the latter may give rise to multiple local optima. Another important distinction is whether the problem is assumed to be differentiable. As for discrete problems, they are first classified into mixed-integer linear programming (MILP) and MINLP.

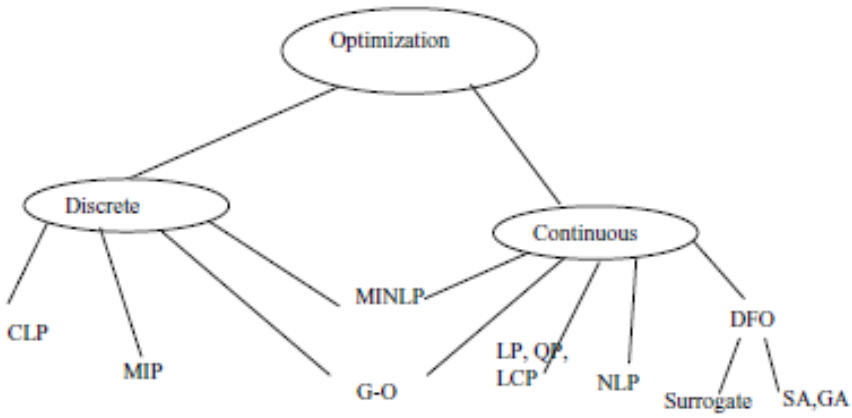

Fig. 1: Tree of classes of optimization problem

For the former, a particularly important case is when we have all the variables to be integers, leading to an integer programming (IP) problem. This problem, in turn, can be classified into many special problems (for example: assignments, traveling salesman, and so on.). The MINLP problem also leads to special problems, although the main distinction, like in the NLP problem, is whether its relaxation is convex or

DOI: 10.5281/zenodo.14484115

DOI: 10.5281/zenodo.14484115

nonconvex.

Regarding their formulation, discrete/continuous optimization problems when represented in algebraic form, correspond to mixed-integer optimization problems that have the following general form:

$$\begin{aligned} &Minimize: Z = f(x,y) \\ &subject\ to: \\ &h(x,y) = 0 \\ &g(x,y) \leq 0 \\ &x \in X, y \in \{0,1\}^m \end{aligned} \quad \ldots\ldots 1$$

where $f(x,y)$ is the objective function (for instance, cost),$h(x,y) = 0$ are the equations that describe the performance of the system (material balances, production rates), and $g(x,y) \leq 0$ are inequalities that define the specifications or constraints for feasible plans and schedules.

Numerous real-world issues, including those related to engineering design, chemical engineering, traffic, management, and economic policy, can be represented as nonlinear constrained optimization problems. The choice variables for nonlinear constrained optimization problems include real variables, integer variables, and discrete variables with discreteness (also known as discrete variables). A nonlinear constrained optimization issue can be mathematically expressed as follows:

$$Mininize: f(n_1, n_2, \ldots, n_t; r_1, r_1, \ldots, r_s)$$

Subject to:

$$g_j(n_1, n_2, \ldots, n_t; r_1, r_1, \ldots, r_s) \geq b_j, j = 1,2, \ldots, m$$

$$a_i \leq n_i \leq b_i, l_k \leq r_k \leq u_k, \quad \ldots\ldots 2$$

where n_i are integers or discrete variables, r_k are real (continuous) variables, $1 \leq i \leq t,\ 1 \leq k \leq s$.

Problem (2) is a very general optimization problem that is subject to general constraints, with real variables, integer variables and discrete variables. Clearly, Problem (2) is a Mixed-Integer Nonlinear Programming Problem (MINLP) and these are more difficult than so-called component redundancy allocation problems (Bashier, 2020).

Furthermore, many problems in Engineering lead to Non-Linear programming

DOI: 10.5281/zenodo.14484115

DOI: 10.5281/zenodo.14484115

Problems whose system of equations arises from their Karush Kuhn Tucker (KKT) conditions cannot be solved analytically. Over the years, we have been taught how to solve equations using various algebraic methods. The methods used include the elimination method and the substitution method. Algebraic methods that can also be used include the quadratic formula, completing the square, and factorization methods. In Linear Algebra, we are taught that solving systems of linear equations can be executed by using row reduction as an algorithm. However, when these methods fail, we use numerical methods to find approximate solution.

Numerical methods are used to find an approximate solutions of equations when real solutions can not be determined via algebraic methods. They construct successive approximations that converge to the solution of an equation or a system of equations. For continuous variable optimization, we consider (MIP) without discrete variables. A key characteristic of a Non-Linear problem (NLP) is whether it is convex or nonconvex, that is, it has a convex objective function and a convex feasible region. Convex feasible regions require $g(x, y)$ to be convex and $h(x, y)$ to be linear. If any NLP is a convex problem, then any local solution is also a global solution to the NLP. Moreover, this solution is unique if the objective function is strictly convex. On the other hand, the KKT conditions can only satisfy local optimality for nonconvex problems.Therefore, a more rigorous (and expensive) search procedure is required to find a global solution.

We first consider solvers based on Successive Quadratic Programming (SQP) as first solution technique to NLP as they allow the construction of a number of NLP algorithms based on Newton steps. Additionally, these solvers have been shown to require the fewest function evaluations to solve NLPs and they can be applied to a broad range of process engineering problems with different structure. SQP uses the equivalent of a Newton step to the KKT conditions of the nonlinear programming problem (Xu And Cheng, 2023; Zhang And Li, (2023)).

In addition to SQP methods, several NLP solvers have been developed and adapted for large scale problems. Generally,these methods require more function evaluations than SQP methods but they perform very well when interfaced to optimization modeling platforms such as AMPL, CUTE or GAMS, where function evaluations are cheap. All of these can be derived from the perspective of applying Newton steps to portions of the KKT conditions of (NLP)(Biegher And Grossmann,2004). LANCELOT (Andreani, Schuverdt And Secchin, 2024) is based on the solution of bound constrained sub-problems.

METHODOLOGY

This paper involves theoretical exploration and numerical computations, and

DOI: 10.5281/zenodo.14484115

DOI: 10.5281/zenodo.14484115

draws insights and ideas from related works to identify open problems and possible solutions. In the first part, theoretical foundation of solutions to nonlinear programming problems with constraints was considered while a numerical method for solving the problem that cannot be solved analytically is proposed in the second part. Furthermore, sample problems with known analytical solutions are compared with the method.

LAGRANGE MULTIPLIER

Consider NLP problem of the form:

Minimize $f(X)$

Subject to

$$g_j(x) \leq 0, j = 1,2,3,\ldots,m \quad \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots 3$$

The above problem can be solved using Lagrange multipliers method for solving the equality constrained problems. Therefore, we need to convert the inequality constraints to equality constraints by introducing some non-negative slack variables, say $s_j^2, j = 1,2,\ldots,m.$

The problem (3) now becomes:

Minimize $f(x)$

Subject to

$$g_j(x,s) = g_j(x) + s_j^2 = 0, j = 1,2,\ldots,m \quad \ldots\ldots\ldots\ldots\ldots\ldots\ldots 4$$

Where

$$s_j = \begin{bmatrix} s_1 \\ s_2 \\ \vdots \\ s_m \end{bmatrix}$$

is a vector of slack variables.

Now, this problem can be solved by using the Lagrange multiplier method. We can define the Lagrange function L, as

$$L(x,s,\lambda) = f(x) + \sum_{j=1}^{m} \lambda_j g_j(x,s) \quad \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots 5$$

DOI: 10.5281/zenodo.14484115

DOI: 10.5281/zenodo.14484115

where

$$x = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix} \text{ is the decision vector}$$

$$s = \begin{bmatrix} s_1 \\ s_2 \\ \vdots \\ s_m \end{bmatrix} \text{ is the slack vector}$$

$$and\ \lambda = \begin{bmatrix} \lambda_1 \\ \lambda_2 \\ \vdots \\ \lambda_m \end{bmatrix} \text{ is the Lagrange Multipliers Vector}$$

Now we write the necessary conditions as

$$\frac{\partial}{\partial x_i} L(x, s, \lambda) = \frac{\partial f(x)}{\partial x_i} + \sum_{j=1}^{m} \lambda_j \frac{\partial g_j(x,s)}{\partial x_i} = 0 \ldots\ldots\ldots\ldots 6$$

$$\frac{\partial}{\partial s_j} L(x, s, \lambda) = 2\lambda_j s_j = 0 \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots 7$$

$$\frac{\partial}{\partial \lambda_j} L(x, s, \lambda) = g_j(x, s) = g_j(x) + s_j^2 = 0 \ldots\ldots\ldots\ldots 8$$

Where $i = 1,2,\ldots,n$ and $j = 1,2,\ldots,m$

Now we have $(n + 2m)$ equations in $(n + 2m)$ variables and hence can solve the system of equations for the optimum solution vector x^*.

This concept was revised by Kuhn and Tucker as described in the conditions as the necessary conditions to be satisfied at relative minimum if $f(x)$ with inequality constraints $g_j(x) \leq 0$, expressed as

$$\frac{\partial f}{\partial x_i} + \sum_{j=1}^{m} \lambda_j \frac{\partial g_j}{\partial x_i} = 0\ i = 1,2,\ldots,n \text{and } \lambda_j \geq 0 \ldots\ldots\ldots\ldots\ldots\ldots 9$$

where λ_j are the Lagrange multipliers. If the set of active constraints is not known, then the Kuhn-Tucker conditions can be stated as follows for the case minimize $f(x)$, subject to $g_j(x) \leq 0$:

$$\frac{\partial f(x)}{\partial x_i} + \sum_{j=1}^{m} \lambda_j \frac{\partial g_j(x)}{\partial x_i} = 0 \text{ i} = 1,2,\ldots,\text{n} \quad \ldots\ldots\ldots\ldots\ldots\ldots 10$$

DOI: 10.5281/zenodo.14484115

DOI: 10.5281/zenodo.14484115

$$\sum_{j=1}^{m} \lambda_j \frac{\partial g_j(x)}{\partial x_i} = 0 \text{ i} = 1,2,\ldots,\text{n} \quad \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots 11$$

$$\begin{aligned} &\lambda_j g_j(x) = 0 && j = 1,2,\ldots,m \\ &g_j(x) \leq 0, && j = 1,2,\ldots,m \\ &\lambda_j \geq 0, j = 1,2,..,m \end{aligned} \quad \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots 12$$

Equations (10), (11) and (12) are called Optimality condition, Feasibility condition and complementary slackness property respectively. X is a KKT point if it satisfies all above properties. The constraint qualification and the corresponding values of the Lagrange multipliers will obey the following relations:

1. λ_j is non-negative $(\lambda_j \geq 0)$ for Minimization of f with $g_j \leq 0$.
2. λ_j is non-negative $(\lambda_j \geq 0)$ for Maximization of f with $g_j \geq 0$.
3. λ_j is non-positive $(\lambda_j \leq 0)$ for Maximization of f with $g_j \leq 0$.
4. λ_j is non-positive $(\lambda_j \leq 0)$ for Minimization of f with $g_j \geq 0$.

THE SUFFICIENT CONDITION FOR OPTIMIZATION PROBLEM

The KKT condition are sufficient conditions if $f(x)$ is convex, and feasible space is convex. Suppose we have constructed Lagrange function as

$$L = f(x) + \sum_{j=1}^{m} \lambda_j \left[g_j(x) + s_j^2 \right]$$

$$\lambda_j \geq 0 \quad \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots 13$$

x^* is relative minima point if $\nabla^2 L|_{x^*}$ is positive definite then x^* is relative maxima points if $f(x)$ is concave and feasible space is convex.

For a general optimization problem with both equality and inequality constraints.

Minimize $f(x)$
Subject to
$g_j(x) \leq 0 (j = 1,2,\ldots,m)$
$h_k(x) = (k = 1,2,\ldots,l)$
$X = (x_1, x_2, \ldots, x_n)^T$

If X^* is a regular point, then X^* is also a local minima of f, if the following conditions are satisfied. We construct the Lagrange function as $L = f(x) + \sum_{j=1}^{m} \lambda_j g_j + \sum_{k=1}^{l} \beta_k h_k$.

DOI: 10.5281/zenodo.14484115

DOI: 10.5281/zenodo.14484115

The optimality condition is as follows:

$$\frac{\partial f}{\partial x_i} + \sum_{j=1}^{m} \lambda_j \frac{\partial g_j(x)}{\partial x_i} + \sum_{k=1}^{l} \beta_k \frac{\partial h_k(x)}{\partial x_i}, i = 1,2,\ldots,n \quad \ldots\ldots 14$$

Feasibility condition:

$$g_j(x) \leq 0, h_k = 0, j = 1,2,\ldots,m, k = 1,2,\ldots,l \quad \ldots\ldots 15$$

Slack property:

$$\lambda_j g_j(x) = 0 \; j = 1,2,\ldots,m \quad \ldots\ldots 16$$

Non-negativity:

$$\lambda_j \geq 0 \; j = 1,2,\ldots,m \quad \ldots\ldots 17$$

β_k is unrestricted in sign, $k = 1,2,\ldots,l$

Prior to now, we made the assumption that calculating the gradient $\nabla f(x)$ or the Hessian matrix $Hf(x)$ for a KKT would be simple and inexpensive. However, in actuality, this is not always the case If $f: \mathbb{R}^n \to \mathbb{R}$ with n large, then computing, $Hf(x)$ requires finding many derivatives, which is expensive even when we can take partial derivatives of f easily. Also, If f is not given to us explicitly, then we cannot compute derivatives of f directly (unless we use some approximation algorithm for derivatives). The secant approach as a solution to the second of these issues in one dimension. Here, we estimated the derivative using the values at the final two locations.

Now, for the n-dimensional case, finding the solution to a system of equations $g(x) = 0$ for a function $g: \mathbb{R}^n \to \mathbb{R}^n$. It requires a determination of the Jacobian in order to approximate Newton's method ∇g; in other words, a linear approximation of g. Here a p[roblem is encountered as the values of g at the last two points are not enough to build a complete linear approximation of g. (In general, such an approximation would need at least $n + 1$ points to be defined.)

Instead of discarding the matrix totally as the secant technique does, our approach to this problem is to keep a matrix that roughly approximates the Jacobian and update it each time we learn new information. Intuitively, when comparing the new value

DOI: 10.5281/zenodo.14484115

DOI: 10.5281/zenodo.14484115

$g(x^{(k+1)})$ to the previous value $g(x^{(k)})$, we can learn something about the rate of change of g in the direction of $x^{(k+1)} - x^{(k)}$, but nothing about the rate of change of g in other directions.

We can build on the justification behind Newton's method to figure out how to do this. With Newton's method, when we are at a point $x^{(k)}$, we make a linear approximation to g.

$$g(x) \approx g(x^{(k)}) + \nabla g(x^{(k)})(x - x^{(k)}). \quad \text{.....18}$$

Then, let $x^{(k+1)}$ be the point where the linear approximation equals 0.

Now suppose that instead of computing the Jacobian $\nabla g(x^{(k)})$, we somehow found an approximation D_k: an $n \times n$ matrix that is our best guess at the partial derivatives of g. Just as with Newton's method, we make a linear approximation—

$$g(x) \approx g(x^{(k)}) = D_k(x - x^{(k)}). \quad \text{.....19}$$

We let $x^{(k+1)}$ be the point where the linear approximation equals 0.

In practice, $g(x^{(k+1)})$ doesn't end up being 0: if everything is going well, it should be closer to 0 than $g(x^{(k)})$ was, but the linear approximation is not exact. This is new information, and we should replace D_k by a new matrix D_{k+1} that takes this new information into account. It's natural to ask that D_{k+1} predict correctly what D_k was wrong about. We now know the value of $g(x^{(k+1)})$, so we can ask that if we replace D_k by D_{k+1} in our previous approximation, it should give that value exactly. That is, we should have—

$$g(x^{(k)}) + D_{k+1}(x^{(k+1)} - x^{(k)}) = g(x^{(k+1)}) \quad \text{.....20}$$

where we previously had

$$g(x^{(k)}) + D_k(x^{(k+1)} - x^{(k)}) = 0 \text{.....21}$$

Second, we ask that in every direction orthogonal to direction $x^{(k+1)} - x^{(k)}$, we still make the same prediction. After all, we didn't go in such directions, so we can't have learned anything new about them. So we require that

$$D_k y = D_{k+1} y \text{whenever} y \cdot (x^{(k+1)} - x^{(k)}) = 0 \text{.....22}$$

DOI: 10.5281/zenodo.14484115

DOI: 10.5281/zenodo.14484115

Together, equations (20), (21) and (22) characterize the change from D_k to D_{k+1}. It is easier to reason in term of the "update" from D_k to D_{k+1}: the difference between D_{k+1} and D_k. If we denote this difference by $U_k = D_{k+1}$, and define $b^{(k)} = x^{(k+1)} - x^{(k)}$, then by taking the difference of (20) and (21), we get

$$Y_k b^{(k)} = g(x^{(k+1)}) \quad \text{.....23}$$

While (22) can be written as $U_k y = 0$ when every $b^{(k)} = 0$. If this matrix exists, it must be unique, because any input y can be written as $y = y^{\parallel} + y^{\perp}$ where $y^{\parallel}$ is a multiple of $b^{(k)}$, and $y^{\parallel}$ is orthogonal to $b^{(k)}$; the equations above define what U_k does to $y^{\parallel}$ and $y^{\perp}$, and therefore they determine what U_k does to y.

One way to get a function $f(y)$ that has this property is to scale $g(x^{(k+1)})$ (the desired nonzero ouput) propertionally to the dot product $y \cdot b^{(k)}$. That is, to set

$$f(y) = \frac{y \cdot b^{(k)}}{b^{(k)} \cdot b^{(k)}} g(x^{(k+1)}). \quad \text{.....24}$$

This resulted into a linear function, so that there actually is some matrix U_k such that $f(y) = U_k y$. We can see this by rewriting $y \cdot b^{(k)}$ as $(b^{(k)})^T y$, so that

$$f(y) = \frac{g(x^{(k+1)})(b^{(k)})^T y}{b^{(k)} \cdot b^{(k)}} \quad \text{.....25}$$

This tells us that the unique update matrix U_k that does the job is

$$U_k = \frac{g(x^{(k+1)})(b^{(k)})^T}{b^{(k)} \cdot b^{(k)}} \quad \text{.....26}$$

Adding this matrix to D_k to get D_{k+1} is called a rank-one update, because the rank of the matrix U_k is 1: its columns are all multiple og $g(x^{(k+1)})$. (And its rows are all multiples of $(b^{(k)})^T$). This makes it a "minimal" change from D_k to D_{k+1} in some sense, and using a rank-one matrix also has some theoretical benefits

BROYDEN'S METHOD

This starts with some starting point $x^{(0)}$; we also need a matrix D_0 to start as our approximation to $\nabla g(x^{(0)})$. (If computing derivatives of g is expensive but not impossible, we could set D_0 equal to the Jacobian, since we only need to do this once. Otherwise, we could set D_0 to the identity matrix because that's simple to work with.)

To compute $x^{(k+1)}$ and D_{k+1} from $x^{(k)}$ and D_k, we:

DOI: 10.5281/zenodo.14484115

DOI: 10.5281/zenodo.14484115

1. Solve $g(x^{(k)}) + D_k(x^{(k+1)} - x^{(k)}) = 0$ for $x^{(k+1)}$; equivalently, set $x^{(k+1)} = x^{(k)} - D_k^{-1}g(x^{(k)})$.
2. Set $D_{k+1} = D_k + \frac{g(x^{(k+1)})(b^{(k)})^T}{b^{(k)} \cdot b^{(k)}}$, where $b^{(k)} = x^{(k+1)} - x^{(k)}$.

From (Broyden, 1969; Martinez, 2000), when using Newton's method computing $\nabla g(x^{(k)})^{-1}$ is avoided; instead, to solve a system of linear equations at each step is preferred. With Broyden's method. however, the iterative step

$$x^{(k+1)} = x^{(k)} - D_k^{-1}g(x^{(k)}) \quad \text{.....27}$$

is actually much faster to use. This is because we do not have to recompute the inverse from scratch after a rank-one update: we can find D_{k+1}^{-1} directly from D_k^{-1}, by a result known as the Sherman-Morrison formula.

The Sherman-Morrison formula says that if A is an $n \times n$ invertible matrix and $u, v \in \mathbb{R}^n$, then $(A + uv^T)^{-1}$ can be computed from A^{-1} as

$$(A + uv^T)^{-1} = A^{-1} - \frac{A^{-1}uv^TA^{-1}}{1+v^TA^{-1}u} \text{.....28}$$

We can use this to compute D_{k+1}^{-1} from D_k^{-1} if we set $A = D_k$, $u = g(x^{(k+1)})$, and $v = \frac{b^{(k)}}{b^{(k)} \cdot b^{(k)}}$. Since the Sherman-Morrison formula never requires multiplying two matrices together (we can write $A^{-1}uv^TA^{-1}$ as $A^{-1}u$ times v^TA^{-1}), applying it is much faster even than solving a system of linear equations.

In this way, we avoid ever having to compute D_k explicitly: only D_k^{-1} is needed. After simplifying the Sherman-Morrison formula to our specific needs, we can resummarize Broyden's method as:

1. Set $x^{(k+1)} = x^{(k)} - D_k^{-1}g(x^{(k)})$.
2. In terms of the vectors $b^{(k)} = x^{(k+1)} - x^{(k)} = -D_k^{-1}g(x^{(k)})$ and $c^{(k)} = D_k^{-1}g(x^{(k+1)})$, set

$$D_{k+1}^{-1} = D_k^{-1} - \frac{c^{(k)}((b^{(k)})^TD_k^{-1})}{b^{(k)} \cdot (b^{(k)}+c^{(k)})} \text{.....29}$$

Even if the evaluation of $\nabla g(x^{(k)})$ were inexpensive, this method would be faster than Newton's method when the number of dimensions n is large.

DOI: 10.5281/zenodo.14484115

DOI: 10.5281/zenodo.14484115

RESULT

Implementation for solving nonlinear programming problems with equality constraints

In general, consider optimization problem

$$\begin{aligned} &\text{Minimize} && f(x_1, x_2, x_, \ldots, x_n) \\ &\text{Subjectto} && g_1(x_1, x_2, x_, \ldots, x_n) = b_1 \\ & && g_2(x_1, x_2, x_, \ldots, x_n) = b_2 \quad \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots 30 \\ & && \vdots \\ & && g_m(x_1, x_2, x_, \ldots, x_n) = b_m \end{aligned}$$

The Lagrangian function L is constructed as

$$L(x_1, x_2, \ldots, x_n, \lambda_1, \lambda_2, \ldots, \lambda_m) = f(x_1, x_2, \ldots, x_n) + \sum_{j=1}^{m} \lambda_j (g_j - b_j), j = 1,2, \ldots, m$$

Where $\lambda_1, \lambda_2, \ldots, \lambda_m$ are Lagrange multipliers, n is the number of variables and m is the number of constraints.

Now, by treating L as a function of $n + m$ unknowns, x_i and λ_j, where $i = 1,2, \ldots, n$ and $j = 1,2, \ldots, m$, the necessary conditions for the extremum of L are given by

$$\frac{\partial L}{\partial x_i} = \frac{\partial f}{\partial x_i} + \sum_{j=1}^{m} \lambda_j \frac{\partial g_j}{\partial x_i} = 0 \; i = 1,2, \ldots, n \quad \ldots\ldots\ldots\ldots\ldots\ldots 31$$

$$\frac{\partial L}{\partial \lambda_j} = g_j(x) = 0 \; j = 1,2, \ldots, m, \lambda_1, \lambda_2, \ldots, \lambda_m \geq 0 \quad \ldots\ldots\ldots\ldots 32$$

The Jacobian matrix of the necessary conditions is computed as

$$\nabla^2 L(x, \lambda) = J(x) = \begin{pmatrix} \nabla_{xx}^2 L(x, \lambda) & \nabla g(x) \\ \nabla g(x)^T & 0 \end{pmatrix}$$

The Jacobian matrix is $(m + n) \times (m + n)$ matrix. Where n and m are numbers of variables and constraints respectively and also assume $m < n$. $\nabla_{xx}^2 L(x, \lambda)$ and $\nabla g(x)$ are $n \times m$ matrices.

Then we can use Newton's method for the solution

DOI: 10.5281/zenodo.14484115

DOI: 10.5281/zenodo.14484115

$$J(x^{(k)})\delta x^{(k)} = -f(x^{(k)}) \quad \text{.....34}$$

Where $\delta x^{(k)} = x^{(k+1)} - x^{(k)}$. Hence, we have

$$\begin{pmatrix} \nabla^2_{xx} L(x,\lambda) & \nabla g(x) \\ \nabla g(x)^T & 0 \end{pmatrix} \begin{pmatrix} \delta x \\ \delta \lambda \end{pmatrix} = -\begin{pmatrix} \nabla_x L(x,\lambda) \\ g(x) \end{pmatrix} \text{.....35}$$

Implementation for solving nonlinear programming problems with inequality constraints

Consider the NLPP below

Minimize $f(X)$

Subject to $g_1(X) \leq 0\ j = 1,2,\dots,m; X = (x_1, x_2, \dots, x_n)^T \in \mathbb{R}^n$

After introducing m positive slack variables s_j to the inequality constraints, we get

$$g_j(X) + s_j^2 = 0\ j = 1,2,\dots,m$$

We can generate the Lagrange function for the problem as

$$L(x_1, x_2, \dots, x_n, s_1, s_2, \dots, s_m, \lambda_1, \lambda_2, \dots, \lambda_m) = f(x_1, x_2, \dots, x_n)$$

$$+ \sum_{j=1}^{m} \lambda_j \left(g_j(x_1, x_2, \dots, x_n) + s_j^2 \right)$$

λ_j $(j = 1,2,\dots,m)$ are Lagrange's multipliers

The necessary conditions for its relative minimum are:

$$\frac{\partial L}{\partial x_i} = \frac{\partial f(x)}{\partial x_i} + \sum_{j=1}^{m} \lambda_j \frac{\partial g(x)}{\partial x_i} = 0\ i = 1,2,\dots,n\ ; j = 1,2,\dots,m$$

$$\frac{\partial L}{\partial \lambda_j} = g_j(x) + s_j^2 = 0\ j = 1,2,\dots,m$$

$$\frac{\partial L}{\partial s_j} = 2\lambda_j s_j = 0\ j = 1,2,\dots,m$$

DOI: 10.5281/zenodo.14484115

DOI: 10.5281/zenodo.14484115

where n and m are number of variables and number of constrains respectively. We have $n + 2m$ equations and $n + 2m$ numbers of unknowns.

The above conditions are called the Karush-Kuhn -Turker (KKT) conditions.

We can apply Newton's method to solve the problem. This Newton's method is of the form

$$\Delta^2 L(x, \lambda_j, s_j)\delta(x) = -\Delta L(x, \lambda_j, s_j)$$

Where $\delta(x) = x^{(k+1)} - x^{(k)} x^{(k+1)} = x^{(k)} + \delta(x)$

$$\Delta L(X, \lambda_j, s_j) = \begin{pmatrix} \frac{\partial L}{\partial x_i} \\ \\ \frac{\partial L}{\partial \lambda_j} \\ \\ \frac{\partial L}{\partial s_j} \end{pmatrix}$$

$$\delta(X) = \begin{pmatrix} \delta X \\ \delta\lambda_j \\ \delta s_j \end{pmatrix} j = 1,2,\ldots,m$$

$$X = \begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{pmatrix}$$

$$\Delta^2 L(X, \lambda_j, s_j) = \begin{pmatrix} \Delta^2_{x_i} L(X, \lambda_j, s_j) & \Delta_{x_i} g(X) & 0 \\ \Delta_{x_i} g(X)^T & 0 & 2Is_j \\ 0 & 2Is_j & 2I\lambda_j \end{pmatrix}$$

When $\Delta^2_{x_i} L(X, \lambda_j, s_j)$ is $n \times n$ matrix $\Delta_{x_i} g(X)$ is $n \times m$ matrix

Hence, the Newton's Method becomes

$$\begin{pmatrix} \Delta^2_{x_i} L(X, \lambda_j, s_j) & \Delta_{x_i} g(X) & 0 \\ \Delta_{x_i} g(X)^T & 0 & 2Is_j \\ 0 & 2Is_j & 2I\lambda_j \end{pmatrix} \begin{pmatrix} \delta X \\ \delta\lambda_j \\ \delta s_j \end{pmatrix} = -\begin{pmatrix} \Delta_{x_i} L(X, \lambda_j, s_j) \\ \Delta_{\lambda_j} L(X, \lambda_j, s_j) \\ \Delta_{s_j} L(X, \lambda_j, s_j) \end{pmatrix}$$

We will now describe the process of solving the system of nonlinear equations derived from the KKT conditions of NLPP with inequality constrained.

DOI: 10.5281/zenodo.14484115

DOI: 10.5281/zenodo.14484115

1. Let $X^{(0)} = \left(x_i^{(0)}, \lambda_j^{(0)}, s_j^{(0)}\right)^T \; i = 1,2,\ldots,n \; ; \; j = 1,2,\ldots,m$

be a given initial vector.

2. Calculate $J(X^{(0)}) = \Delta^2 l(X^{(0)})$ and $F(X^{(0)}) = \Delta L(X^{(0)})$
3. We now need to calculate the vector

$$\delta(X^{(0)}) = \begin{pmatrix} \delta x_i \\ \delta \lambda_j \\ \delta s_j \end{pmatrix} \; i = 1,2,3,\ldots,n \; ; \; j = 1,2,3,\ldots,m$$

In order to calculate $\delta(X^{(0)})$, we solve the system $J(X^{(0)})\delta(X^{(0)}) = -F(X^{(0)})$, using Gaussian Elimination.

4. Once $\delta(X^{(0)})$ is calculated, we can proceed to finish the first iteration by calculating $X^{(1)}$. We have $X^{(1)} = X^{(0)} + \delta(X^{(0)})$
5. Once we have gotten $X^{(1)}$, we repeat the process until $X^{(k)}$ converges to X^*. Which means X^* is the solution to the system $F(X) = 0$.

A set of vectors converges when

$$\| X^{(k+1)} - X^{(k)} \| = 0 \text{ That is;}$$

$$\| X^{(k+1)} - X^{(k)} \| = \sqrt{\left(x_1^{(k+1)} - x_1^{(k)}\right)^2 + \cdots + \left(x_n^{(k+1)} - x_n^{(k)}\right)^2} = 0$$

CONVERGENCY

With the initial guess $X^{(0)}, X^{(1)}$ is calculated. We repeat the process again, until $X^{(k)}$ converges to $\bar{X}$. This indicates we have reached the solution to the system.

When a set of vectors converges, the norm

$$\| X^{(k+1)} - X^{(k)} \| = 0 \text{ This mean that}$$

$$\| X^{(k+1)} - X^{(k)} \| = \sqrt{\left(x_1^{(k+1)} - x_1^{(k)}\right)^2 + \cdots + \left(x_n^{(k+1)} - x_n^{(k)}\right)^2} = 0$$

Newton's method converges quadratically. Quadratic convergence means that the square of the error at one iteration is proportional to the error at the next iteration.

A sequence P_n is said to be converge quadratically if P_n converges to P with order $\alpha = 2$ such that

DOI: 10.5281/zenodo.14484115

DOI: 10.5281/zenodo.14484115

$$\lim_{n\to\infty} \frac{|P_{n+1}-P|}{|P_n-P|}\alpha = \beta \quad \text{.....36}$$

When β is a positive constant.

For Newton's method to converge quadratically, the initial vector $X^{(0)}$ must be sufficiently close to the solution of the system $F = 0$, which is denoted by $\bar{X}$. As well, the Jacobian matrix must exist. The goal of this proof is to show that:

$$\frac{\|X^{(k+1)}-\bar{X}\|}{\|X^{(k)}-\bar{X}\|^2} = \beta \quad \text{.....37}$$

then

$$\begin{aligned} \| e^{(k+1)} \| &= \| X^{(k+1)} - \bar{X} \| \\ &= \| X^{(k)} - J(X^{(k)})^{-1}F(X^{(k)}) - \bar{X} \| \end{aligned}$$

If we set $\| e^{(k)} \| = \| X^{(k)} - \bar{X} \|$ then we have

$$\| e^{(k+1)} \| = \| e^{k} - J(X^{(k)})^{-1}F(X^{(k)}) \|$$

Let the second-order Taylor series be:

$$F(X^{(k)}) \approx F(\bar{X}) + Je^{(k)} + \frac{1}{2}(e^{(k)})^T H(e^{(k)})$$

where $J = J(X^{(k)})$ and H is the Hessian tensor, which is similar to the Hessian matrix.

Wen then have to multiply each side of the Taylor's series by J^{-1}

$$\begin{aligned} J^{-1}(F(X^{(k)})) &\approx J^{-1}\left[F(\bar{X}) + Je^{(k)} + \tfrac{1}{2}(e^{(k)})^T H(e^{(k)})\right] \\ &= e^{(k)} + \frac{J^{-1}}{2}(e^{(k)})^T H(e^{(k)}) \end{aligned} \quad \text{...39}$$

Using (38) and (39) we obtain

$$\begin{aligned} \| X^{(k+1)} - \bar{X} \| &= \| e^{(k+1)} \| \\ &= \| \frac{J^{-1}}{2}(e^{(k)})^T H(e^{(k)}) \| \quad \text{.....40} \\ &\leq \frac{\|J^{-1}\|\|H\|}{2} \| e^{(k)} \|^2 \end{aligned}$$

DOI: 10.5281/zenodo.14484115

DOI: 10.5281/zenodo.14484115

QUASI-NEWTON METHOD

$$X^{(k+1)} = X^{(k)} - A_k^{-1}F(X^{(k)})$$

Where $X = (X_i, \lambda_j, s_j)^T$ This is defined as Broyden's iterative procedure.

In (Floudas And Pardalos, 2008), A_{k+1} is define as

$$A_{k+1} = A_k + \frac{F(X^{(k+1)})(b^{(k)})^T}{b^{(k)T} \cdot b^{(k)}} \quad \text{.....41}$$

Where $b^{(k)} = X^{(k+1)} - X^{(k)}$

From equation (38) we have

$$A_k\left(X^{(k+1)} - X^{(k)}\right) = -F(X^{(k)})$$

$$A_k \cdot \delta^k = -F(X^{(k)}) \quad \text{.....42}$$

However, Broyden's method involves computation of A_k^{-1}, not just A_k, which brings us to the next theorem.

Theorem 1 (Sherma-Morrison formula)

If D is an $n \times n$ invertible matrix and $U, V \in \mathbb{R}^n$ are vectors, then $(D + UV^T)^{-1}$ can be computed as

$$(D + UV^T)^{-1} = D^{-1} - \frac{D^{-1}UV^TD^{-1}}{1+V^TD^{-1}U} \quad \text{.....42}$$

We can use this to approximate A_{k+1}^{-1} from A_k^{-1} if we set $D = A_k, U = F(X^{(k+1)})$ and $V = \frac{b^{(k)}}{b^{(k)T}} \cdot b^{(k)}$

Applying Sherman-Morrison Formula, we can re-summarize Broyden's method as:

- Set $X^{(k+1)} = X^{(k)} - A_k^{-1}F(X^{(k)})$
- Let the vector $b^{(k)} = X^{(k+1)} - X^{(k)}$ and $C^{(k)} = A_k^{-1}F(X^{(k+1)})$, set

$$A_{k+1}^{-1} = A_k^{-1} - \frac{c^{(k)}\left((b^{(k)})^T A_k^{-1}\right)}{(b^{(k)})^T \cdot (b^{(k)} + c^{[}(k)])}$$

DOI: 10.5281/zenodo.14484115

DOI: 10.5281/zenodo.14484115

The steps of Broyden's method are described below:

1. Let $X^{(0)} = \left(x_i^{(0)}, \lambda_j^{(0)}, s_j^{(0)}\right)\ i = 1,2,\ldots,n\ \ j = 1,2,\ldots,m$ be initial vector given
2. Calculate $F(X^{(k)})$
3. We now compute A_0^{-1}. If we do not have enough information to compute A_0 directly, Broyden's method permits us to let $A_0 = J(X^{(0)})$, which implies that $A_0^{-1} = J(X^{(0)})^{-1}$
4. Calculate $X^1 = X^{(0)} - A_0^{-1}F(X^{(0)})$
5. Calculate $F(X^{(1)})$
6. Calculate $C^{(0)} = A_0^{-1}F(X^{(1)})$ and $b^{(0)} = x^{(1)} - X^{(0)}$.
7. Compute $(b^{(0)})^T \cdot A_0^{-1}$ and $b^{(0)} + c^{(0)}$
8. Compute

$$A_1^{-1} = A_0^{-1} - \frac{c^{(0)}\left((b^{(0)})^T \cdot A_0^{-1}\right)}{(b^{(0)})^T \cdot (b^{(0)} + c^{(0)})}$$

9. Take A_1^{-1} that we found in step 8 , and calculate
   $X^{(2)} = X^{(1)} - A_1^{-1}F(X^{(1)})$
10. Continue the process until we converge to $\bar{X} = X^{(k)} = X^{(k+1)}$. This shows that $\bar{X}$ is the solution of the system.

Convergence of quasi-Newton's method

Quasi-Newton's method converges super linearly unlike Newton's method which converges quadratically. That is

$$\lim_{k\to\infty} \frac{\|X^{(k+1)} - P\|}{\|X^{(k)} - P\|} = 0 \quad \ldots\ldots 43$$

where P is the solution to $F(X) = 0$, and $X^{(k)}$ and $X^{(k+1)}$ are successive approximations to P. The proof is similar to the proof of Newton's method.

NUMERICAL VALIDATION

Some NLPP are solved analytically and compared with solutions of Newton's Method and Quasi-Newton's (Broyden's Method) to confirm the accuracy of the method.

DOI: 10.5281/zenodo.14484115

DOI: 10.5281/zenodo.14484115

ANALYTIC SOLUTION OF THE PROBLEMS

Problem 1

Minimize $Z = 9 - 8x_1 - 6x_2 - 4x_3 + 2x_1^2 + 2x_2^2 + x_3^2 + 2x_1x_2 + 2x - 2x_3$
Subjectto: $x_1 + x_2 + 2x_3 \leq 3$

The Lagrange function $L(x, \lambda, s)$ is constructed as

$$L(x, \lambda, s) = 9 - 8x_1 - 6x_2 - 4x_3 + 2x_1^2 + 2x_2^2 + x_3^2 + 2x_1x_2 + 2x - 2x_3 + \lambda(x_1 + x_2 + 2x_3 + s^2)$$

where $\lambda \geq 0$ and s is the slack variable

$$\frac{\partial L}{\partial x_1} = -8 + 4x_1 + 2x_2 + 2x_3 + \lambda = 0 \quad \ldots\ldots 44$$

$$\frac{\partial L}{\partial x_2} = -6 + 4x_2 + 2x_1 + \lambda = 0 \quad \ldots\ldots 45$$

$$\frac{\partial L}{\partial x_3} = -4 + 2x_3 + 2x_1 + 2\lambda = 0 \quad \ldots\ldots 46$$

$$\frac{\partial L}{\partial \lambda} = x_1 + x_2 + 2x_3 + s^2 = 0 \quad \ldots\ldots 47$$

$$\frac{\partial L}{\partial s} = 2\lambda s = 0 \quad \ldots\ldots 48$$

The above conditions are called KKT conditions

Case 1

From equation (48) if $\lambda = 0$ and $s \neq 0$ then $x_1 = 6$, $x_2 = -6$ and $x_3 = 10$ which contradict the constraint. Therefore, the point (6,-6,10) is not a minimum point of Z

Case 2

From equation (48), if $\lambda \neq 0$ and $s = 0$. Then we have the following from equations (44), (45), (46) and (47)

$$\lambda = 8 - 4x_1 - 2x_2 - 2x_3$$
$$\lambda = 6 - 4x_2 - 2x_1$$
$$\lambda = 2 - x_1 - x_3$$
$$x_1 + x_2 + 2x_3 = 0$$

DOI: 10.5281/zenodo.14484115

DOI: 10.5281/zenodo.14484115

The above system of equations is solved analytically to give $x_1 = \frac{7}{3}, x_2 = \frac{1}{9}, x_3 = \frac{-11}{9}$ and $\lambda = \frac{8}{9}$

This solution satisfies all the KKT condition. Therefore, point $\left(\frac{7}{3}, \frac{1}{9}, \frac{-11}{9}\right)$ is a minimum point of Z.

Problem 2

Minimize $f(x, y) = \frac{5}{xy^2}$

Subjectto: $g(x, y) = x^2 + y^2 - 1 = 0$

The Lagrange function is

$$L(x, y, \lambda) = \frac{5}{xy^2} + \lambda(x^2 + y^2 - 1)$$

The necessary conditions are

$$\frac{\partial L}{\partial x} = -5x^{-2}y^{-2} + 2x\lambda = 0 \quad \text{.....49}$$

$$\frac{\partial L}{\partial y} = -10x^{-1}y^{-3} + 2y\lambda = 0 \quad \text{.....50}$$

$$\frac{\partial L}{\partial \lambda} = x^2 + y^2 - 1 = 0 \quad \text{.....51}$$

Thus we get from above

$$2\lambda = \frac{5}{x^2y^2} = \frac{10}{xy^4}$$

$$\frac{1}{x} = \frac{2}{y^2}, y^2 = 2x$$

Then equation (50) is now

$$x^2 + 2x - 1 = 0$$

DOI: 10.5281/zenodo.14484115

DOI: 10.5281/zenodo.14484115

Therefore, $x^* = \frac{1}{\sqrt{3}}$ and $y^* = \frac{\sqrt{2}}{\sqrt{3}}$

$$\lambda = \frac{5\sqrt{3} \cdot 3^2}{2^2} = \frac{45\sqrt{3}}{4}$$

The minimum point of $f(x, y)$ is $\left(\frac{1}{\sqrt{3}}, \frac{\sqrt{2}}{\sqrt{3}}\right)$

Problem 3

Minimize $Z = x_1^2 + x_2^2 + x_3^2 + 4(x_1 - 4) + 4(x_1 + x_2 - 14)$
Subjectto: $x_1 + x_2 + x_3 = 18$
$x_1, x_2,$ $x_3 \geq 0$

The Lagrange equation is

$$L(x_1, x_2, x_3, \lambda) = x_1^2 + x_2^2 + x_3^2 + 4(x_1 - 4) + 4(x_1 + x_2 - 14) + \lambda(x_1 + x_2 + x_3 - 18)$$

The necessary conditions are

$$\frac{\partial L}{\partial x_1} = 2x_1 + 8 + \lambda = 0 \quad \ldots\ldots 52$$

$$\frac{\partial L}{\partial x_2} = 2x_2 + 4 + \lambda = 0 \quad \ldots\ldots 53$$

$$\frac{\partial L}{\partial x_3} = 2x_3 + \lambda = 0 \quad \ldots\ldots 54$$

$$\frac{\partial L}{\partial \lambda} = x_1 + x_2 + 2x_3 - 18 = 0 \quad \ldots\ldots 55$$

$$\lambda \leq 0$$

From the equations (52) to (53), $x_1 = 4, x_2 = 6, x_3 = 8, \lambda = -16$

Thus the maximum point of Z is (4,6,8)

DOI: 10.5281/zenodo.14484115

DOI: 10.5281/zenodo.14484115

Problem 4

Minimize $f(X) = \frac{1}{2}x_1^2 + \frac{1}{2}x_2^2 + \frac{1}{2}x_3^2$

Subject to: $x_1 + x_2 \leq 3$

$3x_1 - x_2 \leq 6$

$x_1 + x_2 + x_3 \leq 6$

$x_1, x_2,$ $x_3 \geq 0$

The Lagrange function is

$$L(X, \lambda, s) = \frac{1}{2}x_1^2 + \frac{1}{2}x_2^2 + \frac{1}{2}x_3^2 + \lambda_1(x_1 + x_2 - 3 + s_1^2) + \lambda_2(3x_1 - x_2 - 6 + s_2^2) + \lambda_3(x_1 + x_2 + x_3 - 6 + s_3^2)$$

where s_1,s_2 and s_3 are slack variables necessary conditions are

$$\frac{\partial L}{\partial x_1} = x_1 + \lambda_1 + 3\lambda_2 + \lambda_3 = 0 \quad \ldots 56$$

$$\frac{\partial L}{\partial x_2} = x_2 + \lambda_1 - \lambda_2 + \lambda_3 = 0 \quad \ldots 57$$

$$\frac{\partial L}{\partial x_3} = x_3 + \lambda_3 = 0 \quad \ldots 58$$

$$\frac{\partial L}{\partial \lambda_1} = x_1 + x_3 - 3 + s_1^2 = 0 \quad \ldots 59$$

$$\frac{\partial L}{\partial \lambda_2} = 3x_1 - x_2 - 6 + s_2^2 = 0 \quad \ldots 60$$

$$\frac{\partial L}{\partial \lambda_3} = x_1 - x_2 + x_3 - 6 + s_3^2 = 0 \quad \ldots 61$$

$$\frac{\partial L}{\partial s_1} = 2\lambda_1 s_1 = 0 \quad \ldots 62$$

$$\frac{\partial L}{\partial s_2} = 2\lambda_2 s_2 = 0 \quad \ldots 63$$

$$\frac{\partial L}{\partial s_3} = 2\lambda_3 s_3 = 0 \quad \ldots 64$$

This system of equations is solved analytically which resulted in global maximum $f(X)$

DOI: 10.5281/zenodo.14484115

DOI: 10.5281/zenodo.14484115

exists at $\lambda_1 = -\frac{3}{2}, \lambda_2 = 0, \lambda_3 = 0$ which gives $x_1 = \frac{3}{2}, x_2 = \frac{3}{2}$ and $x_3 = 0$

NUMERICAL SOLUTIONS TO THE PROBLEMS

Table 1: Numerical solutions to the prolems in section 4.1

Problem	Analytical	Newton	Quasi-Newton	CN	CQ
1	$x_1 = 2.3333$	$x_1 = 2.3333$	$x_1 = 2.3333$	10	10
	$x_2 = 0.1111$	$x_2 = 0.1111$	$x_2 = 0.1111$		
	$x_3 = -1.2222$	$x_3 = -1.2222$	$x_3 = -1.2222$		
2	$x = 0.5774$	$x = 0.5774$	$x = 0.5774$	70	70
	$y = 0.8165$	$y = 0.8165$	$y = 0.8165$		
3	$x_1 = 4$	$x_1 = 4$	$x_1 = 4$	2	2
	$x_2 = 6$	$x_2 = 6$	$x_2 = 6$		
	$x_3 = 8$	$x_3 = 8$	$x_3 = 8$		
4	$x_1 = 1.5$	$x_1 = 1.5$	$x_1 = 1.5$	7	7
	$x_2 = 1.5$	$x_2 = 1.5$	$x_2 = 1.5$		
	$x_3 = 0.0$	$x_3 = 0.0$	$x_3 = 0.0$		

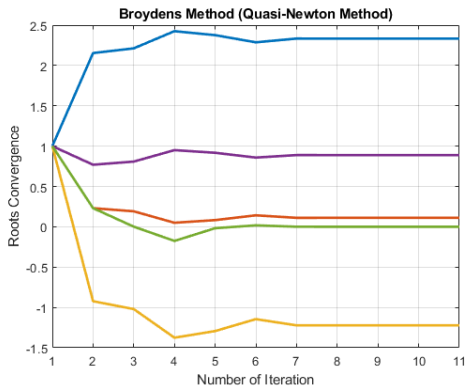

Fig. 2: Graph of problem 1.

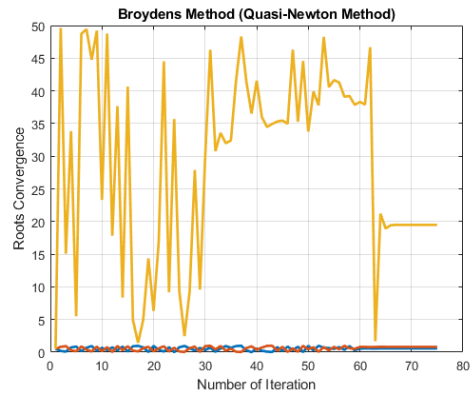

Fig. 3: Graph of problem 2

DOI: 10.5281/zenodo.14484115

DOI: 10.5281/zenodo.14484115

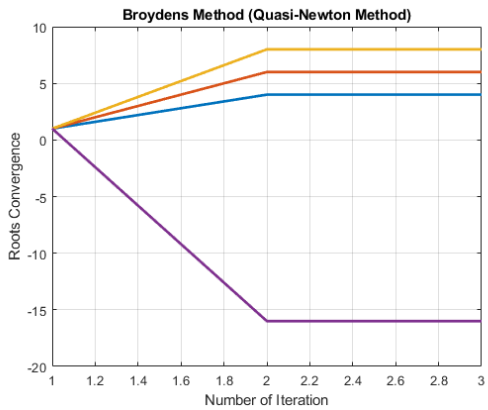

Fig. 4: Graph of problem 3.

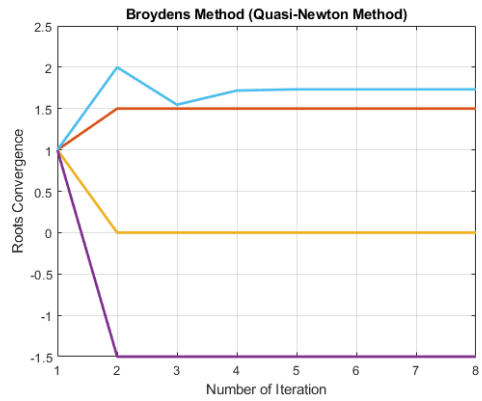

Fig. 5: Graph of problem 4

DISCUSSION

The numerical results obtained from applying the Quasi-Newton Approximation Scheme (QNAS) were compared against existing methods, particularly those utilizing Lagrangian multipliers. The QNAS demonstrated superior accuracy and efficiency, particularly in scenarios where traditional methods struggled to converge or provided less precise solutions. This is significant because many engineering problems involve complex constraints that can lead to difficulties in finding optimal solutions.

One of the standout features of the QNAS is its convergence rate. The results indicated that the method converges more rapidly to the optimal solution compared to other numerical methods. This is particularly important in practical applications where computational time is critical (Wierna, Yago, Lloberas-Valls, Huespe, & Oliver. 2024). The ability to achieve a solution in fewer iterations not only saves time but also reduces computational resource usage, making it a more sustainable option for large-scale problems.

The study highlighted the challenges associated with solving KKT conditions analytically, especially in non-convex problems. The QNAS effectively addresses these challenges by providing a numerical approach that can yield approximate solutions even when traditional algebraic methods fail. This capability is crucial for real-world applications where exact solutions are often unattainable due to the complexity of the constraints involved.

The results showed that the QNAS is versatile and can be applied to a range of NLPPs, including those with convex objective functions and both convex and concave constraints. This flexibility makes it a valuable tool for engineers and researchers

DOI: 10.5281/zenodo.14484115

DOI: 10.5281/zenodo.14484115

dealing with diverse optimization problems. The ability to adapt to different types of constraints enhances its utility in practical scenarios.

The findings suggest several avenues for future research. For instance, the study proposes further exploration of NLPPs with concave objective functions, which could expand the applicability of the QNAS. Additionally, investigating the nature of optimal points when using the KKT method could provide deeper insights into the behavior of solutions in non-linear programming.

The implications of these results extend to various fields of engineering and applied sciences. The ability to efficiently solve NLPPs with inequality constraints can lead to improved designs, optimized resource allocation, and enhanced decision-making processes in engineering projects. The study emphasizes the importance of developing robust numerical methods that can handle the complexities of real-world problems (Björck, 2024).

The results of the study on the Quasi-Newton approximation method for NLPPs with inequality constraints demonstrate its effectiveness in providing accurate and efficient solutions. The method's ability to handle KKT conditions numerically, coupled with its rapid convergence and versatility, positions it as a significant advancement in optimization techniques. These findings not only contribute to the theoretical understanding of non-linear programming but also have practical implications for engineering and applied optimization problems. Further research in this area could lead to even more refined methods and broader applications.

CONCLUSION

In this paper, we have discussed a Quasi-Newton approximation of Nonlinear programming problem with inequality constraints. The numerical results obtained showed clearly that the method is accurate and more efficient than the other existing methods when solving NLPP using Lagrangian multipliers. The method also help in finding approximate solution when NLPP leads to KKT conditions which cannot be solved analytically. we have succeeded in constructing a Quasi-Newton method for finding KKT point of NLPP with convex objective function and convex constraints (minimization) and convex objective function and concave constraint (maximization).

REFERENCES

Andreani, R., Schuverdt, M. L., & Secchin, L. D. (2024). On enhanced KKT optimality conditions for smooth nonlinear optimization. SIAM Journal on Optimization, 34(2), 1515-1539.

Bashier, E. B. (2020). Practical Numerical and Scientific Computing with MATLAB® and Python. CRC Press.

DOI: 10.5281/zenodo.14484115

DOI: 10.5281/zenodo.14484115

Björck, Å. (2024). *Numerical methods for least squares problems.* Society for Industrial and Applied Mathematics.

Chen, Y., & Zhang, Y. (2023). A hybrid Quasi-Newton method for large-scale nonlinear programming problems. Journal of Optimization Theory and Applications, 188(3), 789-805.

Biegler, L. T., & Grossmann, I. E. (2004). Retrospective on optimization. Computers & Chemical Engineering, 28(8), 1169-1192.

Broyden, C. G. (1969). A new method of solving nonlinear simultaneous equations. The omputer Journal, 12(1), 94-99.

Floudas, C. A., & Pardalos, P. M. (Eds.). (2008). Encyclopedia of optimization. Springer Science & Business Media

Liu, H., & Wang, J. (2022). An adaptive Quasi-Newton method for solving nonlinear programming problems with inequality constraints. Computers & Operations Research, 139, 105-120.

Liuzzi, G., Lucidi, S., Rinaldi, F., & Vicente, L. N. (2019). Trust-region methods for the derivative free optimization of nonsmooth black-box functions. SIAM Journal on Optimization, 29(4), 3012-3035.

Martınez, J. M. (2000). Practical quasi-Newton methods for solving nonlinear systems. Journal of computational and Applied Mathematics, 124(1-2), 97-121.

Martins, J. R., & Ning, A. (2021). Engineering design optimization. Cambridge University Press.

Wierna, P., Yago, D., Lloberas-Valls, O., Huespe, A., & Oliver, J. (2024). On the Efficient and AccurateNon-linear Computational Modeling of Multilayered Bending Plates. State of the Art and a Novel Proposal: The 2 D+ Multiscale Approach. *Archives of Computational Methods in Engineering*, 1-56.

Xu, Y., & Chen, S. (2023). Convergence analysis of a Quasi-Newton method for constrained optimization problems. Mathematical Programming, 189(1), 1-25.

Zhang, L., & Li, X. (2023). A new approach to solving nonlinear programming problems using modified Quasi-Newton methods. Applied Mathematics and Computation, 426, 127-145.

DOI: 10.5281/zenodo.14484115