Southeast-Asian J. of Sciences, Vol. 8, No. 1 (2020) pp. 28-43

EFFICIENT INTERIOR-POINT ALGORITHM FOR SOLVING THE GENERAL NON-LINEAR PROGRAMMING PROBLEMS

Enas Omer[*], **S. Y. Abdelkader**[†]
and
Mahmoud El-Alem[‡]

Department of Mathematics, Faculty of Science,
Alexandria University, Alexandria, Egypt.

[*] *e-mail: enas.o.s@gmail.com*
[†] *email: shyashraf@yahoo.com*
[‡] *e-mail: mmelalem@yahoo.com; mmelalem@hotmail.com*

Abstract

An Interior-point algorithm with a line-search globalization is proposed for solving the general nonlinear programming problem. At each iteration, the search direction is obtained as a resultant of two orthogonal vectors. They are obtained by solving two square linear systems. An upper-triangular linear system is solved to obtain the Lagrange multiplier vector. The three systems that must be solved each iteration are reduced systems obtained using the projected Hessian technique. This fits well for large-scale problems. A modified Hessian technique is embedded to provide a sufficient descent for the search direction. Then the length of the direction is decided by backtracking line search with the use of a merit function to generate an acceptable next point.

The performance of the proposed algorithm is validated on some well-known test problems and with three well-known engineering design problems. In addition, the numerical results are compared to other efficient

Key words: Newton's method, Goodman's method, projected Hessian, interior-point, line-search, nonlinear programming, Numerical comparisons.
2010 AMS Classification: 49M37, 65K05, 90C35, 90C47.

methods. The results show that the proposed algorithm is effective and promising.

1 Introduction

The general nonlinear programming problem (NLP) is the most general class of optimization problems where it aims to minimize a nonlinear objective function subject to a set of nonlinear equality and inequality constraints. This problem exists in applied mathematics, engineering, management and many other applications. The importance of such problems encouraged considerable research in this area to develop algorithms to solve such problems. One of the most effective methods for solving these problems is the Newton interior-point method due to its fast local convergence [12].

The start was in 1984 when Karmarkar [19] announced a fast polynomial-time interior-point method for linear programming. Since that time, interior-point methods have rapidly and noticeably advanced which impact on the evolution of the theory and practice of constrained optimization. Many remarkable primal-dual interior-point methods have proven merit for solving Problem (NLP) [4, 13].

Das [9] and Dennis et al [10] generalized the use of the scaling matrix introduced by Coleman and Li [8] for solving the unconstrained optimization to Problem (NLP). El-Alem et al [12] proved the local and q-quadratic convergence of the method. More recently, based on the interior point approach and Coleman-Li scaling matrix, Abdelkader et al [1] suggested an interior-point trust-region algorithm. The method decomposes the sequential quadratic programming (SQP) subproblem into two trust region subproblems to compute the normal and the tangential components of the trial step. The method was proved to be globally convergent [2].

Other primal-dual interior-point algorithm was proposed by Jian et al [18]. This algorithm is a QP-free in which the QPs are replaced by systems of linear equations with the same coefficient matrix that formed by using a 'working set' technique to determine the active set.

Different algorithms were suggested based on the (SQP) method. In order to obtain the search direction, Jian and et al [16, 17] with different techniques solved a QP subproblem and a system of linear equation to obtain a master and an auxiliary directions respectively. The auxiliary direction in [16] was needed to improve the master direction to guarantee superlinearly convergence for the method. On the other hand, Jian et al [17] needed an auxiliary direction to overcome the Maratos effect [21]. The search direction is then a combination of the two directions.

This paper is based on the works [1, 8, 9, 10, 12] and the concept suggested by Goodman [14] which shows that the extended system of Newton method for equality constrained optimization (EQ) can be reduced into two systems of lower dimensions. We extend Goodman's concept to Problem (NLP) to overcome the disadvantage of solving the extended system once at each iteration especially for large-scale problems .

This paper is organized as follows. In Section 2, we set some preliminaries and notations. The suggested algorithm is proposed in Section 3. The implementations of the proposed algorithm on some well-known test problems are reported in Section 4. Section 5 contains concluding remarks.

2 Preliminaries

We consider the general nonlinear programming problem of the form:

$$\begin{array}{ll} \text{Minimize} & f(x) \\ \text{Subject to} & h(x)=0 \\ & a \leqslant x \leqslant b, \end{array} \tag{2.1}$$

where $f:\Re^n \to \Re$, $h:\Re^n \to \Re^m$, $a \in (\Re \cup (-\infty))^n$, $b \in (\Re \cup (+\infty))^n$, and $m<n$. The functions f and $h_i, i=1,2,...,m$ are assumed to be at least twice continuously differentiable. The Lagrangian function associated with Problem (2.1) is :

$$L(x,\lambda,\alpha,\beta) = l(x,\lambda) - \alpha^T(x-a) - \beta^T(b-x),$$

where $l(x,\lambda) = f(x) + \lambda^T h(x)$, $\lambda \in \Re^m$ is the Lagrange multiplier vector associated with the equality constraints and $\alpha, \beta \in \Re^n$ are multipliers associated with the bounds.

The KKT conditions for a point $x_* \in \Re^n$ to be a solution for Problem (2.1) are the existence of multipliers $\lambda_* \in \Re^m, \alpha_*, \beta_* \in \Re^n_+$ such that $(x_*, \lambda_*, \alpha_*, \beta_*)$ satisfies:

$$\begin{array}{lcl} \nabla_x l(x,\lambda) - \alpha + \beta & = & 0 \\ h(x) & = & 0 \\ a \leqslant x \leqslant b & & \\ \alpha(x-a) & = & 0 \\ \beta(b-x) & = & 0. \end{array} \tag{2.2}$$

Consider the Coleman-Li diagonal scaling matrix $D_\lambda(x)$ (simplified by $D(x)$) whose diagonal elements are defined as:

$$d^{(i)}(x) = \begin{cases} \sqrt{x^{(i)} - a^{(i)}}, & \text{if } (\nabla_x l(x,\lambda))^{(i)} \geq 0 \text{ and } a^{(i)} > -\infty, \\ \sqrt{b^{(i)} - x^{(i)}}, & \text{if } (\nabla_x l(x,\lambda))^{(i)} < 0 \text{ and } b^{(i)} < \infty, \\ 1, & \text{otherwise.} \end{cases}$$

The scaling matrix $D(x)$ transforms the KKT conditions (2.2) to the conditions that (x_*, λ_*) satisfies the following $(n+m)\times(n+m)$ nonlinear system of equations:

$$\begin{array}{lcl} D^2(x)\nabla_x l(x,\lambda) & = & 0 \\ h(x) & = & 0, \end{array} \tag{2.3}$$

with the restriction that $a \leqslant x_* \leqslant b$.

2.1 Extended System of Problem (2.1)

Let $a \leqslant x \leqslant b$. Newton's method on the nonlinear system (2.3) gives:

$$\begin{array}{rl} [D^2(x)\nabla_x^2 l(x,\lambda)+ & diag(\nabla_x l(x,\lambda))diag(\eta(x))]\Delta x + D^2(x)\nabla h(x)\Delta\lambda = \\ & = -D^2(x)\nabla_x l(x,\lambda) \\ \nabla h(x)^T\Delta x & = -h(x), \end{array}$$

where η is the vector defined as $\eta^{(i)}(x) = \frac{\partial((d^i(x))^2)}{\partial x^{(i)}}, i = 1, 2, ..., n$. Or equivalently:

$$\eta^{(i)}(x) = \begin{cases} 1, & \text{if } (\nabla_x l(x,\lambda))^{(i)} \geq 0 \text{ and } a^{(i)} > -\infty, \\ -1, & \text{if } (\nabla_x l(x,\lambda))^{(i)} < 0 \text{ and } b^{(i)} < \infty, \\ 0, & \text{otherwise.} \end{cases}$$

This gives the following linear system:

$$\begin{bmatrix} B & D^2(x)\nabla h(x) \\ \nabla h(x)^T & 0 \end{bmatrix} \begin{bmatrix} \Delta x \\ \Delta\lambda \end{bmatrix} = -\begin{bmatrix} D^2(x)\nabla_x l(x,\lambda) \\ h(x) \end{bmatrix}, \tag{2.4}$$

where $B = D^2(x)\nabla_x^2 l(x,\lambda) + diag(\nabla_x l(x,\lambda))diag(\eta(x))$. The restriction $a < x < b$ implies that the scaling matrix $D(x)$ is necessarily nonsingular. Multiplying the first block of System (2.4) by $D^{-1}(x)$ and scaling the step by $\Delta x = D(x)s$, arise the following extended system:

$$\begin{bmatrix} H & D(x)\nabla h(x) \\ (D(x)\nabla h(x))^T & 0 \end{bmatrix} \begin{bmatrix} s \\ \Delta\lambda \end{bmatrix} = -\begin{bmatrix} D(x)\nabla_x l(x,\lambda) \\ h(x) \end{bmatrix}, \tag{2.5}$$

where $H = D(x)\nabla_x^2 l(x,\lambda)D(x) + diag(\nabla_x l(x,\lambda))diag(\eta(x))$. After solving (2.5) for s, we set $\Delta x = D(x)s$. But there is no guarantee that the next iterate point will satisfy:

$$a < x + \Delta x < b. \tag{2.6}$$

A damping parameter is needed to force (2.6). Das [9] uses the following damping parameter at each iteration k:

$$\tau_k = \min\Big\{1, min_i\Big\{c_k^{(i)}, d_k^{(i)}\Big\}\Big\}, \tag{2.7}$$

where

$$c_k^{(i)} = \begin{cases} \frac{a^{(i)} - x_k^{(i)}}{\Delta x_k^{(i)}} & \text{if } a^{(i)} > -\infty \ \&\ \Delta x_k^{(i)} < 0 \\ 1 & \text{otherwise,} \end{cases}$$

and

$$d_k^{(i)} = \begin{cases} \frac{b^{(i)} - x_k^{(i)}}{\Delta x_k^{(i)}} & \text{if } b^{(i)} < \infty \ \& \ \Delta x_k^{(i)} > 0 \\ 1 & \text{otherwise.} \end{cases} .$$

We multiply τ_k by 0.99 to insure that (2.6) will hold.

2.2 Overall Algorithm

We outline the interior-point Newton algorithm for solving Problem (2.1):

Algorithm 1.

```
    Given $x_0 \in \Re^n$, such that $a < x_0 < b$ and $\lambda_0 \in \Re^m$. For $k = 0, 1, ...,$
  until
    convergence, do the following steps:
  Step 1.
    Compute Newton's step $s_k$ and $\Delta\lambda_k$ by solving System (2.5).
    Set $\Delta x_k = D(x_k) s_k$.
  Step 2.
    Compute the damping parameter $\tau_k$ using (2.7).
  Step 3.
    Set $x_{k+1} = x_k + 0.99\tau_k \Delta x_k$ and $\lambda_{k+1} = \lambda_k + \Delta\lambda_k$.
```

This algorithm has a local q-quadratic rate of convergence [12] which is the main advantage of it. But the disadvantage of using extended system (2.5) to obtain Newton's step is that the dimension of the system is directly proportional with that of the problem. In the interior-point approach, we add non-negative slack variables to the inequality constraints to convert them to equalities. This technique will cause an increase to the number of both variables and equality constraints. Consequently, the dimension of the problem will increase. This disadvantage was the motivation of our work. In this paper, we extend Goodman's method [14] for problem (EQ) to problem (NLP) to overcome this difficulty.

Finally to simplify the notations, we set D_k to denote $D(x_k)$, l_k to denote $l(x_k, \lambda_k)$, ..., and so on. We assume that $(D_k \nabla h_k)$ has a full column rank.

3 Proposed Algorithm

Consider the QR factorization of $D_k \nabla h_k$ as follows:

$$D_k \nabla h_k = \begin{bmatrix} Y_k & Z_k \end{bmatrix} \begin{bmatrix} R_k \\ 0 \end{bmatrix}, \qquad (3.8)$$

where Y_k is an $n \times m$ matrix whose columns form an orthonormal basis for the column space of $(D_k \nabla h_k)$, Z_k is an $n \times (n - m)$ matrix with orthonormal columns spanning the null space of $(D_k \nabla h_k)^T$. i.e., $Z_k^T(D_k \nabla h_k) = 0$ and R_k is an $m \times m$ nonsingular upper triangular matrix.

The null space matrix Z_k obtained is not guaranteed to be smooth in the region of interest. There are many techniques to enforce this when necessary (see Nocedal and Overton [24] for more detail).

Multiply the first block of the extended system (2.5) by Z_k^T, gives:

$$\begin{bmatrix} Z_k^T H_k \\ (D_k \nabla h_k)^T \end{bmatrix} s_k = - \begin{bmatrix} Z_k^T D_k \nabla_x l_k \\ h_k \end{bmatrix}. \tag{3.9}$$

We decompose the step s_k as follows:

$$s_k = Y_k u_k + Z_k v_k, \tag{3.10}$$

where $Y_k u_k$ is the normal component and $Z_k v_k$ is the tangential one. If we use this decomposition of the step in system (3.9), the second block gives:

$$(D_k \nabla h_k)^T Y_k u_k = -h_k, \tag{3.11}$$

and the first block gives:

$$(Z_k^T H_k Z_k) v_k = -Z_k^T (D_k \nabla_x l_k + H_k Y_k u_k). \tag{3.12}$$

There is no guarantee that the matrix $(Z_k^T H_k Z_k)$ in system (3.12) be positive definite. Nocedal and Wright [25] disscussed strategies for modifying the Hessian matrices and set some restrictions to these strategies to guarantee sufficient positive definiteness. One of these strategies is called eignvalue modification. This strategy replaces $(Z_k^T H_k Z_k)$ by a positive definite approximation matrix B_k, in which all negative eigenvalues of $(Z_k^T H_k Z_k)$ are shifted by a small positive number but in some what larger than the machine accuracy ϵ. We set $\rho = \sqrt{\epsilon}$ and $\mu = \max(0, \rho - \delta_{min})$, where δ_{min} denotes the smallest eignvalue of $(Z_k^T H_k Z_k)$. Then, the modified matrix is of the form $B_k = (Z_k^T H_k Z_k) + \mu I$. This modification generates positive definite approximation matrix B_k. This is summarized in the following scheme:

Scheme 3.1. *(Modifying $(Z_k^T H_k Z_k)$)*
Set $B_k = Z_k^T H_k Z_k$, $\rho = 10^{-8}$
Evaluate the smallest eignvalue δ_{min} of B_k **If** *$\delta_{min} < \rho$ then, $B_k = B_k + (\rho - \delta_{min}) I$*

The step is computed from (3.10) and is guaranteed to be descent as $D_k \nabla h_k$ has a full column rank and B_k is positive definite. The unscaled step $\Delta x_k = D_k s_k$ is computed. After that, we search among the search direction Δx_k the appropriate step size using the backtracking line-search algorithm [25]. During the backtracking procedure, we seek a step size $\gamma_k \in (0, 1]$ that provides sufficient reduction in the merit function $P(x_k, r_k) = f_k + \frac{r_k}{2}\|h_k\|^2$, where $r > 0$ is a penalty parameter:

$$P(x_k + \gamma_k \Delta x_k) \leq P_k + \alpha\gamma_k \nabla P_k^T \Delta x_k, \tag{3.13}$$

where $\alpha \in (0, \frac{1}{2}]$. The backtracking algorithm used is as follows:

Scheme 3.2. *(Backtracking line search)*
Given $\alpha \in (0, \frac{1}{2}]$. *Set* $\gamma_k = 1$ **While** $P(x_k + \gamma_k \Delta x_k) > P_k + \alpha\gamma_k \nabla P_k^T \Delta x_k$
Set $\gamma_k = \frac{\gamma_k}{2}$

At iteration k, to compute the Lagrange multiplier λ_k, Goodman [14], in solving Problem (EQ), formed another QR factorization for ∇h_{k+1} after computing the iterate point x_{k+1} to get Y_{k+1} and used it to solve for λ_{k+1} the following system:

$$\nabla h_{k+1}\lambda_{k+1} = -\nabla f_{k+1}.$$

It gives rise to the following system to obtain λ_{k+1}:

$$R_{k+1}\lambda_{k+1} = -Y_{k+1}^T \nabla f_{k+1}.$$

In our algorithm, we solve the first block of the extended system (2.5) for the Lagrange multiplier step $\Delta\lambda_k$:

$$(D_k \nabla h_k)\Delta\lambda_k = -(D_k \nabla_x l_k + H_k s_k).$$

Note, we use the same QR factorization (3.8) of $D_k \nabla h_k$. Multiply both sides by Y_k^T, gives:

$$R_k \Delta\lambda_k = -Y_k^T (D_k \nabla_x l_k + H_k s_k). \tag{3.14}$$

This is an upper-triangular system of equations that needs a back substitution to obtain $\Delta\lambda_k$. Then, we set:

$$\lambda_{k+1} = \lambda_k + \Delta\lambda_k.$$

We will call our proposed algorithm (EIPA). It stands for "Efficient Interior-Point Algorithm" for solving Problem (NLP). The detailed description of (EIPA) is stated:

Algorithm 2. *(EIPA)*
Given $x_0 \in \Re^n$, *such that* $a < x_0 < b$. *Evaluate* $\lambda_0 \in \Re^m$. *Set* $\rho = 10^{-8}$, $r = 1$, $\alpha = 10^{-4}$ *and* $\varepsilon > 0$. *While* $\|D_k \nabla_x l_k\|_2 + \|h_k\|_2 > \varepsilon$, *do the following:*
Step 1. *(QR factorization for* $(D_k \nabla h_k)$*)*
(a) *Compute the scaling matrix* D_k.
(b) *Obtain the QR factorization for* $(D_k \nabla h_k)$.

```
Step 2. (Compute the step $\Delta x_k$)
        (a) Modify the projected Hessian $(Z_k^T H_k Z_k)$ using scheme (3.1).
        (b) Compute the orthogonal components $u_k$ and $v_k$ using (3.11) and (3.12).
        (c) Set $s_k = Y_k u_k + Z_k v_k$, $\Delta x_k = D_k s_k$.
Step 3. (Backtracking line search)
        Evaluate the step length $\gamma_k$ using scheme (3.2).
Step 4. (Interiorization)
        (a) Compute the damping parameter $\tau_k$ using (2.7).
        (b) Set $x_{k+1} = x_k + 0.99\tau_k\gamma_k\Delta x_k$.
Step 4. (Update Lagrange multiplier $\lambda_{k+1}$)
        (a) Compute Lagrange step $\Delta\lambda_k$ by solving (3.14).
        (b) Set $\lambda_{k+1} = \lambda_k + \Delta\lambda_k$.
Step 5. (Update $D_k$, $H_k$ and $r$)
    Update both scaling matrix $D_k$ and $H_k$. Set $r = 10 \times r$.
    End while
```

4 Numerical Results

In this section, we report the results of our numerical implementations of EIPA for solving Problem (NLP). The results show that EIPA is effective and promising. The code was written in MATLAB R2009b on Windows 10 with a machine epsilon 10^{-16}. Different numerical implementations were performed to show the computational efficiency of EIPA and its competitiveness relative to other existing efficient algorithms. During the numerical implementations, the constants are set as follows: $\rho = 10^{-8}, \alpha = 10^{-4}$. The penalty parameter $r = 1$ at the first iteration and is updated using $r_{k+1} = 10 \times r_k$. EIPA is terminated successfully if the termination criterion is satisfied. On the other hand, if 500 iterations were completed without satisfying the termination condition, it is called a failure. The following table describes the abbreviations that are used during our implementations:

4.1 Comparison with Established Algorithms

We set some comparisons between EIPA and other algorithms using test problems from Hock-Schittkowski Collection [15]. The initial points and the terminating tolerance are chosen to be the same as those in the compared algorithms. In Table 4.2 results of EIPA using test problems from [15] are listed with those from IPTRA [1]. EIPA demonstrated competitiveness with IPTRA [1]. The number of NF in EIPA is larger than that in IPTRA in some problems because of the NF counted inside back-tracking trials. Table 4.3shows comparisons between EIPA and the algorithm in [18]. We refer to this method as ALGO1.

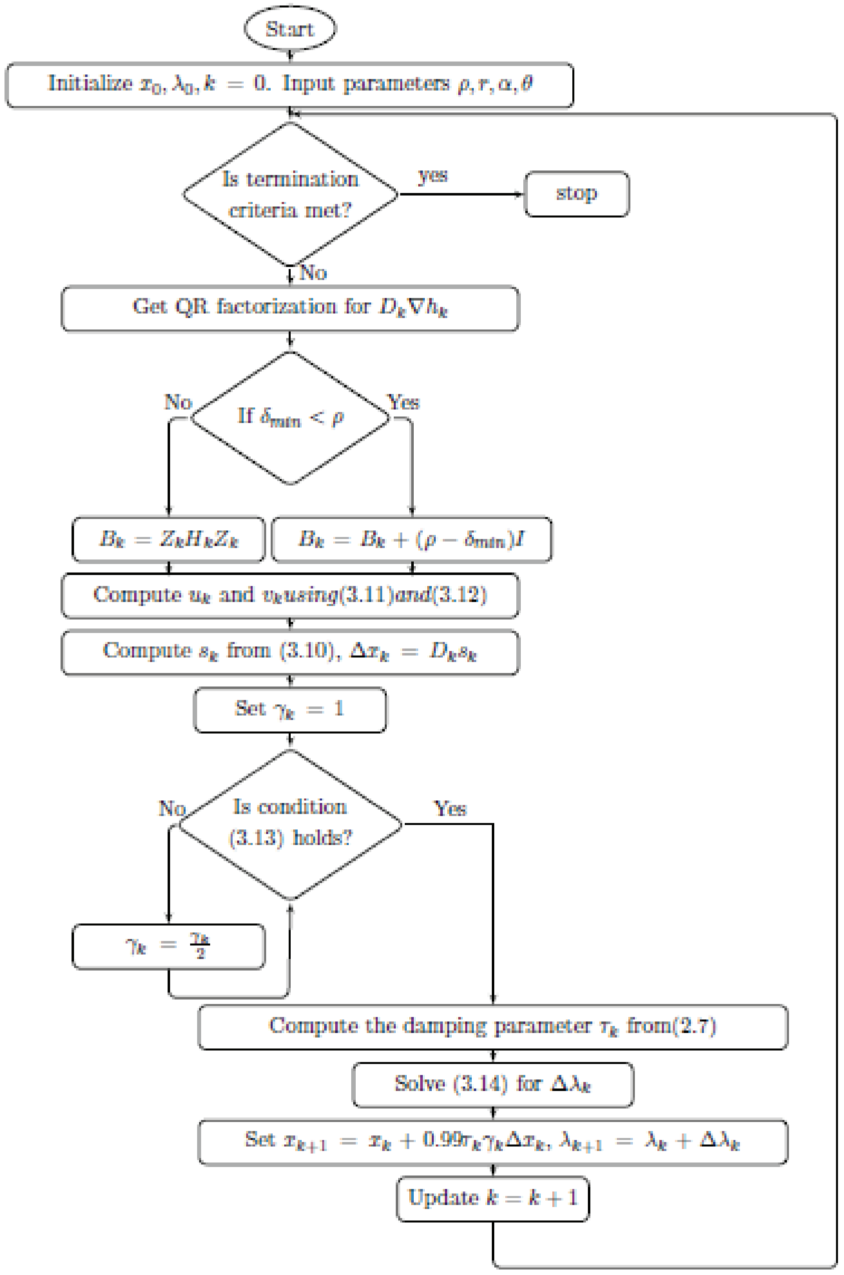
Start
Initialize $x_0, \lambda_0, k = 0$. Input parameters ρ, r, α, θ
Is termination criteria met?
yes
stop
No
Get QR factorization for $D_k \nabla h_k$
No
If $\delta_{min} < \rho$
Yes
$B_k = Z_k H_k Z_k$
$B_k = B_k + (\rho - \delta_{min})I$
Compute u_k and $v_k using (3.11) and (3.12)$
Compute s_k from (3.10), $\Delta x_k = D_k s_k$
Set $\gamma_k = 1$
No
Is condition (3.13) holds?
Yes
$\gamma_k = \frac{\gamma_k}{2}$
Compute the damping parameter τ_k from(2.7)
Solve (3.14) for $\Delta\lambda_k$
Set $x_{k+1} = x_k + 0.99\tau_k\gamma_k\Delta x_k$, $\lambda_{k+1} = \lambda_k + \Delta\lambda_k$
Update $k = k + 1$

Table 4.1 The abbreviations used in numerical results

Abbreviation	Description
HS	The name of the problem as in Hock-Schittkowski-Collection [15]
n	Number of variables of the problem
me	Number of equality constraints
mi	Number of inequality constraints
NI	Number of iterations
NF	Number of function evaluations
FV	The final value of the objective function
AC	The value of $\|D_k \nabla_x l_k\| + \|h_k\|$ at the solution
CPU	The CPU time in seconds
–	Data is not available

The results show that EIPA is obviously better than ALGO1 in almost all reported test problems. In Table 4.4,the performance of EIPA is compared against other algorithms based on the ideas of sequential quadratic programming, which are SNQP [17], and ALGO2 [16]. The numerical results show that EIPA succeeded to obtain the lower NI, NF and the CPU time relative to SNQP [17], and ALGO3 [16] in almost all reported test problems.

4.2 Classical Engineering Design Problems

To validate the proposed algorithm EIPA, we use three well-known engineering design problems which are tension/compression spring design problem [5], welded beam design problem [7] and multistage heat exchanger design problem [6]. The outputs of design variables and the optimal solution of those problems produced when applying EIPA are compared with those obtained by both mathematical and heuristic approaches.

4.2.1 Tension/Compression Spring Design Problem

This problem aims to minimize the weight f of the spring (as shown in Fig. 4.1) subject to constraints on minimum deflection, shear stress, surge frequency, limits on outside diameter and on design variables. The problem consists of three decision variables which are mean coil diameter D, wire diameter d and number of active coils N. The mathematical formulation is found in Arora [5].

Table 4.5 shows the comparison of the results of the problem obtained from EIPA and from other approaches as Gravitational Search Algorithm GSA [23], Grey Wolf Optimizer GWO [22], Chaotic Grey Wolf Optimizer CGWO [20], Interior-Point Trust-Region Algorithm IPTRA [1] and Constrained Guided Particle Swarm Optimization CGPSO [3]. From the results, it can be seen that EIPA outperforms the best solution of the indicated algorithms.

Table 4.2 Numerical comparisons between EIPA and IPTRA

Prob.	n/me/mi	IP	Code	NI	NF	AC	CPU
HS17	2/0/5	$(0,1)^T$	EIPA	51	207	$9.8831e-006$	0.069
			IPTRA	7	8	$9.8648e-009$	0.045
HS21	2/0/5	$(5,2)^T$	EIPA	6	7	$8.2924e-012$	0.007
			IPTRA	4	5	$4.0984e-014$	0.027
HS24	2/0/5	$(1,0.5)^T$	EIPA	17	22	$4.6769e-009$	0.044
			IPTRA	5	6	–	0.098
HS30	3/0/7	$(2,1,1)^T$	EIPA	4	5	$2.7925e-009$	0.009
			IPTRA	5	6	$5.4610e-008$	0.030
HS37	3/0/8	$(10,10,10)^T$	EIPA	5	6	$5.4319e-009$	0.007
			IPTRA	7	8	$8.6735e-007$	0.068
HS41	4/1/8	$(0.5,0.5,0.5,1)^T$	EIPA	2	3	$9.6000e-019$	0.004
			IPTRA	5	6	$1.4168e-008$	0.032
HS53	5/3/10	$(-6,2,2,2,2)^T$	EIPA	3	4	$1.8848e-010$	0.005
			IPTRA	4	5	$8.0960e-008$	0.028
HS60	3/1/6	$(2,2,2)^T$	EIPA	6	7	$4.7199e-012$	0.007
			IPTRA	7	8	–	0.046
HS65	3/0/7	$(1,1,0)^T$	EIPA	46	198	$3.3715e-009$	0.092
			IPTRA	9	10	$1.0674e-010$	0.050
HS71	4/1/9	$(2,4,4,2)^T$	EIPA	6	7	$9.6054e-009$	0.008
			IPTRA	6	7	$1.2910e-010$	0.041
HS74	4/3/10	$(1,1,0,0)^T$	EIPA	8	68	$4.6908e-017$	0.049
			IPTRA	15	16	$4.8247e-007$	0.105
HS75	4/3/10	$(1,1,0,0)^T$	EIPA	8	9	$3.1548e-013$	0.011
			IPTRA	16	17	$1.4066e-009$	0.094

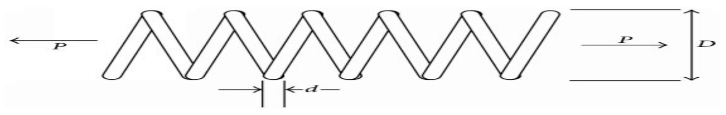

Figure 4.1: Schematic diagram of tension/spring design

4.2.2 Welded Beam Design Problem

The objective of welded beam design problem (as shown in Fig. 4.2) is to minimize the cost subject to constraints on shear stress, bending stress, buckling load on the bar, end deflection of the beam and other side constraints. The problem consists of four variables namely, weld thickness h, length of bar attached to the weld l, bars height t and bars thickness b. The formulation of this problem is found in Coello [7].

Table 4.6 shows the results of the problem when applying EIPA in compar-

Table 4.3 Numerical comparisons between EIPA and ALGO1

Prob.	n/me/mi	IP	Code	NI	NF	FV	CPU
HS6	2/1/0	$(6,6)^T$	EIPA	3	6	0	0.005
			ALGO1	9	364	$2.4199e-007$	0.03
HS7	2/1/0	$(0,1)^T$	EIPA	5	10	-1.73205	0.006
	2/1/0		ALGO1	8	15	-1.7320	0.01
HS9	2/1/0	$(0,0)^T$	EIPA	3	4	-0.49999	0.005
			ALGO1	18	34	-0.49985	0.02
HS27	3/1/0	$(0,0,0)^T$	EIPA	3	6	0.04000	0.004
			ALGO1	28	484	0.039958	0.05
HS28	2/0/5	$(0,0,0)^T$	EIPA	1	2	$2.4651e-032$	0.003
			ALGO1	11	38	$7.5674e-008$	0.01
HS29	3/0/1	$(1,1,1)^T$	EIPA	10	19	-22.627417	0.011
			ALGO1	11	24	-22.627	0.01
HS32	3/1/4	$(0.1,0.7,0.1)^T$	EIPA	8	9	1.00000	0.011
			ALGO1	19	33	0.98818	0.02
HS33	3/0/5	$(0,0,3)^T$	EIPA	14	17	-3.99999	0.048
			ALGO1	15	20	-4.5178	0.02
HS40	4/3/0	$(2,-1,0,1)^T$	EIPA	8	11	-0.2500000	0.009
			ALGO1	49	108	-0.25000	0.05
HS42	4/2/0	$(1,1,1,1)^T$	EIPA	4	55	13.8577330	0.042
			ALGO1	36	70	13.883	0.03
HS43	4/0/3	$(0,0,0,0)^T$	EIPA	15	91	-44.000000	0.005
			ALGO1	12	29	-44.000	0.02
HS48	5/2/0	$(3,5,-3,2,-2)^T$	EIPA	1	2	$2.4651e-030$	0.003
			ALGO1	21	55	$3.1361e009$	0.02
HS51	5/3/0	$(2.5,0.5,2,-1,0.5)^T$	EIPA	1	2	$1.7379e-030$	0.004
			ALGO1	29	132	$2.2808e005$	0.03
HS52	5/3/0	$(1,-0.5,-1,0,1)^T$	EIPA	1	2	5.3266475	0.003
			ALGO1	31	45	5.2930	0.03
HS56	7/4/0	$(1,1,1,a,a,a,b)^T$	EIPA	4	5	-3.45600	0.007
			ALGO1	21	43	-2.6183	0.06
HS62	3/1/6	$(0.7,0.2,0.1)^T$	EIPA	12	13	-26272.5	0.031
			ALGO1	8	19	-26273	0.02
HS81	5/3/10	$(-1.7,1,1.5,-0.8,-0.8)^T$	EIPA	5	6	0.0539468	0.009
			ALGO1	19	37	0.064109	0.05
HS93	6/0/8	$(5.5,4.4,12,11.8,0.7,0.8)^T$	EIPA	13	74	135.075	0.065
			ALGO1	21	43	136.29	0.04
HS100	7/0/4	$(-1.7,1,1.5,-0.8,-0.8)^T$	EIPA	21	67	680.630	0.031
			ALGO1	8	22	680.63	0.02

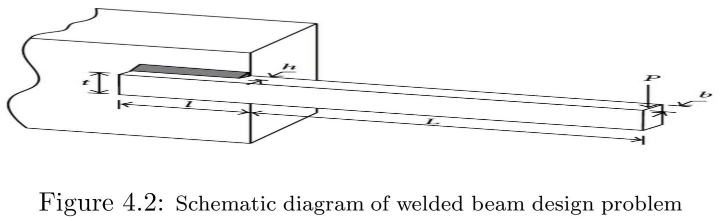

Figure 4.2: Schematic diagram of welded beam design problem

Table 4.4 Numerical comparisons between EIPA, ALGO2 and SNQP

Prob.	n/me/mi	IP	Code	NI	NF	FV	CPU
HS12	2/0/1	$(6,6)^T$	EIPA	7	8	-30.0000	0.007
			ALGO2	20	41	-30.0000	0.06
			SNQP	19	29	-29.9999	–
HS29	3/0/1	$(-4,-4,-4)^T$	EIPA	63	423	-22.6274	0.135
			ALGO2	12	46	-22.6274	0.05
			SNQP	13	42	-22.6274	–
HS31	3/0/7	$(2,4,7)^T$	EIPA	5	6	6.00000	0.006
			ALGO2	17	309	6.00000	0.06
			SNQP	13	42	-22.6274	–
HS33	3/0/6	$(1,4,6)^T$	EIPA	10	67	-4.58578	0.05
			ALGO2	45	570	-4.58578	0.33
			SNQP	23	116	4.58572	–
HS34	3/0/8	$(2,2,2)^T$	EIPA	8	9	-0.83403	0.008
			ALGO2	15	166	-0.83403	0.06
			SNQP	–	–	–	–
HS35	3/0/4	$(1,2,3)^T$	EIPA	10	11	0.11111	0.011
			ALGO2	7	67	0.11111	0.03
			SNQP	13	0	0.11111	–
HS66	3/0/8	$(0,0,100)^T$	EIPA	6	7	0.51816	0.008
			ALGO2	64	1067	0.518163	0.48
			SNQP	–	–	–	–
HS76	4/0/7	$(1,2,3,4)^T$	EIPA	13	14	-4.68181	0.016
			ALGO2	21	345	-4.68181	0.11
			SNQP	16	0	-4.68181	–

ing with those of GSA [23], GWO [22], CGWO [20], IPTRA [1] and CGPSO [3]. The results show that EIPA has the best optimum cost relative to the one obtained by GSA [23], GWO [22] and CGWO [20]. However, EIPA is almost the same as the one obtained from IPTRA [1] and CGPSO [3].

4.2.3 Multistage Heat Exchanger Design Problem

This problem is solved by Avriel et al [6]. The objective of this problem is to minimize the sum of the heat transfer areas of the three exchangers (as shown in Fig. 4.3) subject to six inequality constraints. The design variables are heat transfer areas of the three exchangers A_1, A_2 and A_3, the temperatures of the main fluid produced from stage (1) and (2), T_1 and T_2 and temperatures of the hot fluid entering the three heat exchangers t_{11}, t_{21} and t_{31}.

Table 4.7shows the results obtained by EIPA, the algorithm of Avriel et al [6], BA [26] and IPTRA [1]. EIPA produces an optimum solution almost the same as those from Avriel [6] and IPTRA [1]. However, EIPA has better

Table 4.5 Numerical results of tension/compression spring design problem

Design variables	GSA (2014)	GWO (2014)	CGWO (2016)	IPTRA (2018)	CGPSO (2019)	EIPA
D	0.050276	0.051690	0.052796	0.051689	–	0.106382
d	0.323680	0.323680	0.804380	0.356717	–	0.250000
N	13.525410	13.525410	2.0000000	11.288965	–	2.0000000
f	0.0127022	0.0127022	0.0119598	0.0126652	0.0126722	0.0113171

Table 4.6 Numerical results of welded beam design problem

Design variables	GSA (2014)	GWO (2014)	CGWO (2016)	IPTRA (2018)	CGPSO (2019)	EIPA
h	0.1821	0.2056	0.343891	0.205727	–	0.205742
l	3.8569	3.4783	1.883570	3.470389	–	3.470664
t	9.0368	9.0368	9.03133	9.036980	–	9.036346
b	0.2057	0.2057	0.212121	0.205727	–	0.205742
f	1.8799	1.7262	1.72545	1.724884	1.72489	1.72494

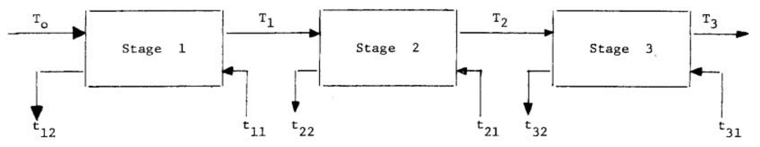

Figure 4.3: Schematic diagram of multistage heat exchanger design problem

optimum solution than the one from BA [26].

5 Conclusion

In this paper, we have proposed a new algorithm for solving problem (NLP) by extending Goodman's method [14] for solving Problem (EQ) to Problem (NLP). The main result of this paper is the formulation of the reduced linear system of dimension $(n \times n)$ which we need to solve at each iteration to generate the next iterate point. This result overcomes the disadvantage of solving the extended system of dimension $(n+m) \times (n+m)$ suggested by Das [9], Dennis et al [10] and El-Alem et al [12]. The numerical results carried out on some standard test problems and three engineering design problems. The results show efficiency of EIPA compared to other algorithms.

Table 4.7.1 Numerical results of multistage heat exchanger design problem

Design variables	Avriel (1971)	BA (2012)	IPTRA (2018)	EIPA
A_1	567	579.30675	579.30668443	579.306684425
A_2	1357	1359.97076	1359.970668094	1359.970668051
A_3	5125	5109.97052	5109.9706669	5109.9706680
T_1	181	182.01770	182.017699592	182.017699581
T_2	295	295.60118	295.60117330	295.60117327
t_{11}	219	217.98230	217.982300431	217.982300418
t_{21}	286	286.41653	286.416526324	286.416526303
t_{31}	395	395.60118	395.60117331	395.60117327
f	7049	7049.24803	7049.24801950	7049.24802052

References

[1] Abdelkader, S., EL-Sobky, B., EL-Alem, M. (2018). *A computationally practical interior-point trust-region algorithm for solving the general nonlinear programming problems.* Southeast-Asian J. of Sciences. 6:39-55.

[2] Abdelkader, S., "A trust-region algorithm for solving the general nonlinear programming problem", Ph.D thesis, Alexandria University, Alexandris, Egypt (2018).

[3] Abdelhalim, A., Nakata, K., El-Alem, M., Eltawil, A. (2019). *A hybrid evolutionary-simplex search method to solve nonlinear constrained optimization problems.* Soft Computing. https://doi.org/10.1007/s00500-019-03756-3

[4] Argaez, M., Tapia, R. (2002). *On the global convergence of a modified augmented lagrangian linesearch interior-point Newton method for nonlinear programming.* Journal of Optimization Theory and Applications. 114:1-25.

[5] Arora, J. (1989). *Introduction to Optimum Design.* McGraw-Hill, New York..

[6] Avriel, M., Williams, A. (1971). *An extension of geometric programming with applications in engineering optimization.* Journal of Engineering Mathematics. 5:458-72.

[7] Coello, C. (2000). *Use of a self-adaptive penalty approach for engineering optimization problems.* Comput Ind. 41(2):113-127.

[8] Coleman, T., Li, Y. (1996). *An interior trust region approach for nonlinear minimization subject to bounds.* SIAM J. Optimization. 6:418-445.

[9] Das, I. (1996). *An interior-point algorithm for the general nonlinear programming problem with trust-region globalization* . Technical Report 96-61. Institute for Computer Applications in Science and Engineering, NASA Langley Research Center Hampton, VA, USA.

[10] Dennis, J., Heinkenschloss, M., Vicente, L. (1998). *Trust-Region interior-point sqp algorithms for a class of nonlinear programming problems.* SIAM Journal on Control and Optimization. 36:1750-1794.

[11] El-Alem, M. (1999). *A global convergence theory for Dennis ,El-Alem and Maciel's class of trust-region algorithms for constrained optimization without assuming regularity.* SIAM J. Optimization. 9:965-990.

[12] El-Alem, M., El-Sayed, S., El-Sobky, B. (2004). *Local convergence of the interior-point Newton method for general nonlinear programming.* Journal of Optimization Theory and Applications. 120:487-502.

[13] El-Bakry, A., Tapia, R., Tsuchiya, T., Zhang, Y. (1996). *On the formulation and theory of the Newton interior-point method for nonlinear programming.* Journal of Optimization Theory and Application. 89:507-541.

[14] Goodman, J. (1985). *Newton's method for constrained optimization.* Math Programming. 33:162-171.

[15] Hock, W., Schittkowski, K. (1981). *Test examples for nonlinear programming codes.* Springer. 187.

[16] Jian, J., Guo, C., Tang, C., Bai, Y. (2014). *A new superlinearly convergent algorithm of combining QP subproblem with system of linear equations for nonlinear optimization.* Journal of Computational and Applied Mathematics. 273:88-102.

[17] Jian, J., Ke, X., Zheng, H., Tang, C. (2009). *A method combining norm-relaxed QP subproblems with systems of linear equations for constrained optimization.* J. Comput. Appl. Math. 223:1013-1027.

[18] Jian, J., Zeng, H., Ma, G., Zhu, Z. (2017). *Primal-dual interior point QP-free algorithm for nonlinear constrained optimization.* Journal of Inequalities and Applications. 1-25.

[19] Karmarkar, N. (1984). *A new polynomial-time algorithm for linear programming.* Combinatorica. 4:373-395.

[20] Kohli, M., Arora, S. (2017), *Chaotic grey wolf optimization algorithm for constrained optimization problems.* Journal of Computational Design and Engineering. 5:458-472.

[21] Maratos, N. (1978). *Exact penalty function algorithms for finite dimensional and control optimization problems.* Ph.D. thesis. London university.

[22] Mirjalili, S., Mirjalili, S., Lewis, A. (2014). *Grey wolf optimizer.* Advances in Engineering Software. 69:46-61.

[23] Mirjalili S., Lewis, A. (2014). *Adaptive gbest-guided gravitational search algorithm.* Neural Comput Appl 25(7):1569-1584.

[24] Nocedal, J., Overton, M. (1985). *Projected Hessian updating algorithms for nonlinearly constrained optimization.* SIAM J. NUMER. ANAL. 22:821-850.

[25] Nocedal, J., Wright, S. (1999). *Numerical optimization.* Springer-Verlag New York Berlin Heidelberg.

[26] Yang, X., Gandomi, A. (2012). *Bat algorithm:A novel approach for global engineering optimization.* Engineering Computations. 29(5):464-483.