Efficient Parameter Selection for Scaled Trust-Region Newton Algorithm in Solving Bound-constrained Nonlinear Systems

Hengameh Mirhajianmoghadam¹,  S. Mahmood Ghasemi²

¹. Department of Electrical Engineering, Ferdowsi University of Mashhad, Mashhad, Iran hengame.mirhajian@gmail.com
². Department of Mathematics, University of Houston, Houston, TX, USA Mahmood@math.uh.edu

Abstract

We investigate the problem of parameter selection for the scaled trust-region Newton (STRN) algorithm in solving bound-constrained nonlinear equations. Numerical experiments were performed on a large number of test problems to find the best value range of parameters that give the least algorithm iterations and function evaluations. Our experiments demonstrate that, in general, there is no best parameter to be chosen and each specific value shows an efficient performance on some problems and a weak performance on other ones. In this research, we report the performance of STRN for various choices of parameters and then suggest the most effective one.

Key words: Trust-Region Methods, Nonlinear System of Equations, Bound-Constrained Optimizations, Parameter Selection

1 Introduction

Optimization plays a key role in contemporary science, including engineering, statistics, computer science, physics, and applied mathematics. If the physics of a phenomenon is properly and comprehensively captured in the associated mathematical model, a huge number of real-world problems translate into solving mathematical problems; especially in the form of optimization [15,12,20,13]. Most practical problems may include nonlinear constraints, but constrained optimization remarkably relies on the techniques used in unconstrained optimization. Consequently, the more fruitful unconstrained optimization algorithms are, the more efficient methods can be proposed for the constrained optimization. Therefore, it is of extreme importance to investigate the properties of the main algorithms used in unconstrained
optimization. Here, we examine the performance of a vital algorithm in the trust-region paradigm. We study the efficient parameter selection for the scaled trust-region Newton (STRN) algorithm in solving bound-constrained nonlinear systems. We demonstrate that its performance notably relies on the choices of its hyperparameters and further suggest how to choose the most effective parameters.

We consider the STRN algorithm proposed by Bellavia et al. [2] and its Matlab solver STRSCNE [3]. This numerical algorithm solves the bound-constrained nonlinear system of equations using an affine scaling trust-region method. The problem is

$$F(x) = 0; \quad x \in \Omega,$$

where $F(x) = (F_1(x), \ldots, F_n(x))^T$ and $\Omega = \{x \in \mathbb{R}^n | l \leq x \leq u\}$. The vectors $l \in (\mathbb{R} \cup -\infty)^n$ and $u \in (\mathbb{R} \cup \infty)^n$ are lower and upper bounds, respectively. The function $F$ is continuously differentiable in an open set $X \subset \mathbb{R}^n$ containing the $n$-dimensional box $\Omega$. These kind of systems appear in chemical process modeling and in steady-state simulation [21,4].

An approach to solve (1.1) is a bound-constrained nonlinear least square problem:

$$\min_{x \in \Omega} f(x) := \frac{1}{2} \|F(x)\|_2^2$$

Nonlinear least square problems have been studied in the literature [18,10,11,23]. Bellavia et al. [2] generalized the trust-region strategy for unconstrained systems of nonlinear equations to bound-constrained systems and proposed the STRN; a reliable method for tackling (1.1). This method generates feasible iterates with locally and globally fast convergence properties. A large set of problems was used to test the efficiency of the STRN. In [2], a comparison with the ASTN [10] and IGNT [11] and in [6,7] a comparison between the STRN and NMAadapt [8] has been performed and superiority of the method has been proved. The method has been widely applied in engineering fields [5,14,9,17,19].

In this paper, we run the iterative algorithm STRN using its implementation in Matlab solver called STRSCNE (Scaled Trust-Region Solver for Constrained Nonlinear Equations) on various problems and extract the most useful parameter values. The motivation behind this fine tuning is that the fast-increasing availability of massive data sets has boosted up the use of sophisticated optimization algorithms [1,16] with fast convergence.

The remaining of the paper is as follows: in section 2, we explain the STRN algorithm. Section 3 presents the parameter selecting experiments and the achieved numerical results. Section 4 is the conclusion together with the related figures and tables.
2 The STRN Algorithm

Let \( x_k \in \text{int}(\Omega) \) be the current iteration. Next iteration is \( x_{k+1} = x_k + p_k \) where \( p_k \) is computed by solving the following elliptical trust-region subproblem

\[
\min_p \, m_k(p) \quad \text{subject to} \quad \|D_k p\| \leq \Delta_k.
\]

(2.1)

Here, \( \Delta_k \) is the trust-region size, \( D_k = D(x_k) \) is the diagonal scaling matrix such that:

\[
D(x) = \text{diag}(|v_1(x)|^{-\frac{1}{2}}, |v_2(x)|^{-\frac{1}{2}}, \ldots, |v_n(x)|^{-\frac{1}{2}}),
\]

and \( v(x) \) denotes the vector function given by:

\[
\begin{align*}
v_i(x) &= x_i - u_i \quad \text{if} \quad (\nabla f(x))_i < 0, \quad \text{and} \quad u_i < \infty; \\
v_i(x) &= x_i - l_i \quad \text{if} \quad (\nabla f(x))_i \geq 0, \quad \text{and} \quad u_i < -\infty; \\
v_i(x) &= -1 \quad \text{if} \quad (\nabla f(x))_i < 0, \quad \text{and} \quad u_i = \infty; \\
v_i(x) &= 1 \quad \text{if} \quad (\nabla f(x))_i \geq 0, \quad \text{and} \quad u_i = -\infty,
\end{align*}
\]

and \( v = (v_1, \ldots, v_n) \).

The quadratic model for \( f \) in (2.1) is as follows:

\[
m_k(p) = \frac{1}{2} \|F_k' p + F_k\|^2 = \frac{1}{2} \|F_k\|^2 + F_k^T F_k' p + \frac{1}{2} p^T F_k' F_k^T F_k' p = f_k + \nabla f_k^T p + \frac{1}{2} p^T F_k'^T F_k' p.
\]

The scaled steepest descent direction \( d_k \) is given by \( d_k = -D_k^{-2} \nabla f_k \). The trial step \( p_k \) can be calculated by the following procedure (see [2,3] for detail).

**Procedure to calculate the trial step:**

Let \( \nabla f_k, D_k, \) and \( \Delta_k \) be given.

1. Calculate the Newton step \( p_k^N \) by solving \( F_k' p_k^N = -F_k \)

2. If \( \|D_k p_k^N\| \leq \Delta_k \) then set \( p_k = p_k^N \) and stop.

3. Compute \( \bar{p}_k^u = -\left(\frac{D_k^{-1} \nabla f_k}{\|F_k' D_k^{-2} \nabla f_k\|^2}\right) D_k^{-1} \nabla f_k \)

4. If \( \|\bar{p}_k^u\| \geq \Delta_k \), then set \( \bar{p}_k = \Delta_k D_k^{-1} \nabla f_k / \|D_k^{-1} \nabla f_k\| \)
else set \( \bar{p}_k^N = D p_k^N \) and compute \( \mu \) solving \( \|\bar{p}_k^u + (1 - \mu)(\bar{p}_k^N - \bar{p}_k)\|^2 = \Delta_k^2 \) and set \( \bar{p}_k = \bar{p}_k^u + (1 - \mu)(\bar{p}_k^N - \bar{p}_k) \)

5. Let \( p_k = D_k^{-1} \bar{p}_k \).
Now to ensure that the next iterate stays within $\Omega$, let's calculate the step size $\lambda(p_k)$ along $p_k$ to the boundary

$$\lambda(p_k) = \begin{cases} 
\infty & \text{if } \Omega = \mathbb{R}^n \\
\min_{1 \leq i \leq n} \Lambda_i & \text{if } \Omega \subset \mathbb{R}^n,
\end{cases}$$

where

$$\Lambda_i = \begin{cases} 
\max \left\{ \frac{i-x_k}{p_{ki}}, \frac{u_i-x_k}{p_{ki}} \right\} & \text{if } p_{ki} \neq 0 \\
\infty & \text{if } p_{ki} = 0.
\end{cases}$$

if $\lambda(p_k) > 1$ then $x_k + p_k$ is within $\Omega$, if $\lambda(p_k) \leq 1$ then a step back along $p_k$ is necessary. Parameter $\theta$ controls the amount of truncation:

$$x_{k+1} = x_k + \alpha(p_k)$$

$$\alpha(p_k) = \begin{cases} 
p_k & \text{if } \lambda(p_k) > 1 \\
\max\{\theta, 1 - \|p_k\|\} \lambda(p_k)p_k & \text{if } \lambda(p_k) \leq 1.
\end{cases}$$

$\theta \in (0, 1)$ is a fixed constant and it is one of the parameters that we will find an optimal range for.

In order to warranty sufficient reduction, we have to consider the Cauchy point $p^c_k$, the minimizer of $m_k$ along the scaled steepest descent direction $d_k = -D_k^{-2} \nabla f_k$, at the other hand new iterate should lie within the trust-region so the steepest descent has to satisfy the trust-region bound [18]

$$p^c_k = \tau_k d_k = -\tau_k D_k^{-2} \nabla f_k,$$

where

$$\tau_k = \arg\min_{r > 0} \left\{ m_k(\tau d_k) : \|D_k d_k\| \leq \Delta_k \right\} = \min \left\{ \frac{\|D_k^{-1} \nabla f_k\|^2}{\|F_k D_k^{-2} \nabla f_k\|^2, \|D_k^{-1} \nabla f_k\|} \right\}.$$ 

Then we test if the step $\alpha(p_k)$ satisfies the following condition:

$$\rho_k^c(p_k) = \frac{m_k(0) - m_k(\alpha(p_k))}{m_k(0) - m_k(\alpha(p^c_k))} \geq \beta_1$$

where $\beta_1 \in (0, 1)$ is constant. If this condition holds, we discard $p_k$ and set $p_k = p^c_k$. The agreement between the model $m_k$ and the merit function $f$ can be achieved by testing the following condition:

$$\rho_k^f(p_k) = \frac{f(x_k) - f(x_k + \alpha(p_k))}{m_k(0) - m_k(\alpha(p_k))} \geq \beta_2$$
where $\beta_2 \in (0, 1]$. If this condition holds then $x_k + \alpha(p_k)$ is the next iterate. If not then we have to decrease the trust-region size by

$$\Delta_k = \min\{\alpha_1 \Delta_k, \alpha_2 \|D_k \alpha(p_k)\|\}$$

for $0 < \alpha_1 \leq \alpha_2 < 1$ and recalculate new step.

Finally, in order to accelerate the convergence rate, we should take big steps by increasing the trust-region radius wisely. If the agreement between the function and the model is strong enough then we shouldn’t miss the opportunity to take a better improvement. The condition below satisfies this:

$$\rho_f^c(p_k) = \frac{f(x_k) - f(x_k + \alpha(p_k))}{m_k(0) - m_k(\alpha(p_k))} \geq \beta_3,$$

$\beta_3 \in (0, 1]$ is a constant such that $\beta_2 < \beta_3 < 1$.

If this condition holds then

$$\Delta_{k+1} = \max\{\Delta_k, \gamma \|D_k \alpha(p_k)\|\}, \quad \gamma > 1$$

otherwise the trust-region radius remains same.

The STRSCNE algorithm is as follows:

Algorithm, The Scaled Trust-Region Solver [3]
- Initialization:
  Given $x_0 \in int(\Omega)$, $\Delta_0 > 0$, $\theta \in (0, 1)$, $0 < \alpha_1 \leq \alpha_2 < 1$, $\beta_1 \in (0, 1]$, $0 < \beta_2 < \beta_3 < 1$.
- For $k = 0, 1, \ldots$ do:
  1. Compute $F_k$.
  2. Check for convergence.
  3. Compute the matrix $D_k$ by using (1.5).
  4. Compute the matrix $F'_k$.
  5. Compute $p^N_k$ by solving the linear system $F'_k p^N_k = -F_k$.

6. Repeat
6.1. Compute an approximate solution $p_k$ of (2.1) by using Procedure to calculate the trial step.
6.2. Compute $\tau_k$ by and the Cauchy point $p^c_k$.
6.3. Compute $\alpha(p_k)$ and $\alpha(p^c_k)$.
6.4. Compute $\rho_f^c(p_k)$.
6.5. If $\rho_f^c(p_k) < \beta_1$ then set $p_k = p^c_k$.
6.6. Set $\Delta^*_k = \Delta_k$ and decrease $\Delta_k$
6.7. Compute $\rho_f^c(p_k)$.

Until $\rho_f^c(p_k) \geq \beta_2$
7. Set $x_{k+1} = x_k + \alpha(p_k), \Delta_k = \Delta^*_k$
8. If $\rho_f^c \geq \beta_3$ then
   set $\Delta_{k+1} = \max\{\delta_k, \gamma \|D_k \alpha(p_k)\|\}$
else set $\Delta_{k+1} = \Delta_k$
The algorithm is convergent when we have: $\|F_{k+1}\| \leq 10^{-8}$.

Failure happens if
1) A maximum number of iterations are performed.
2) A maximum number of F-evaluations are performed.
3) The trust-region size is reduced below $10^{-8}$
4) The relative change in the function value satisfies

$$\|F_{K+1} - F_k\| \leq 10^{-14} \|F_k\|$$

5) The norm of the scaled gradient of the merit function becomes small:

$$\left\| D_k^{-1} \nabla f_k \right\| < 10^{-14}$$

6) The scaling matrix $D_k$ cannot be computed.

### 3 Numerical Experiments of Parameter Tuning

In our experiments, we used Matlab 2018b. For the initialization, we set $\Delta_0 = 1, \theta = 0.99995, \alpha_1 = 0.25, \alpha_2 = 0.5, \beta_1 = 0.1, \beta_2 = 0.25, \beta_3 = 0.75, \gamma = 2$. The tested problems come from NLE library [22] accessible through: www.polymath-software.com/library. The number of the iterations and the number of the function evaluations follow the same behavior, lots of the problems are indifferent to changes in the parameters but some of them are influenced by different values of the parameters. In this section, we run the STRN for different values of parameters in order to find the best values with least iteration and function evaluation. In each section we fix all the parameters as indicated above and impose the variation only on the specific one.

The fact is that there is no "Best" parameter. A parameter works pretty good for some problems and pretty bad for other problems, so we have to choose the one with overall better performance. Tables 1-6 show the iterations for different starting points.

#### 3.1 $\alpha_1$ - controls the size reduction of the trust-region

Among the 45 studied problems, 32 problems are insensitive for different values of $\alpha_1$ while 13 problems are sensitive for at least one of the initial points. After consecutive runs of the algorithm, we narrowed them down to some candidate values for each parameter. Table 1 shows these values. There is no best parameter for $\alpha_1$. For example, $\alpha_1 = 0.4$ is the best option for solving Foureq1 starting from the third
while for Threeq6 starting from the fourth $x_0$, or seveneq2b starting from Third $x_0$, $\alpha_1 = 0.3$ is much better. Generally speaking, a value between (0.4,0.5) is the best choice (Fig 1). It seems that a harsh and severe shrinking of the trust-region is not always the best approach, we prefer to cut the trust-region size at most in half.

![Figure 1.](image)

Fine tuning for $\alpha_1$-Problem Nineq1, Fourth

3.2 $\alpha_2$- controls the size reduction of the trust-region

Among the 45 studied problems, 36 problems are insensitive to different values of $\alpha_2$ while 9 problems are sensitive for at least one of the initial points. Again, there is no best choice and depending on the problem the performance changes. For example, for Threeq5 (Fig 2) 0.24 and 0.45 are best choices while for other problems they are not suitable. by looking at Table 2 the overall performance of $\alpha_2 = 0.45, 0.6$ are slightly better. $\alpha_2 = 0.45$ is the best choice for problem Seveneq2b, it converges to the answer in 130 iterations while other selections of the parameter need a large iteration number until getting convergent. $\alpha_2 = 0.4$ and $\alpha_2 = 0.7$ also show pretty good performance.

3.3 $\beta_1$-used for accuracy requirements

Among the 45 studied problems, 43 problems are insensitive for different values of $\beta_1$ while 2 problems are sensitive to at least one of the initial points. By looking at Table 3, as $\beta_1$ increases (The reduction in the model when we use $p_k$ is close to the reduction in the model when we use $p_k$) the algorithm becomes faster. $\beta_1 = 0.2$
shows good performance. The problems are mostly insensitive to the different values of $\beta_1$.

### 3.4 $\beta_2$ - used to ensure agreement between the model and the objective

Among the 45 studied problems, 36 problems are insensitive for different values of $\beta_2$ while 9 problems are sensitive for at least one of the initial points. From Table 4 clearly $\beta_2 = 0.15$ is the best choice. As $\beta_2$ increases the agreement between the model and the function becomes harder and the method becomes slower. We have this situation in problem Nineeq1 (Fig 3):
3.5 $\beta_3$-controls the updating of the trust-region size

Among the 45 studied problems, 31 problems are insensitive to different values of $\beta_3$ while 14 problems are sensitive for at least one of the initial points. $\beta_3 = 0.75, 0.6$ are the best choices. It means that we shouldn’t wait for a great agreement between the model and the function and should take the opportunity to increase the trust-region size.

3.6 $\theta$-used to ensure strictly feasible iterates

Among the 45 studied problems, 28 problems are insensitive to different values of $\theta$ while 17 problems are sensitive for at least one of the initial points. Looking at Table 6, $\theta = 0.9$ gives the best results. $\theta = 0.95$ is also a good choice, it works as good as the traditional $\theta = 0.99995$, so a big truncation of $p_k$ shows as good performance as small truncation. $\theta = 0.7$ is working surprisingly good. It solves the problem Threeq4a in the least possible iterations.

3.7 $\gamma$-controls the size enlargement of the trust-region

Among the 45 studied problems, 21 problems are insensitive to different values of $\gamma$ while 24 problems are sensitive for at least one of the initial points. Looking at Fig 4 in most of the cases $\gamma = 8, 10$ show better performance comparing to $\gamma = 2$. It seems that taking big steps and going through large trust-region sizes is risky but it worth to take this risk since in most of the situations it shows faster convergence.

4 Conclusion

The problems Twoeq5a, Twoeq5b, Twoeq7, Threeq5, and 11eq1 are nearly sensitive to changes in the parameters. The sensitive problems that can be affected by different values of the parameters are: Twoeq 6, Threeq1, Threeq4a, Threeq4b, Threeq6, Threeeq6, Foureq1, Sixeq4b, Seveneq2b, Nineq1 and super sensitive problems that can be affected by all of the 7 parameters are Seveneq2b, and Nineq1.

The results show that taking big steps by increasing the trust-region size is risky, but it causes faster convergence and it worth to take the risk. Also, we prefer not to cut the trust-region too much, we prefer to stay in the Cauchy direction as much as possible and take easier criteria to check the agreement between the model and the merit function.

The interesting point is that for some of the problems (Threeq6, Sixeq4b, Seve- neq2b, and 11eq1) only specific values of $\beta_3$ and $\theta$ work well. We concentrate on $\theta$
since it plays a more important role. The fact that the method fails for some values of the \( \theta \) motivates us to try different values of this parameter before giving up. This can be done by introducing a sequence of parameters as initial values of \( \theta \) and not a constant one. That means in STRSCNE algorithm the initialization step should be modified as follows:

STRSCNE with a variable \( \theta \)

- Initialization:
  Given \( x_0 \in int(\Omega), \Delta_0 > 0, \theta_i = \{0.7 + 0.025i\}_{i=0}^{12}, 0 < \alpha_1 \leq \alpha_2 < 1, \beta_1 \in (0, 1], 0 < \beta_2 < \beta_3 < 1 \), Let \( i = 0 \) and \( \theta = \theta_0 \).

Steps 1-8

9. If \( \|F_{k+1}\| > 10^{-8} \) then repeat the algorithm. If \( \|F_{k+1}\| \leq 10^{-8} \):

9.1 If Ierr=0 then
STOP and report the solution.
else set \( i = i + 1, \theta = \theta_i \) and go to step 1 unless \( i = 12 \).

It is either convergent to the solution or repeats the algorithm for different values of \( \theta \) before reporting a defeat.

This can be a useful approach if the goal is solving a problem several times in a restricted amount of time. For example Problem Sixeq4b starting from third \( x_0 \) and \( \theta = 0.6 \) is convergent to the solution in 9 iterations while for \( \theta = 0.99995 \) the number of iterations is 350. For the first choice of \( \theta \) it takes 1.24 seconds and for the second choice, it takes 41.72 seconds. This happens in several other problems too.
Table 1
The Algorithm's Iterations for the selected values of $\alpha_1$

| First $x_0$ | Problem | $\alpha_1 = 0.2$ | $\alpha_1 = 0.3$ | $\alpha_1 = 0.4$ | $\alpha_1 = 0.5$ | $\alpha_1 = 0.6$ | $\alpha_1 = 0.7$ |
|-------------|---------|------------------|------------------|------------------|------------------|------------------|------------------|
| Twoeq7      |         | 9                | 9                | 7                | 7                | 7                | 7                |
| Second $x_0$|         |                  |                  |                  |                  |                  |                  |
| Threeq1     |         | 42               | 32               | 32               | 32               | 32               | 32               |
| Threeq4b    |         | 7                | 7                | 7                | 7                | 6                |                  |
| foureq1     |         | 10               | 10               | 10               | 10               | 16               | 15               |
| Third $x_0$ |         |                  |                  |                  |                  |                  |                  |
| twoeq4b     |         | 7                | 8                | 7                | 7                | 7                | 7                |
| Twoeq6      |         | 10               | 9                | 12               | 9                | 9                | 9                |
| Threeq1     |         | 31               | 35               | 30               | 32               | 32               | 32               |
| Threeq4a    |         | 62               | 54               | 118              | 70               | 62               | 62               |
| Threeq4b    |         | 10               | 7                | 7                | 7                | 7                | 7                |
| Foureq1     |         | 13               | 13               | 11               | 12               | 12               | 12               |
| Sixeq4b     |         | 326              | 301              | 299              | 334              | 334              | 334              |
| Seveneq2b   |         | 117              | 35               | 112              | 84               | 84               | 84               |
| Seveneq3a   |         | 8                | 9                | 7                | 7                | 7                | 7                |
| Fourth $x_0$|         |                  |                  |                  |                  |                  |                  |
| Twoeq6      |         | 9                | 7                | 9                | 7                | 7                | 7                |
| Twoeq9      |         | 6                | 7                | 6                | 8                | 8                | 8                |
| Threeq4a    |         | 51               | 52               | 53               | 54               | 54               | 54               |
| Threeq6     |         | 100              | 76               | 225              | 192              | 192              | 192              |
| Nineq1      |         | 20               | 17               | 18               | 15               | 15               | 15               |

Table 2
The Algorithm's Iterations for the selected values of $\alpha_2$

| Second $x_0$ | Problem | $\alpha_2 = 0.3$ | $\alpha_2 = 0.4$ | $\alpha_2 = 0.45$ | $\alpha_2 = 0.5$ | $\alpha_2 = 0.6$ | $\alpha_2 = 0.7$ |
|--------------|---------|------------------|------------------|-------------------|------------------|------------------|------------------|
| Twoeq6       |         | 14               | 16               | 16                | 16               | 13               | 17               |
| Threeq1      |         | 45               | 32               | 32                | 32               | 32               | 32               |
| Threeq5      |         | 22               | 18               | 16                | 20               | 19               | 21               |
| foureq1      |         | 13               | 13               | 10                | 16               | 13               | 12               |
| Third $x_0$  |         |                  |                  |                  |                  |                  |                  |
| Twoeq6       |         | 12               | 13               | 14                | 13               | 11               | 10               |
| Threeq4a     |         | 52               | 65               | 65                | 67               | 67               | 67               |
| Threeq4b     |         | 9                | 7                | 7                 | 7                | 7                | 7                |
| Sixeq4b      |         | 321              | 493              | 495               | 496              | 480              | 497              |
| Seveneq2b    |         | 277              | 252              | 130               | 199              | 165              | 157              |
| Fourth $x_0$ |         |                  |                  |                  |                  |                  |                  |
| Twoeq6       |         | 6                | 12               | 6                 | 6                | 6                | 6                |
| Threeq4a     |         | 51               | 51               | 52                | 59               | 53               | 54               |
| Nineq1       |         | 22               | 18               | 20                | 20               | 19               | 20               |

Table 3
The Algorithm's Iterations for the selected values of $\beta_1$

| First $x_0$ | Problem | $\beta_1 = 0.05$ | $\beta_1 = 0.1$ | $\beta_1 = 0.15$ | $\beta_1 = 0.2$ | $\beta_1 = 0.25$ |
|-------------|---------|------------------|------------------|------------------|------------------|------------------|
| Nineq1      |         | 200              | 200              | 21               | 21               | 21               |
| Third $x_0$ |         |                  |                  |                  |                  |                  |
| Seveneq2b   |         | 199              | 199              | 199              | 179              | 179              |
Table 4
The Algorithm’s Iterations for the selected values of $\beta_3$

|       | $\beta_3 = 0.6$ | $\beta_3 = 0.7$ | $\beta_3 = 0.75$ | $\beta_3 = 0.8$ | $\beta_3 = 0.85$ |
|-------|-----------------|-----------------|-------------------|-----------------|-----------------|
| Twoeq7| 9               | 10              | 8                 | 8               | 8               |
| Threeq1 | 5              | 5               | 5                 | 6               | 6               |
| Threeq6| 8               | 8               | 8                 | 8               | 25              |
| Second $x_0$ |   |               |                   |                  |                  |
| Threeq1 | 38              | 40              | 32                | 45              | 58              |
| Threeq5 | 20              | 20              | 20                | 42              | 93              |
| Threeq6 | 116             | 303             | 305               | 307             | -               |
| Threeq7 | 18              | 13              | 13                | 13              | 13              |
| Seveneq2b | 17             | 14              | 14                | 14              | 14              |
| Third $x_0$ |   |               |                   |                  |                  |
| Twoeq5b | 8               | 8               | 7                 | 7               | 7               |
| Twoeq6 | 13              | 13              | 13                | 13              | 10              |
| Threeq1 | 38              | 31              | 24                | 24              | 39              |
| Threeq2 | 39              | 39              | 39                | 44              | 44              |
| Threeq4a | 44             | 53              | 67                | 64              | 75              |
| Fourreq1 | 11             | 13              | 13                | 13              | 20              |
| Sixeq4b | 380             | 361             | 350               | 342             | 337             |
| Seveneq2b | 207            | 202             | 199               | 206             | 178             |
| Fourth $x_0$ |   |               |                   |                  |                  |
| Threeq4a | 33              | 54              | 59                | 60              | 64              |
| Threeq6 | 313             | 96              | 95                | 93              | -               |
| Nineq1 | 19              | 19              | 20                | 20              | 20              |
Table 5
The Algorithm’s Iterations for the selected values of $\theta$

| First $x_0$ | $\theta = 0.6$ | $\theta = 0.7$ | $\theta = 0.8$ | $\theta = 0.9$ | $\theta = 0.95$ | $\theta = 0.99995$ |
|-------------|-----------------|-----------------|-----------------|-----------------|-----------------|------------------|
| Nineq1      | 14              | 12              | 14              | 8               | 51              | 200              |
| Teneq1a     | 15              | 14              | 14              | 14              | 14              | 14               |

| Second $x_0$ |
|--------------|
| Twoeq5       | 16              | 17              | 15              | 14              | 14              | 16               |
| Threeq1      | 42              | 32              | 40              | 32              | 32              | 32               |
| Threeq5      | 15              | 15              | 15              | 20              | 20              |                  |
| Threeq6      | 97              | 122             | 134             | 102             | 319             | 305              |
| Threeq7      | 11              | 11              | 10              | 14              | 14              | 13               |
| Foureq1      | 13              | 11              | 13              | 10              | 13              | 16               |
| Sixeq4b      | -               | -               | 10              | 10              | 9               | 7                |
| Teneq1b      | 80              | 40              | 97              | 90              | 79              | 14               |
| Teneq2b      | 13              | 12              | 12              | 11              | 11              | 11               |

| Third $x_0$ |
|-------------|
| Twoeq6      | 18              | 15              | 15              | 16              | 12              | 13               |
| Threeq7     | 8               | 8               | 8               | 7               | 7               | 7                |
| Threeq4a    | 70              | 54              | 57              | 58              | 67              | 67               |
| Threeq4b    | 7               | 6               | 7               | 6               | 6               | 7                |
| Foureq1     | 12              | 12              | 13              | 13              | 13              | 13               |
| Sixeq4b     | 9               | 9               | 170             | 260             | 253             | 350              |
| Seveneq2b   | 40              | 153             | -               | 21              | 41              | 199              |
| Teneq2a     | 11              | 11              | 10              | 9               | 9               | 9                |
| Elevenq1    | -               | -               | -               | -               | 22              | 19               |

| Fourth $x_0$ |
|--------------|
| Threeq4a    | 47              | 62              | 48              | 51              | 52              | 59               |
| Sixeq4b     | 9               | 8               | 8               | 7               | 7               | 7                |
| Nineq1      | 24              | 23              | 22              | 22              | 19              | 20               |

Figure 4. Different $\gamma$ s performance on sensitive problems
References

[1] Azencott, R., Muravina, V., Hekmati, R., Zhang, W., & Paldino, M. (2019). Automatic clustering in large sets of time series. In Contributions to Partial Differential Equations and Applications (pp. 65-75). Springer, Cham.

[2] Bellavia, S., Macconi, M., & Morini, B. (2003). An affine scaling trust-region approach to bound-constrained nonlinear systems. Applied Numerical Mathematics, 44(3), 257-280.

[3] Bellavia, S., Macconi, M., & Morini, B. (2004). STRSCNE: A scaled trust-region solver for constrained nonlinear equations. Computational Optimization and Applications, 28(1), 31-50.

[4] Bullard, L. G., & Biegler, L. T. (1991). Iterative linear programming strategies for constrained simulation. Computers & chemical engineering, 15(4), 239-254.

[5] Ferkl, L., & Meinsma, G. (2007). Finding optimal ventilation control for highway tunnels. Tunnelling and underground space technology, 22(2), 222-229.

[6] Hekmati, R. (2016). On efficiency of non-monotone adaptive trust region and scaled trust region methods in solving nonlinear systems of equations. Biquarterly Research Journal of Control and Optimization in applied Mathematics, 1(1), 31-40.

[7] Hekmati, R., & Mirhajianmoghadam, H. (2018). Nested performance profiles for benchmarking software. arXiv preprint arXiv:1809.06270.

[8] Hongwei, L. (2008). A Non-monotone Adaptive Trust Region Algorithm for Nonlinear Equations for Recurrent Event Data [J]. Acta Mathematicae Applicatae Sinica, 6.

[9] Hosseini, M., Islam, R., Kulkarni, A., & Mohsenin, T. (2017, April). A scalable fpga-based accelerator for high-throughput mcmc algorithms. In 2017 IEEE 25th Annual International Symposium on Field-Programmable Custom Computing Machines (FCCM) (pp. 201-201). IEEE.

[10] Kanzow, C. (2001). An active set-type Newton method for constrained nonlinear systems. In Complementarity: applications, algorithms and extensions (pp. 179-200). Springer, Boston, MA.

[11] Kozakevich, D. N., Martinez, J. M., & Santos, S. A. (1997). Solving nonlinear systems of equations with simple constraints. Computational & Applied Mathematics.

[12] Layegh, M., Ghodsí, F. E., & Hadipour, H. (2018). Improving the electrochemical response of nanostructured MoO3 electrodes by Co doping: Synthesis and characterization. Journal of Physics and Chemistry of Solids, 121, 375-385.

[13] Layegh, M., Ghodsí, F. E., & Hadipour, H. (2020). Experimental and theoretical study of Fe doping as a modifying factor in electrochemical behavior of mixed-phase molybdenum oxide thin films. Applied Physics A, 126(1), 14.

[14] Manzacca, G., Cincotti, G., & Hingerl, K. (2007). Ultrafast switching by controlling Rabi splitting. Applied Physics Letters, 91(23), 231920.
[15] Mousavi, S., Taghiabadi, M. M. R., & Ayanzadeh, R. (2019). A survey on compressive sensing: Classical results and recent advancements. arXiv preprint arXiv:1908.01014.

[16] Najarian, M., & Lim, G. J. (2020). Optimizing infrastructure resilience under budgetary constraint. Reliability Engineering & System Safety, 198, 106801.

[17] Najarian, M., Sarmast, Z., Ghasemi, S. M., & Sarmadi, S. (2018). Evolutionary Vertical Size Reduction: A novel Approach for Big Data Computing. rm, 55, 7.

[18] Nocedal, J., & Wright, S. (2006). Numerical optimization. Springer Science & Business Media.

[19] Pirhooshyaran, M., & Snyder, L. V. (2017, August). Optimization of inventory and distribution for hip and knee joint replacements via multistage stochastic programming. In Modeling and optimization: Theory and applications (pp. 139-155). Springer, Cham.

[20] Schweidtmann, A. M., Clayton, A. D., Holmes, N., Bradford, E., Bourne, R. A., & Lapkin, A. A. (2018). Machine learning meets continuous flow chemistry: Automated optimization towards the Pareto front of multiple objectives. Chemical Engineering Journal, 352, 277-282.

[21] Shacham, M. (1986). Numerical solution of constrained non-linear algebraic equations. International journal for numerical methods in engineering, 23(8), 1455-1481.

[22] Shacham, M., Brauner, N., & Cutlip, M. B. (2002). A web-based library for testing performance of numerical software for solving nonlinear algebraic equations. Computers & Chemical Engineering, 26(4-5), 547-554.

[23] Ulbrich, M. (2001). Nonmonotone trust-region methods for bound-constrained semismooth equations with applications to nonlinear mixed complementarity problems. SIAM Journal on Optimization, 11(4), 889-917.