Southeast-Asian J. of Sciences, Vol. 6, No. 1 (2018) pp. 39-55

A COMPUTATIONALLY PRACTICAL INTERIOR-POINT TRUST-REGION ALGORITHM FOR SOLVING THE GENERAL NONLINEAR PROGRAMMING PROBLEMS

S. Y. Abdelkader*†, B. EL-Sobky*‡ and M. EL-Alem*

Department of Mathematics, Faculty of Science,
Alexandria University, Alexandria, Egypt.
e-mail: shyashraf@yahoo.com;bothinaelsobky@yahoo.com;
mmelalem@yahoo.com; and mmelalem@hotmail.com

Abstract

An interior-point trust-region algorithm for solving the general nonlinear programming problem is proposed. In the algorithm, an interior-point Newton method with Coleman-Li scaling matrix is used. A trust-region globalization strategy is added to the algorithm to insure global convergence. A projected Hessian technique is used to simplify the trust-region subproblems.

A Matlab implementation of the algorithm was used and tested against some existing codes. In addition, four case studies were presented to test the performance of the proposed algorithm. The results showed that the algorithm out perform some existing methods in literature.

Key words: Newton's method, interior-point, trust-region, projected-Hessian, nonlinear programming, Matlab implementations, numerical comparisons, case study.
MSC 2010 : 90C30, 90C55, 65K05, 49M37.

1 Introduction

Various approaches have been proposed and used to solve the following general nonlinear programming problem

$$\begin{array}{ll} \textit{minimize} & f(x) \\ \textit{subject to} & C(x) = 0, \\ & a \leq x \leq b, \end{array} \tag{1.1}$$

where $f : \Re^n \rightarrow \Re$, $C : \Re^n \rightarrow \Re^m$, $a \in \{\Re \bigcup \{-\infty\}\}^n$, $b \in \{\Re \bigcup \{+\infty\}\}^n$, $m < n$, and $a < b$. We assume that the functions f and C are at least twice continuously differentiable.

The Lagrangian function associated with Problem (1.1) is given by

$$L(x, \lambda, \mu, \nu) = \ell(x, \lambda) - \mu^T(x - a) - \nu^T(b - x), \tag{1.2}$$

where $\ell(x, \lambda) = f(x) + \lambda^T C(x)$ and λ, μ, and ν are the Lagrange multiplier vectors associated with the equality constraint $C(x) = 0$, and the inequality constraints $(x - a) \geq 0$ and $(b - x) \geq 0$ respectively.

The first-order necessary conditions for a point x_* to be a solution of problem (1.1) are the existence of multipliers $\lambda_* \in \Re^m$, $\mu_* \in \Re^n_+$, and $\nu_* \in \Re^n_+$, such that $(x_*, \lambda_*, \mu_*, \nu_*)$ satisfies

$$\nabla_x \ell(x_*, \lambda_*) - \mu_* + \nu_* = 0, \tag{1.3}$$

$$C(x_*) = 0, \tag{1.4}$$

$$a \leq x_* \leq b, \tag{1.5}$$

and for all i corresponding to $x^{(i)}$ with finite bound, we have

$$\mu_*^{(i)}(x_*^{(i)} - a^{(i)}) = 0, \tag{1.6}$$

$$\nu_*^{(i)}(b^{(i)} - x_*^{(i)}) = 0, \tag{1.7}$$

In addition to that, for any i corresponding to $x^{(i)}$ with infinite bound the corresponding $\mu_*^{(i)}$ or $\nu_*^{(i)}$ is zero.

Motivated by the strategy in [12], we define the diagonal scaling matrix $D(x)$ whose diagonal elements are given by

$$d^{(i)}(x) = \begin{cases} \sqrt{(x^{(i)} - a^{(i)})}, & \text{if } \nabla_x \ell(x, \lambda) \geq 0 \text{ and } a^{(i)} > -\infty, \\ \sqrt{(b^{(i)} - x^{(i)})}, & \text{if } \nabla_x \ell(x, \lambda) < 0 \text{ and } b^{(i)} < +\infty \ , \\ 1, & \text{otherwise.} \end{cases} \tag{1.8}$$

The scaling matrix $D(x)$ was first introduced in [6] for unconstrained optimization problem with simple bound and was used by [7], [12],[13].

Using the scaling matrix $D(x)$, the first-order necessary conditions (1.3)-(1.7) are equivalent to the following conditions

$$\begin{aligned} D^2(x)\nabla_x \ell(x,\lambda) &= 0, && (1.9)\\ C(x) &= 0, && (1.10)\\ a \leq x &\leq b. && (1.11) \end{aligned}$$

By applying Newton's method on the nonlinear system (1.9),(1.10), we obtain

$$\begin{aligned} [D^2(x)\nabla_x^2 \ell(x,\lambda) &+ diag(\nabla_x \ell(x,\lambda))diag(\eta(x))]\Delta x && (1.12)\\ &+ D^2(x)\nabla C(x)\Delta\lambda = -D^2(x)\nabla_x \ell(x,\lambda),\\ \nabla C(x)^T \Delta x &= -C(x), && (1.13) \end{aligned}$$

where $\eta^{(i)}(x) = \frac{\partial((d^{(i)}(x))^2)}{\partial x^{(i)}}, \quad i = 1, \ldots, n$. More details about the derivation of equations (1.13) is given in [7],[12].

Inforcing $a < x < b$ makes $D(x)$ nonsingular. Now multiplying both sides of Equation (1.13) by $D^{-1}(x)$, we obtain

$$\begin{aligned} [D(x)\nabla_x^2 \ell(x,\lambda) &+ D^{-1}(x)diag(\nabla_x \ell(x,\lambda))diag(\eta(x))]\Delta x\\ &+ D(x)\nabla C(x)\Delta\lambda = -D(x)\nabla_x \ell(x,\lambda),\\ \nabla C(x)^T \Delta x &= -C(x). \end{aligned}$$

If we scale the step using $\Delta x = D(x)s$, the above system will have the form

$$Bs + D(x)\nabla C(x)\Delta\lambda = -D(x)\nabla_x \ell(x,\lambda), \quad (1.14)$$

$$(D(x)\nabla C(x))^T s = -C(x). \quad (1.15)$$

where

$$B = D(x)\nabla_x^2 \ell(x,\lambda)D(x) + diag(\nabla_x \ell(x,\lambda))diag(\eta(x)).$$

The above system shares the advantages and the disadvantages of Newton's method. Form Newton's good side, under the standard assumptions for Newton's method for problem (1.1), the method converges quadratically to a stationary point (x_*, λ_*) [12]. On the other side, it has the disadvantage of local convergence. This means that the starting point (x_0, λ_0) must be sufficiently closed to (x_*, λ_*) in order to guarantee convergence. In other word, it may not converge at all if the starting point is far away from the solution.

Trust-region approach is a very successful approach to insure global convergence from any starting point, see [6],[11]. To add a trust-region constraint, we have to rewrite the extended system (1.14)-(1.15) as a minimization problem. An equivalent problem is the following quadratic programming problem

$$\begin{array}{ll} minimize & (D\nabla_x \ell)^T s + \frac{1}{2} s^T B s\\ subject\ to & (D\nabla C)^T s + C = 0 \end{array} \quad (1.16)$$

Notice that the first order necessary conditions of problem (1.16) coincides with (1.14)-(1.15).

If a trust-region constraint is simply added to Problem (1.16), the resulting problem will take the form

$$\begin{array}{ll} \textit{minimize} & (D_k \nabla_x \ell_k)^T s + \frac{1}{2} s^T B_k s \\ \textit{subject to} & (D_k \nabla C_k)^T s + C_k = 0, \\ & \|s\|_2 \leq \delta_k. \end{array} \tag{1.17}$$

But this trust-region subproblem may be infeasible because the intersecting points between the trust-region constraint and the hyperplane of the linearized constraints may not exist. Even if they intersect, there is no guarantee that the intersecting set will remain nonempty if the trust-region radius is decreased.

The reduced Hessian is a successful approach to overcome the difficulty of having a possible infeasible trust-region subproblem. This approach was suggested by Byrd [4] and Omojokun [19].

The following notations are used throughout the rest of the paper. A subscripted function means the value of the function evaluated at a particular point. For example, $f_k \equiv f(x_k)$, $C_k \equiv C(x_k)$, $D_k \equiv D_k(x_k)$ and so on. We use the notation $\nabla_x \ell_k^{(i)}$ to denote the ith component of the vector $\nabla_x \ell_k$ and $x_k^{(i)}$ to denote the ith component of the vector x_k, and so on. Finally, all norms used in this paper are ℓ_2-norms.

The paper is organized as follows. In section 2, a detailed description of the main steps of the interior-point trust-region Algorithm IPTRA is given. Section 3 contains a Matlab implementation and reports of the numerical results of Algorithm IPTRA. Section 4 contains four case studies of Algorithm IPTRA. Finally, Section 5 contains some concluding remarks.

2 Description of the Algorithm

In this section a detailed description of the proposed interior-point trust-region algorithm (IPTRA) for solving problem (1.1) is given.

2.1 Computing a trial step

The reduced-Hessian approach is used to compute a trial step s_k. In this approach, the trial step s_k is decomposed into two orthogonal components; the normal component s_k^n and the tangential component s_k^t. The trial step s_k has the form $s_k = s_k^n + Z_k \bar{s}_k^t$, where Z_k is a matrix whose columns form an orthonormal basis for the null space of $(D_k \nabla C_k)^T$.

We obtain the normal component s_k^n by solving the following trust-region subproblem

$$\begin{array}{ll} \textit{minimize} & \frac{1}{2}\|(D_k \nabla C_k)^T s^n + C_k\|^2 \\ \textit{subject to} & \|s^n\| \leq \zeta \delta_k, \end{array} \tag{2.1}$$

for some $\zeta \in (0,1)$, where δ_k is the trust-region radius.

Given the normal component s_k^n, the step $\bar{s}_k^t$ is computed by solving the following trust-region subproblem

$$\begin{array}{ll} \textit{minimize} & [Z_k^T(D_k\nabla_x \ell_k + B_k s_k^n)]^T \bar{s}^t + \frac{1}{2}\bar{s}^{t^T} Z_k^T B_k Z_k \bar{s}^t \\ \textit{subject to} & \|Z_k \bar{s}^t\| \leq \Delta_k, \end{array} \tag{2.2}$$

where $\Delta_k = \sqrt{\delta_k^2 - \|s_k^n\|^2}$. Once the step $\bar{s}_k^t$ is computed, the tangential component s_k^t is given by $s_k^t = Z_k \bar{s}_k^t$. To solve the trust-region subproblems (2.1) and (2.2) we use the dogleg algorithm.

Having computed the trial step s_k, the scaled step $\Delta x_k = D_k s_k$, is computed. A damping parameter τ_k is needed to ensure that the new point $x_k + \Delta x_k$ lies inside the box constraint. It is computed using the following Scheme:

Scheme 2.1. *(computing τ_k)*

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

$$\textit{Compute } v_k^{(i)} = \begin{cases} \frac{b^{(i)} - x_k^{(i)}}{\Delta x_k^{(i)}}, & \textit{if } b^{(i)} < \infty \textit{ and } \Delta x_k^{(i)} > 0 \\ 1, & \textit{otherwise.} \end{cases}$$

Set $\tau_k = \min\{1, \min_i\{u_k^{(i)}, v_k^{(i)}\}\}$

Since it is always require that $\{x_k\}$ satisfy, for all k, $a < x_k < b$, another damping θ_k in the step may be needed to insure $a < x_k < b$. This can be stated in algorithmic form as follows

Scheme 2.2. *(computing θ_k)*

If $a < x_k + \tau_k \Delta x_k < b$, *then set* $\theta_k = 1$.

Else choose $\theta_k \in (0.99, 1)$.

End if.

After computing the scaled step Δx_k, we set the trial step $\omega_k = \theta_k \tau_k \Delta x_k$ and $x_{k+1} = x_k + \omega_k$. Estimates for the Lagrange multiplier λ_{k+1} is needed. To estimate the Lagrange multiplier λ_{k+1} we solve

$$\min_{\lambda \in R^m} \quad \|\nabla f_{k+1} + \nabla C_{k+1}\lambda\|^2. \tag{2.3}$$

We test whether the point (x_{k+1}, λ_{k+1}) will be accepted and taken as a next iterate. To test for that, a merit function is needed. The following augmented Lagrangian

$$\Phi(x, \lambda; \rho) = f(x) + \lambda^T C(x) + \rho\|C(x)\|^2, \tag{2.4}$$

is used as a merit function, where $\rho > 0$ is the penalty parameter. The actual reduction in the merit function in moving from (x_k, λ_k) to (x_{k+1}, λ_{k+1}) is defined as

$$Ared_k = \Phi(x_k, \lambda_k; \rho_k) - \Phi(x_{k+1}, \lambda_{k+1}; \rho_k). \tag{2.5}$$

The predicted reduction in the merit function is defined as

$$\begin{aligned} Pred_k &= -\nabla_x \ell(x_k, \lambda_k)^T \omega_k - \frac{1}{2}\omega_k^T \nabla_x^2 \ell_k \omega_k - \Delta\lambda_k^T (C_k + \nabla C_k^T \omega_k) \\ &\quad + \rho_k [\|C_k\|^2 - \|C_k + \nabla C_k^T \omega_k\|^2], \end{aligned} \tag{2.6}$$

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

2.2 Updating the penalty parameter ρ_k

The penalty parameter ρ_k is updated to ensure that

$$Pred_k \geq \frac{\rho_k}{2}\|C_k\|^2 - \|C_k + \nabla C_k^T \omega_k\|^2.$$

To update ρ_k, we use the scheme proposed in [10].

Scheme 2.3. *(computing ρ_k)*
Set $\rho_k = \rho_{k-1}$.
If $Pred_k < \frac{\rho_k}{2}[\|C_k\|^2 - \|C_k + \nabla C_k^T \omega_k\|^2]$, then set

$$\rho_k = \frac{2[\nabla_x \ell(x_k, \lambda_k)^T \omega_k + \frac{1}{2}\omega_k^T \nabla_x^2 \ell_k \omega_k + \Delta\lambda_k^T (C_k + \nabla C_k^T \omega_k)]}{\|C_k\|^2 - \|C_k + \nabla C_k^T \omega_k\|^2} + \beta, \tag{2.7}$$

where $\beta > 0$ is a small fixed constant.

2.3 Testing the Step and Updating δ_k

The point (x_{k+1}, λ_{k+1}) needs to be tested to determine whether it will be accepted. We do this by comparing $Ared_k$ to $Pred_k$, as in the following scheme.

Scheme 2.4. *(testing (x_{k+1}, λ_{k+1}) and uptading δ_k)*
If $\frac{Ared_k}{Pred_k} < \gamma_1$, where $0 < \gamma_1 < 1$.

> *Reduce the trust-region radius by setting $\delta_k = \alpha_1 \|\Delta x_k\|$, where $\alpha_1 \in (0, 1)$.*
> *Compute another trial point (x_{k+1}, λ_{k+1}).*

Else if $\gamma_1 \leq \frac{Ared_k}{Pred_k} < \gamma_2$, $0 < \gamma_1 < \gamma_2 < 1$, then

Accept the point (x_{k+1}, λ_{k+1}).
Set the trust-region radius: $\delta_{k+1} = \max(\delta_k, \delta_{min})$, *where* δ_{min} *is a fixed constant.*

Else

Accept the point (x_{k+1}, λ_{k+1}).
Set the trust-region radius: $\delta_{k+1} = \min\{\delta_{max}, \max\{\delta_{min}, \alpha_2\delta_k\}\}$, *where* δ_{max} *is a fixed constant,* $(\delta_{max} > \delta_{min})$ *and* $\alpha_2 > 1$.

End if.

Finally, the algorithm is terminated when $\|D_k\nabla_x\ell_k\| + \|C_k\| \leq \varepsilon$, for some $\varepsilon > 0$. A formal description of the proposed interior-point trust-region algorithm (IPTRA) for solving problem (1.1) is presented in Algorithm 1.

Algorithm 1. *(IPTRA)*
Step 0. (Initialization)

Given $x_0 \in \Re^n$ *such that* $a < x_0 < b$. *Evaluate* λ_0, D_0.
Set $\rho_{-1} = 1$ *and* $\beta = 0.1$. *Choose* ε, σ, α_1, α_2, γ_1, *and* γ_2 *such that* $\varepsilon > 0$, $\sigma > 0$, $0 < \alpha_1 < 1 < \alpha_2$, *and* $0 < \gamma_1 < \gamma_2 < 1$. *Choose* δ_{min}, δ_{max}, *and* δ_0 *such that* $\delta_{min} < \delta_{max}$, $\delta_0 \in [\delta_{min}, \delta_{max}]$. *Set* $k = 0$.

Step 1. (Test for convergence)

If $\|D_k\nabla_x\ell_k\| + \|C_k\| \leq \varepsilon$, *then terminate the algorithm.*

Step 2. (Compute a trial step)

If $\|C_k\| = 0$, *then*

a) Set $s_k^n = 0$.
b) Compute the step $\bar{s}_k^t$ *by solving Subproblem (2.2)*
c) Set $s_k = Z_k\bar{s}_k^t$.

Else

a) Compute the normal component s_k^n *by solving Subproblem (2.1).*
b) If $\|Z_k^T(D_k\nabla_x\ell_k + B_k s_k^n)\| = 0$, *then set* $\bar{s}_k^t = 0$.
Else, compute $\bar{s}_k^t$ *by solving subproblem (2.2).*
End if.
c) Set $s_k = s_k^n + Z_k\bar{s}_k^t$, $\Delta x_k = D_k s_k$.

End if.

Step 3. (Test for the box interiority)

a) Compute the damping parameter τ_k using Scheme 2.1.
b) Set $x_{k+1} = x_k + \tau_k \Delta x_k$.
c) Compute θ_k according to Scheme 2.2

Step 4. (Compute Lagrange multipliers λ_{k+1})

Compute λ_{k+1} by solving Subproblem (2.3).

Step 5. (Update the scaling matrix)

Compute D_{k+1}.

Step 6. (Update the penalty parameter ρ_k)

Updating ρ_k according to Scheme 2.3

Step 7. (Test the step and update the trust-region radius)

Test the step and update δ_k according to Scheme 2.4

Step 8. Set $k = k + 1$ and go to Step 1.

3 Numerical results

In this section, we report our numerical experience with the proposed trust-region algorithm IPTRA for solving Problem (1.1). Our program was written in MATLAB and run under MATLAB Version 7 with machine epsilon about 10^{-16}.

Given a starting point x_0 such that $a < x_0 < b$, we chose the initial trust-region radius $\delta_0 = max(\|s_0\|, \delta_{min})$, where $\delta_{min} = 10^{-3}$ and s_0 is the full Cauchy step of the constraints $C(x)$ at x_0 (i.e. $s_0 = -\frac{\nabla C_0^T \nabla C_0}{\nabla C_0^T \nabla^2 C_0 \nabla C_0} \nabla C_0^T C_0$). We chose the maximum trust-region radius to be $\delta_{max} = 10^5 \delta_0$. The values of the constants that are needed in step 0 of Algorithm IPTRA were set to be $\gamma_1 = 10^{-4}$, $\gamma_2 = 0.5$, $\alpha_1 = 0.5$, $\alpha_2 = 2$, $\varepsilon = 10^{-8}$, $\varepsilon_1 = 10^{-8}$, $\varepsilon_2 = 10^{-8}$. $\theta = .9995$ and $\beta = 0.1$.

For computing the component of the trial steps, we have used the dogleg algorithm. Successful termination with respect to the proposed trust-region algorithm means that the termination condition of the algorithm is met with $\varepsilon = 10^{-8}$. On the other hand, unsuccessful termination means that the number of iterations is greater than 500 or the number of function evaluations is greater than 1000.

Numerical results obtained using Algorithm IPTRA have been reported and summarized in Tables (3.1) and (3.2). The problems which were tested in these tables were taken from Hock and Schittkowski [16]. The following abbreviations have been used:

HS : The number of problem as it is given in Hock and Schittkowski [16].
n : number of variables.
me : number of equality constraints.
mi : number of inequality constraints.
Niter : number of iterations of Algorithm IPTRA.
Nfunc : number of function evaluations of Algorithm IPTRA.
ngrad : value of $\|D_k \nabla_x \ell_k\| + \|C_k\|$.
Fiter: number of iteration of Fletcher's algorithm [14].
Ffunc: number of function evaluation of Fletcher's algorithm [14].

The numerical results of Algorithm IPTRA for some test problems of Hock and Schittkowski [16] at a feasible starting point with respect to the bounds have been listed. A comparison of our results with those obtained by Matlab is presented in Table (3.1).

A comparison of our numerical results against the corresponding results of Fletcher [14] at the same starting point indicated in Hock and Schittkowski is presented in Table (3.2). Solution obtained by the proposed algorithm are exactly identical to those given by Hock and Schittkowski.

4 Case Studies

In this section, four mechanical design problems are presented as case studies IPTRA algorithm.

Case 1. *Design of a Pressure Vessel [17]*
A cylindrical vessel is capped at both ends by hemispherical heads as shown in figure (4.1). The objective is to minimize the total cost, including the cost of the material, forming and welding. There are four design variables: T_s (thickness of the shell), T_h (thickness of the head), R (inner radius) and L(length of the cylindrical section of the vessel not including the head). T_s and T_h are integer multiples of 0.0625 inch, which are the available thicknesses of rolled steel plates, and R and L are continuous variables.

Table 3.1: Numerical results of the Algorithm IPTRA for the Hock and Schittkowski's test problems.

problem	n	me	mi	starting point	Algorithm IPTRA ngrad	Algorithm IPTRA $\frac{\#Niter}{\#Nfunc}$	Matlab $\frac{\#Niter}{\#Nfunc}$
HS1	2	0	1	(-2,1)	9.6006e-008	24/29	37/121
HS17	2	0	5	(0,1)	9.8648e-009	7/8	10/42
HS20	2	0	5	(0,1)	2.3282e-007	6/7	7/24
HS21	2	0	5	(5,2)	4.0984e-014	4/5	11/36
HS24	2	0	5	(1,0.5)	4.9898e-016	5/6	15/49
HS30	3	0	7	(2,1,1)	5.4610e-008	5/6	6/28
HS31	3	0	7	(2,2,0)	1.4414e-009	6/7	12/59
HS34	3	0	8	(5,2,3)	1.1253e-012	9/10	19/81
HS35	3	0	4	(0.5,0.5,0.5)	2.1551e-008	6/7	11/48
HS36	3	0	7	(10,10,10)	4.8587e-010	6/7	7/32
HS38	4	0	8	(3,1,3,1)	1.4350e-007	11/12	55/308
HS41	4	1	8	(0.5,0.5,0.5,1)	1.4168e-008	5/6	16/85
HS45	5	0	10	(0.5,0.7,1,2,3)	5.5541e-009	7/8	19/125
HS53	5	3	10	(2,2,2,2,2)	8.0960e-008	4/5	8/55
HS55	6	6	8	(0.5,1,1,0.5,1,2)	2.1991e-008	6/7	9/71
HS65	3	0	7	(1,1,0)	1.0674e-010	9/10	fail
HS66	3	0	8	(3,1.5,2)	2.5305e-009	8/9	14/60
HS71	4	1	9	(2,4,4,2)	1.2910e-010	6/7	8/46
HS74	4	3	10	(1,1,0,0)	4.8247e-007	15/16	15/79
HS75	4	3	10	(1,1,0,0)	1.4066e-009	16/17	10/55

Table 3.2: Numerical results of IPTRA and the corresponding results of Fletcher's algorithm.

problem	n	me	mi	Algorithm IPTRA # Niter / # Nfunc	filterSD # Fiter/ # Ffunc
HS6	2	1	0	16 / 28	4 / 8
HS12	2	0	1	5/12	8/23
HS19	2	0	6	7 / 8	6 / 7
HS23	2	0	9	8 / 9	6 / 6
HS26	3	1	0	19/ 23	5/61
HS32	3	1	4	6 / 7	3 / 15
HS39	4	2	0	9 / 10	20 / 121
HS43	4	0	3	8/10	8/57
HS60	3	1	6	7 / 8	3 / 35
HS63	3	2	3	7 / 8	9 / 24
HS80	5	2	10	6 / 7	6 / 30
HS81	5	3	10	6 / 7	11 / 103
HS93	6	0	8	10 / 17	4 / 213

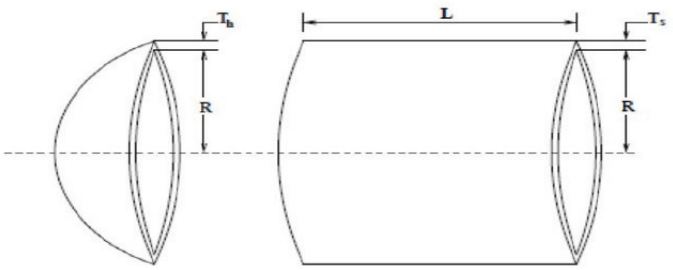

Figure 4.1: The pressure vessel design problem

Using the same notation given in [17], the problem can be stated as follows:

$$\begin{aligned}
&\textit{minimize } f(x) = 0.6224x_1x_3x_4 + 1.7781x_2x_3^2 + 3.1661x_1^2x_4 + 19.84x_1^2x_3\\
&\textit{subject to}\\
&\qquad g_1(x) = -x_1 + 0.0193x_3 \leq 0\\
&\qquad g_2(x) = -x_2 + 0.00954x_3 \leq 0\\
&\qquad g_3(x) = -\pi x_3^2x_4 - (4/3)\pi x_3^3 + 1296000 + \leq 0\\
&\qquad g_4(x) = x_4 - 240 \leq 0\\
&\qquad 0 \leq x_1, x_2 \leq 100,\\
&\qquad 10 \leq x_3, x_4 \leq 200.
\end{aligned}$$

A comparison of best solutions of this problem using different optimization techniques against ours is presented in Table (4.1). From Table (4.1), it can be seen that the solution found by IPTRA is better than the solutions found by other techniques which listed in the table.

Table 4.1: Comparison of the best solutions for the pressure vessel design problem.

Design variables	Kannan and Kramer (1994)[17]	Deb (1997)[9]	He and Wang (2007)[15]	Lobato and Steffen (2014)[18]	Algorithm IPTRA IPTRA
x_1	1.125000	0.937500	0.812500	0.812500	0.7781686412897
x_2	0.625000	0.500000	0.437500	0.437500	0.3846491626309
x_3	58.29100	48.32900	42.09126	42.09127	40.3196187240987
x_4	43.69000	112.6790	176.7465	176.7466	200
f	7198.0428	6410.3811	6061.0777	6061.0778	5885.332773005870

Case 2. *Welded Beam Design [20]*
A welded beam is designed for minimum cost subject to constraints on shear stress (τ), bending stress in the beam (σ) buckling load on the bar (P_c), end deflection of the beam(δ), and side constraints. There are four design variables as shown in figure (4.2), $h(x_1)$, $l(x_2)$, $t(x_3)$ and $b(x_4)$. The problem can be

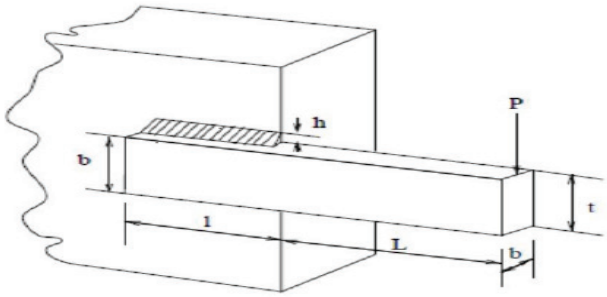

Figure 4.2: The welded beam design problem

stated as follows:

$$
\begin{aligned}
&Minimize\ f(x) = 1.10471x_1^2x_2 + 0.04811x_3x_4(14.0 + x_2)\\
&\ subject\ to\\
&\qquad g_1(x) = \tau(x) - \tau_{max} \leq 0\\
&\qquad g_2(x) = \sigma(x) - \sigma_{max} \leq 0\\
&\qquad g_3(x) = x_1 - x_4 \leq 0\\
&\qquad g_4(x) = 0.10471x_1^2 + 0.04811x_3x_4(14.0 + x_2) - 5.0 \leq 0\\
&\qquad g_5(x) = 0.125 - x_1 \leq 0\\
&\qquad g_6(x) = \delta - \delta_{max} \leq 0\\
&\qquad g_7(x) = P - P_c \leq 0\\
&\qquad \tau = \sqrt{(\tau_1)^2 + 2\tau_1\tau_2\frac{x_2}{2R} + (\tau_2)^2}\\
&\qquad \tau_1 = \frac{P}{\sqrt{2}x_1x_2}\\
&\qquad \tau_2 = \frac{MR}{J}\\
&\qquad M = P(L + \frac{x_2}{2})
\end{aligned}
$$

$$R = \sqrt{\frac{x_2^2}{4} + (\frac{x_1 + x_3}{2})^2}$$
$$J = 2\{\sqrt{2}x_1x_2[\frac{x_2^2}{12} + (\frac{x_1 + x_3}{2})^2]\}$$
$$\sigma(x) = \frac{6PL}{(x_4x_3^2)}$$
$$\delta(x) = \frac{4PL^3}{(Ex_4x_3^3)}$$
$$P_c = \frac{4.013E\sqrt{\frac{x_3^2x_4^6}{36}}}{L^2}(1 - \frac{x_3}{2L}\sqrt{\frac{E}{4G}})$$
$$0.1 \leq x_1, x_4 \leq 2$$
$$0.1 \leq x_2, x_3 \leq 10$$
$$P = 6000, L = 14, E = 30 \times 10^6, G = 12 \times 10^6$$
$$\tau_{max} = 13600, \sigma_{max} = 30000, \delta_{max} = 0.25$$

Table 4.2 presents a comparison between best solutions of different optimization techniques against ours for this problem. From which it can be seen that IPTRA solution is better than the solutions found by other techniques which listed in the table.

Table 4.2: Comparison of the best solutions for the welded beam design problem.

Design variables	Deb (1991)[8]	Coello (2000)[5]	He and Wang (2007)[15]	Lobato and Steffen (2014)[18]	Algorithm IPTRA
x_1	0.248900	0.208800	0.202369	0.208796	0.205727860241585
x_2	6.173000	3.420500	3.544214	3.412545	3.470389674407195
x_3	8.178900	8.997500	9.048210	8.910044	9.036980598594390
x_4	0.253300	0.210000	0.205723	0.210001	0.205727860244276
f	2.433116	1.748309	1.728024	1.7318117	1.724884179475869

Case 3. *Minimization the Weight of a Tension/Compression String*
This problem was proposed by Arora [1], Belegundu [3] and He and Wang [15]. It consists of minimizing the weight of a tension compression spring shown in figure (4.3) subject to constraints on minimum deflection, shear stress, surge frequency, limits on outside diameter and on design variables. The design variables are the mean coil diameter D, the wire diameter d and the number of active coils N.

The mathematical formulation of this problem can be described as follows:

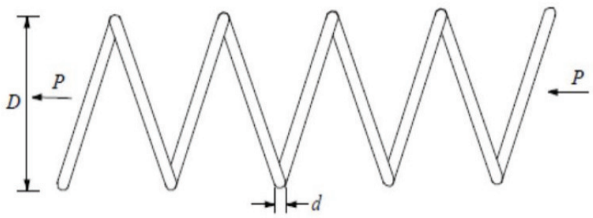

Figure 4.3: Tension/compression string problem

$$
\begin{aligned}
&\textit{minimize } f(x) = (x_3 + 2)x_2x_1^2 \\
&\textit{subject to} \\
&\qquad g_1(x) = 1 - \frac{x_2^3 x_3}{71785x_1^4} \leq 0 \\
&\qquad g_2(x) = \frac{4x_2^2 - x_1x_2}{12566(x_2x_1^3 - x_1^4)} + \frac{1}{5108x_1^2} - 1 \leq 0 \\
&\qquad g_3(x) = 1 - \frac{140.45x_1}{x_2^2x_3} \leq 0 \\
&\qquad g_4(x) = \frac{x_1 + x_2}{1.5} - 1 \leq 0 \\
&\qquad 0.05 \leq x_1 \leq 2 \\
&\qquad 0.25 \leq x_2 \leq 1.3 \\
&\qquad 2 \quad \leq x_3 \leq 15.
\end{aligned}
$$

Our solution for this problem is compared to the best solutions of different optimization techniques in Table (4.3), it can be seen that IPTRA solution is better than solutions found by other techniques listed in the table.

Table 4.3: Comparison of the best solutions for the tension/compression spring design problem.

Design variables	Belegundu (1982)[3]	Arora (1989)[1]	He and Wang (2007)[15]	Lobato and Steffen (2014)[18]	Algorithm IPTRA
x_1	0.050000	0.053396	0.051728	0.051744	0.051689061592032
x_2	0.315900	0.399180	0.357644	0.357754	0.356717752054532
x_3	14.25000	9.185400	11.244543	11.56132	11.288965032614222
f	0.012674	0.012730	0.012674	0.012789	0.012665232787753

Case 4. *Heat Exchanger Design [2]*
A fluid having a given flow rate W and specific heat C_p is heated from temperature T_0 to T_3 by passing three heat exchangers in series. In each heat exchanger (stage) the cold stream is heated by a hot fluid having the same flow rate W and specific heat C_p as the cold stream. The temperatures of the hot fluid entering the heat exchangers, t_{11}, t_{21} and t_{31} and the overall heat transfer coefficients U_1, U_2, U_3 of the exchangers are known constants. Optimal design involves minimizing the sum of the heat transfer areas of the three exchangers, $A_T = A_1 + A_2 + A_3$, as shown in figure (4.4).

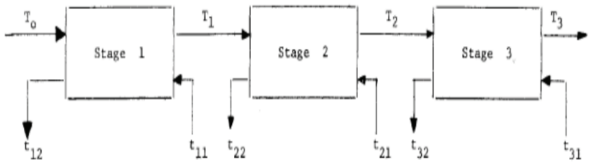

Figure 4.4: Three-stage heat exchanger system

The mathematical formulation of this problem can be described as follows [20]:

$$
\begin{aligned}
&\textit{minimize } A_T = A_1 + A_2 + A_3 \\
&\textit{subject to} \\
&\qquad g_1(x) = 0.0025(T_1 + t_{12}) - 1 \le 0 \\
&\qquad g_2(x) = 0.0025(-T_1 + T_2 + t_{22}) - 1 \le 0 \\
&\qquad g_3(x) = 0.01(-T_2 + t_{32}) - 1 \le 0 \\
&\qquad g_4(x) = 100A_1 - A_1 t_{12} + 833.33252T_1 - 83333.333 \le 0 \\
&\qquad g_5(x) = A_2 T_1 - A_2 t_{22} - 1250T_1 + 1250T_2 \le 0 \\
&\qquad g_6(x) = A_3 T_2 - A_3 t_{32} - 2500T_2 + 1250000 \le 0 \\
&\qquad 100 \le A_1 \le 10,000 \\
&\qquad 1000 \le A_i \le 10,000, i = 2, 3 \\
&\qquad 10 \le T_i \le 1000, i = 1, 2 \\
&\qquad 10 \le t_{i2} \le 1000, i = 1, 2, 3.
\end{aligned}
$$

From Table (4.4), it can be seen that the solution of the three-stage heat exchanger system problem found by IPTRA is almost the same by other competing technique (Avriel [2]).

Table 4.4: Comparison of the solutions for three-stage heat exchanger system problem.

Design variable	Avriel [2]	Algorithm IPTRA
A_1	567	579.306684436986
A_2	1357	1359.970668094694
A_3	5125	5109.970666971246
T_1	181	182.017699592019
T_2	295	295.601173303897
t_{12}	219	217.982300431680
t_{22}	286	286.416526324619
t_{32}	395	395.601173312338
A_T	7049	7049.248019502926

5 Conclusion remarks

In this paper, an interior point trust-region algorithm has been introduced to solve a general nonlinear programming problem. A Coleman-Li scaling matrix is used with an interior-point Newton method in the proposed algorithm. The reduced-Hessian technique is used to overcome the difficulty of having an infeasible trust-region subproblem and to simplify it.

A Matlab code of the proposed algorithm is implemented and tested against some existing codes. The results confirm the efficiency of the IPTRA algorithm. In addition to that, four case studies have been implemented to test the performance of the proposed algorithm where the results showed that the solution of the IPTRA algorithm is better than the best solutions of other techniques except for case 4 where the results were almost the same. We believe the proposed algorithm can be applied for solving real-world application problems efficiently.

References

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

[2] Avriel, M. and Williams, A. C., An extension of geometric programming with application in engineering optimization, Journal of Engineering Mathematics, 5, (1971), 187-194.

[3] Belegundu, A. D., A study of mathematical programming methods for structural optimization, Department of Civil and Environmental Engineering, University of Iowa, Iowa City, Iowa,1982.

[4] Byrd, R., Robust trust region methods for nonlinearly constrained optimization, A talk presented at the Second SIAM Conference on Optimization, Houston, (1987).

[5] Coello, C. A. C., Use of a self-adaptive penalty approach for engineering optimization problems, Computers in Industry, 41, (2), (2000), 113-127.

[6] Coleman, T. R., and Li, Y., An interior trust region approach for nonlinear minimization subject to bounds, SIAM Journal on Optimization, 6, (1996), 418-445.

[7] Das, I., 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, (1996).

[8] Deb, K., Optimal design of a welded beam via genetic algorithms, AIAA Journal, 29, (11), (1991), 2013-2015.

[9] Deb, K., GeneAS: A robust optimal design technique for mechanical component design, In: Dasgupta, D., Michalewicz, Z. (Eds.), Evolutionary Algorithms in Engineering Applications, Springer, Berlin,(1997), 497-514.

[10] El-Alem, M., A global convergence theory for the CelisDennisTapia trust-region algorithm for constrained optimization, SIAM Journal on Numerical Analysis, 28, (1) ,(1991), 266290.

[11] El-Alem, M. and Tapia, R., Numerical experience with a polyhedral-norm CDT trust-region algorithm, Journal of Optimization Theory and Applications, 85, (3), (1995), 575-591.

[12] El-Alem, M., El-Sayed, M., and El-Sobky, B., Local convergence of interior - point newton method for constrained optimization. Journal of Optimization Theory and Applica- tions, 120, (3), (2004), 487-502.

[13] El-Alem, M., El-Sobky, B., and Aziz, M. A., Extending goodman's method to general nonlinear programming, Bulletin Of Pure and Applied Mathematics, 3, (2) (2009), 135-141.

[14] Fletcher, R. A., sequential linear constraint programming algorithm for NLP. SIAM Journal on Optimization, 22, (3), (2012), 772-794.

[15] He, Q., and L., W., An effective co-evolutionary particle swarm optimization for constrained engineering design problems, Engineering Applications of Artificial Intelligence, 20, (2007), 89-99.

[16] Hock, W., and Schittkowski, K., Test examples for nonlinear programming codes, Vol. 187 of Lecture Notes in Economics and Mathematical Systems. Springer-Verlag, Berlin, (1981).

[17] Kannan, B. K., and Kramer, S. N., An augmented lagrange multiplier based method for mixed integer discrete continuous optimization and its applications to mechanical design, Journal of Mechanical Design. Transactions of ASME, 116, (1994), 318-320.

[18] Lobato, F. S., and Steffen Jr. V., Fish swarm optimization algorithm applied to engineering system design, Latin American Journal of Solids and Structures, 11, (2014), 143-156.

[19] Omojokun, E., Trust-region strategies for optimization with nonlinear equality and in- equality constraints, PhD thesis, Department of Computer Science, University of Colorado, Boulder, Colorado, (1989).

[20] Rao, S. S., Engineering Optimization. John Wiley and Sons, third edition, (1996).