A new limited memory method for unconstrained nonlinear least squares

Morteza Kimiaei · Arnold Neumaier

Abstract

This paper suggests a new limited memory trust region algorithm for large unconstrained black box least squares problems, called LMLS. Main features of LMLS are a new non-monotone technique, a new adaptive radius strategy, a new Broyden-like algorithm based on the previous good points, and a heuristic estimation for the Jacobian matrix in a subspace with random basis indices. Our numerical results show that LMLS is robust and efficient, especially in comparison with solvers using traditional limited memory and standard quasi-Newton approximations.

Keywords Black box least squares · Limited memory method · Trust region method · Non-monotone technique

1 Introduction

In this paper, we consider the unconstrained nonlinear least squares problem

$$\min f(x) := \frac{1}{2} \| E(x) \|_2^2$$

s.t. \( x \in \mathbb{R}^n \),

(1)

with high-dimensional \( x \in \mathbb{R}^n \) and continuously differentiable \( E : \mathbb{R}^n \rightarrow \mathbb{R}^r (r \geq n) \), possibly expensive. However, we assume that no derivative information is available.

1.1 Related work

In recent years, there has been a huge amount of literature on least squares and its applications. Here we just list a useful book and paper:

- Ortega and Rheinboldt (2000) introduced an excellent book, both covering algorithms and their analysis.
- An excellent paper, both covering Levenberg–Marquardt algorithms, quasi-Newton algorithms, and trust region algorithms and their local analysis without nonsingularity assumption, has been introduced by Yuan (2011).

Derivative free unconstrained nonlinear black box least squares solvers can be classified in two ways according to how the Jacobian matrix is estimated, and according to whether they are based on line search or on trust region:

- Quasi-Newton approximation. Sorber et al. (2012) introduced MINLBFGS (a limited memory BFGS algorithm) and MINLBFGSDL (a trust region algorithm using a dogleg algorithm and limited memory BFGS approximation).
- Finite difference approximation. There are many trust region methods using the finite difference method for the Jacobian matrix estimation such as CoDoSol and STRSCNE by Bellavia et al. (2004, 2012), NMPNTR by Kimiaei (2016), NATRN and NATRLS by Amini et al. (2016); Amini et al. (2016), LSQNONLIN from the MATLAB Toolbox, NLSQERR (an adaptive trust region strategy) by Deuflhard (2011), and DOGLEG by Nielsen (2012). They are suitable for small- and medium-scale problems. Line search methods using the finite difference approximation are NLEQ (a damped affine invariant
Newton method) by Nowak and Weismann (1990) and MINFNC(G (a family of nonlinear conjugate gradient methods) by Sorber et al. (2012).

**FMINUNC** by MATLAB Optimization Toolbox is an efficient solver for small- and medium-scale problems. It uses the finite difference method to estimate the gradient vector and the standard quasi-Newton method to estimate the Hessian matrix. In fact, FMINUNC disregards the least squares structure and only has access to function values. Nevertheless, it will be shown that FMINUNC is more efficient than LSQNONLIN using the least squares structure.

To solve the least squares problem (1), trust region methods use linear approximations of the residual vectors to make surrogate quadratic models whose accuracy are increased by restricting their feasible points. These methods use a computational measure to identify whether an agreement between an actual reduction of the objective function and a predicated reduction of surrogate quadratic model function is good or not. If this agreement is good, the iteration is said successful and the trust region radius is expanded; otherwise, the iteration is said unsuccessful and the trust region radius is reduced, for more details see (Conn et al. 2000; Nocedal and Wright 1999).

The efficiency of trust region methods depends on how the trust region radius is updated (see, e.g., Ahookhosh et al. 2013; Amini et al. 2016; Amini et al. (2016); Esmaeili and Kimiae (2014b, a, 2015); Fan (2006); Fan and Pan (2009, 2010); Kimiae (2017); Yu and Pu (2008)) and whether non-monotone techniques are applied (see, e.g., Ahookhosh and Amini 2011; Ahookhosh et al. 2015, 2013; Amini et al. 2016; Amini et al. (2016); Deng et al. (1993); Grippo et al. (1986); Grippo and Scandi (2007); Kimiae (2016, 2017); Yu and Pu (2008)). Rounding errors may lead two problems:

(i) The model function may not decrease numerically for some iterations. In this case, if there is no decrease in the function value for such iterations, trust region radii are expanded possibly several times which is an unnecessary expansion for them,

(ii) The model function may decrease numerically but the objective function may not decrease in the cases where iterations are near a valley, deep with a small creek at the bottom and steep sides. In this case, trust region radii are reduced possibly many times, leading to the production of quite a small radius, or even a failure.

Non-monotone techniques can be used in the hope of overcoming the second problem.

### 1.2 Overview of the new method

We suggest in Sect. 2 a new trust region-based limited memory algorithm for unconstrained black box least squares problems, called LMLS. This algorithm uses

- a non-monotone ratio and an adaptive radius formula to quickly reach the minimizer when the valley is narrow;
- a Broyden-like algorithm to get a decrease in the function value when the trust region radius is so small and iteration is unsuccessful;
- a finite difference approximation in a subspace with random basis indices to estimate the Jacobian matrix;
- either a Gauss–Newton or a dogleg algorithm in a subspace with random basis indices to solve the trust region subproblems.

Numerical results for small- to large-scale problems are given in Sect. 3 showing the fact that the new method is suitable for large-scale problems and is more robust and efficient than solvers using limited memory and standard quasi-Newton approximations.

### 2 The trust region method

In this section, we construct an improved trust region algorithm for handling problems in high dimensions:

- In Sect. 2.1 a Gauss-Newton direction in a subspace with random basis indices is introduced.
- In Sect. 2.2 a non-monotone term and an adaptive technique are constructed to quickly reach the minimizer in the presence of a narrow valley.
- In Sect. 2.3 a dogleg algorithm in a subspace with random basis indices is discussed.
- In Sect. 2.4 a Broyden-like technique is suggested based on the old best points.
- In Sect. 2.5 our algorithm using new enhancements is introduced.

We write \( J(x) \) for the Jacobian matrix of the residual vector \( E \) at \( x \). Then the gradient vector is \( g(x) := \nabla f(x) := J(x)^T E(x) \) and the Hessian matrix is

\[
G(x) := J(x)^T J(x) + \nabla^2 E(x)^T E(x)
\]

If the residual vector \( E(x) \) is small, the second term in \( G(x) \) is small. Hence, we approximate \( G(x) \) by the Gauss-Newton Hessian matrix \( J(x)^T J(x) \). We define the quadratic surro-
A new limited memory method for unconstrained nonlinear least squares

Next, if the complementary of old subspace basis indices set is determined. In this case, if $\mathcal{I}$ consists of at least $m_{sn}$ members, $\mathcal{I}$ is a random subset of $\{1, \ldots, m_{sn}\}$ with the $|S_{old}^c|$ members; otherwise, it is a permutation of $\{1, \ldots, |S_{old}^c|\}$. Then, a new subspace basis indices set is determined by $S := S_{old}^c(\mathcal{I})$ and its complementary is found by $S^c := S_{old}^c \setminus S$. But if $S_{old}^c$ is empty, a new subspace basis indices set and its complementary are restarted and chosen in the same way as the initial subspace basis indices set and its complementary, respectively.

For any $i \in S$,

- the step size is computed by

$$h_i := \begin{cases} \gamma_s & \text{if } x_i = 0, \\ \gamma_s (\text{sign } x_i) \max \{ |x_i|, \frac{\|x\|_1}{n} \} & \text{otherwise}, \end{cases}$$

where $0 < \gamma_s < 1$ is a tiny factor and sign $x_i$ identifies the sign of $x_i$, taking one of values $-1$ (if $x_i < 0$), 0 (if $x_i = 0$), and 1 (if $x_i > 0$).
- the random approximation coordinate direction $p$ discussed in Kimiaei (2020) is used with the difference that its $i$th component is updated by $p_i = p_i + h_i$.
- the new trial residual $E(x + p)$ and the new column $(E(x + p) - E)/h_i$ of the Jacobian matrix are computed.

It is well known that standard quasi-Newton methods are more robust than limited memory quasi-Newton ones, but they cannot handle problems in high dimensions; for standard quasi-Newton methods, see (Dennis and Moré 1977; Dennis and Walker 1981; Nocedal 1992; Schnabel 1989), and for limited memory quasi-Newton methods, see (Liu and Nocedal 1989; Nazareth 1979; Nocedal 1992). On the other hand, finite difference methods are more efficient than standard quasi-Newton ones. Hence, if used in a subspace with random basis indices, they can be more efficient than limited memory quasi-Newton methods for small- up to large-scale problems.

Using $S$ and $S^c$ generated and updated by SRFD, we construct a new subspace Gauss-Newton direction by

$$(p_{sn})_i := 0 \text{ for } i \in S^c \text{ and } (p_{sn})_S := -(J_S^T J_S)^{-1} J_S^T E. \quad (7)$$

### 2.2 New non-monotone and adaptive strategies

In this subsection, a new non-monotone term – stronger than the objective function $f$ – is constructed and a new adaptive radius formula to update $\Delta$ is derived from it. They help LMLS in finite precision arithmetic to quickly reach the minimizer in the cases where the valley is deep with a small creek at the bottom and steep sides.

Our non-monotone term is updated not only for successful but also for unsuccessful iterations that may have happened.
ratio is defined by

\[
\rho_{nm}^k := \frac{f_{nm}^k - f(x_{k-1} + p_{k-1}^1)}{Q_{k-1}(0) - Q_{k-1}(p_{k-1}^1)},
\]

where \(p_{k-1}^1\) is a solution of the following trust region subproblem in a subspace with the random basis indices set \(S\) by SRFD

\[
\min \tilde{Q}_{k-1}(p) := \frac{1}{2} \|E_{k-1} + J_{S,k-1} p\|^2 := f_{k-1} + p^T g_{S,k-1}
\]

\[+ \frac{1}{2} (J_{S,k-1} p)^T J_{S,k-1} p\]

s.t. \(p \in \mathbb{R}^m\) and \(\|p\| \leq \Delta_{nm}^k\)

with \(f_{k-1} := f(x_{k-1}), E_{k-1} := E(x_{k-1}), J_{S,k-1} := J_S(x_{k-1}),\) and \(g_{S,k-1} := (J_{S,k-1})^T E_{k-1}\).

### 2.3 A subspace dogleg algorithm

We define the Cauchy step by

\[
p_c := -t^* g_S, \quad t^* := \text{argmin}\{\tilde{Q}(-t g_S) | t \geq 0, \|t g_S\| \leq \Delta_{nm}\}.
\]

The goal is to solve the trust region subproblem (14) such that

\[
\|p\| \leq \Delta_{nm} \quad \text{and} \quad \tilde{Q}(p) \leq \tilde{Q}(p_c)
\]

hold. After the subspace Gauss-Newton direction is computed by (7), if it is outside a trust region, a subspace dogleg algorithm, called subDogleg, is used resulting in an estimated step enforcing (16).

The model function \(\tilde{Q}\) is reduced by (15) if \(d_{sn} := \tilde{Q}(0) - \tilde{Q}(p_{\text{sn}}) > 0\). subDogleg first identifies whether \(d_{sn} > 0\) or not. Then we have one of the following cases:

**CASE 1.** If \(d_{sn} > 0\), the scaled steepest descent step

\[
p_{sd} := - \frac{g_S^T g_S}{(J_S g_S)^T(J_S g_S)} g_S
\]

is computed. If it is outside the trust region, an estimated solution of (3) is either the Cauchy step computed by (15) or

\[
\begin{align*}
\rho_{nm}^k := & \frac{f_{nm}^k - f(x_{k-1} + p_{k-1}^1)}{Q_{k-1}(0) - Q_{k-1}(p_{k-1}^1)}, \\
\min \tilde{Q}_{k-1}(p) := & \frac{1}{2} \|E_{k-1} + J_{S,k-1} p\|^2 := f_{k-1} + p^T g_{S,k-1} \\
& + \frac{1}{2} (J_{S,k-1} p)^T J_{S,k-1} p \\
\text{s.t.} & \quad p \in \mathbb{R}^m \quad \text{and} \quad \|p\| \leq \Delta_{nm}^k
\end{align*}
\]
the **dogleg step**

\[ p_{dl} := p_{sd} + t(p_{sn} - p_{sd}); \quad (18) \]

both of (17) and (18) are on the trust region boundary. Here \( t \) is found by solving the equation \( p_{sd} + t(p_{sn} - p_{sd}) = \Delta_{nm} \). If the condition \( dp := (p_{sd})^T(p_{sn} - p_{sd}) \leq 0 \) holds, a positive root is computed by

\[ t := \frac{-dp + \sqrt{dp^2 + \|p_{sn} - p_{sd}\|^2(\Delta_{nm}^2 - \|p_{sd}\|^2)}}{\|p_{sn} - p_{sd}\|^2} \in (0, 1). \quad (19) \]

Otherwise, \( t \) is computed by

\[ t := \frac{\Delta_{nm}^2 - \|p_{sd}\|^2}{dp + \sqrt{dp^2 + \|p_{sn} - p_{sd}\|^2(\Delta_{nm}^2 - \|p_{sd}\|^2)}} \in (0, 1); \quad (20) \]

e.g., see Nielsen 2012.

**Case 2.** If \( dq_{sn} \leq 0 \), the model function \( \tilde{q} \) is convex since the matrix \((J_S g_S)^T(J_S g_S)\) is symmetric and positive semidefinite. An estimated solution of (3) is either \( p_{sd} \) computed by (17) if it is inside the trust region or the Cauchy step \( p := \Delta_{nm}(p_{sd}/\|p_{sd}\|) \) according to \( p_{sd} \), otherwise.

### 2.4 Broyden-like technique

Before a successful iteration is found by a trust region algorithm, the trust region subproblems may be solved many times with high computational cost. Instead, our idea is to use a new algorithm based on the previous best points in the hope of finding a point with good function value. Whenever **LMLS** cannot decrease the function value, a new Broyden-like technique, called **BroydenLike**, is used in the hope of getting a decrease in the function value. Let \( x^1, \ldots, x^{m_s} \) be the \( m_s \) best point stored in \( X \). Then a point in the affine space spanned by such points has the following form

\[ x_\xi := X z, \quad z \in \mathbb{R}^{m_s}, \quad e^T z = 1, \quad (21) \]

where \( e \in \mathbb{R}^{m_s} \) is a vector all of whose components are one. Given \( B := \begin{pmatrix} X \\ e \end{pmatrix} \), the linear approximation \( E(x_\xi) \approx B z \) is used to replace (1) by the surrogate problem

\[ \min \frac{1}{2} \|Bz\|^2 \quad \text{s.t.} \quad e^T z = 1. \quad (22) \]

This is a quality constrained convex quadratic problem in \( m_s \) variables and hence can be solved in closed form. Then a QR factorization is made in the form \( B = QR \), where \( Q \) is an orthogonal matrix and \( R \) is a square upper triangular matrix. By setting \( Z := R^{-1} \), we make the substitution \( z := Z y \), define \( a^T := e^T Z \), and obtain the \( m_s \)-dimensional minimal norm linear feasibility problem

\[ \min \frac{1}{2} \|y\|^2 \quad \text{s.t.} \quad a^T y = 1 \quad (23) \]

whose solution is \( y := a / \|a\| \). Hence, a new trial point for the next algorithm is

\[ x^{\text{trial}} := x_\xi := XR^{-1} a / \|a\|. \quad (24) \]

**BroydenLike** tries to find a point with better function value when no decrease in \( f \) is found along \( p \). It takes \( X, F, x, E, B, f, \delta_f \), and \( f_{nm} \) as input and uses the following tuning parameter:

\[ \gamma \in (0, 1) \quad \text{(tiny factor for adjusting } \delta_f), \]
\[ \gamma > 1 \quad \text{(parameter for expanding } \delta_f), \]
\[ \gamma_s \quad \text{(tiny parameter for the finite difference step size),} \]
\[ m_s \quad \text{(memory for affine space),} \]
\[ 0 < \gamma_r < 1 \quad \text{(tiny parameter for adjusting the scaled random directions).} \]

It returns a new value of \( \delta_f \) and \( f_{nm} \) (and \( f, X, F, E, S \), and \( J_S \) if a decrease in \( f \) is found) as output.

**Algorithm 1 BroydenLike, a Broyden-like method**

1. Make a QR factorization for the matrix \( B \) and get the square upper triangular matrix \( R \).

2. Compute the new trial point \( x^{\text{trial}} \) by (24).

3. Compute \( E^{\text{trial}} := E(x^{\text{trial}}) \) and set \( f^{\text{trial}} := 0.5 \|E^{\text{trial}}\|^2 \).

4. if \( f^{\text{trial}} \leq f_{nm} \) then \( \triangleright \) the new trial point is accepted

5. Set \( x := x^{\text{trial}}, E := E^{\text{trial}}, \) and \( f := f^{\text{trial}} \).

6. Save \( x \) in \( X \) and \( f \) in \( F \) by **update****XF**.

7. Compute \( S \) and \( J_S := J_S(x) \) by **SRFD**.

8. end if

9. Update \( \delta_f \) by (9) and compute \( f_{nm} \) by (10).

\[ \odot \text{Springer} \]
Since the Jacobian matrix may be singular or indefinite, a new point may move toward either a maximum point or saddle point. To remedy this disadvantage, BroydenLike does not lead to accept such a point with largest function value.

### 2.5 A limited memory trust region algorithm

We describe all steps of a new limited memory algorithm, called LMLS using the new subspace direction (7), the new non-monotone technique (10), the new adaptive radius strategy (11), and BroydenLike.

In each iteration, an estimated solution of the trust region subproblem (14) is found. Whenever the condition \( \rho_\text{nm} \geq \gamma_t \) holds, the iteration is successful while updating both the non-monotone term (10) and adaptive radius formula (11), and estimating the Jacobian matrix in a subspace with random basis indices by SRFD. Otherwise, the iteration is unsuccessful. In this case, BroydenLike is performed in the hope of finding a decrease in the function value. If a decrease in the function value is found, the iteration becomes successful; otherwise, it remains unsuccessful while reducing the radius and updating the non-monotone term (10) until a decrease in the function value is found and the iteration becomes successful.

LMLS solves unconstrained nonlinear black box least squares problem. This algorithm takes the initial point \( x^0 \), and maximal number of function evaluations (\( n_f \max \)). It uses the following tuning parameters:

- \( m_{sn} \) (subspace dimension),
- \( \gamma_t \in (0, 1) \) (parameter for trust region),
- \( 0 < \sigma_1 < 1 < \sigma_2 \) (parameters for updating \( \lambda \)),
- \( \gamma^\text{init} \) (parameter for updating the initial \( \delta_f \)),
- \( 0 < \gamma_f < 1 \) (tiny factor for adjusting \( \delta_f \)),
- \( \lambda^\text{init} \) (lower bound for \( \lambda \)),
- \( \lambda^\text{min} \) (upper bound for \( \lambda \)),
- \( \gamma_s \) (tiny parameter for adjusting finite difference step sizes),
- \( 0 < \gamma_r < 1 \) (tiny parameter for adjusting random approximation coordinate directions).

It returns a solution \( x^\text{best} \) of a nonlinear least squares problem as output.

LMLS was implemented in MATLAB; the source code is available at

http://www.mat.univie.ac.at/~neum/software/LMLS.

---

**Algorithm 2 LMLS, limited memory method for least squares problems**

**Initialization**

1: Choose \( x^0 \in (\underline{x}, \overline{x}) \).
2: Compute \( E^0 := E(x^0) \) and set \( f^0 := 0.5 \| E^0 \| ^2 \). Then set \( X := x^0 \) and \( F := f^0 \).
3: Identify the initial subspace basis indices set \( S \) and \( J_S(x^0) \) by SRFD.
4: Compute \( g^0 := J_S^T(x^0) E^0 \) and \( \delta_S^0 \) by (9).
5: for \( \ell = 0, 1, 2, \ldots \) do

   Compute the \( \ell \text{th} \) search direction \( p^\ell \)

   6: Compute the subspace Gauss-Newton direction \( p^\ell := p^\ell_0 \) by (7).

   7: if \( \| p^\ell_0 \| \geq \Delta_{nm}^\ell \) then

      Enforcing the descent condition \( (g^\ell)^T p^\ell < 0 \) if it needs

      10: if \( (g^\ell)^T p^\ell \geq 0 \) then

         Due to rounding errors

         11: Find \( \text{ind} = \{ i \mid g_i^\ell p_i^\ell > 0 \} \).

         12: Set \( p_i^\ell := -p_i^\ell \) for \( i \in \text{ind} \).

      end if

   end if

   Compute a new trial point and its function value

   14: Compute \( x^\text{trial} := x^\ell + p^\ell, E^\text{trial} := E(x^\text{trial}), \) and \( f^\text{trial} := 0.5 \| E^\text{trial} \| ^2 \).

   Stopping test

   15: if \( n_f \max \) is exceeded then

      LMLS ends, resulting \( x^\text{best} = x^\ell \).

   end if

   Identify whether the \( \ell \text{th} \) iteration is successful or not

   18: Compute \( \rho_\text{nm}^\ell \) by (13).

   19: if \( \rho_\text{nm}^\ell < \gamma_t \) then

      Unsuccessful iteration

      20: Call BroydenLike with the goal of satisfying \( f^\text{trial} \leq f_{\text{nm}} \).

      21: if \( f^\text{trial} > f_{\text{nm}} \) then

         BroydenLike cannot decrease \( f \)

         22: Update \( \delta_S^\ell \) by (9) and \( f^\ell_\text{nm} \) by (10).

         23: Reduce \( \lambda^\ell \) to (12) and the radius \( \Delta_{nm}^\ell \) to (11).

         end if

      else

         Successful iteration

         26: Set \( x^{\ell + 1} := x^\text{trial}, f^{\ell + 1} := f^\text{trial} \), and \( E^{\ell + 1} := E^\text{trial} \).

         27: Compute the new indices set \( S \) and \( J_S(x^{\ell + 1}) \) by SRFD.

         28: Compute \( g^{\ell + 1} := J_S(x^{\ell + 1})^T E^{\ell + 1} \).

         29: Update \( \delta_S^{\ell + 1} \) by (9) and \( f^{\ell + 1}_{\text{nm}} \) by (10).

         30: Expand \( \delta_S^{\ell + 1} \) to (12) and \( \Delta_{nm}^{\ell + 1} \) to (11).

         31: Store \( x^{\ell + 1} \) in \( X \) and \( f^{\ell + 1} \) in \( F \) by updateXF.

         end if

   end if

33: end for
3 Numerical results

We updated the test environment constructed by Kimiae et al. (2019) to use test problems suggested by Lukšan et al. (2018). LMLs is compared with unconstrained least squares and unconstrained optimization solvers, for some of which we had to choose options different from the default to make them competitive in the first subsection.

3.1 Codes compared

Least squares solvers:

- **CoDoSol** is a solver for constrained nonlinear systems of equations, obtained from

  [http://codosol.de.unifi.it](http://codosol.de.unifi.it).

  It combines Newton method and a trust region method, see Bellavia et al. (2012). The following option was used:

  \[
  \text{parms} = [\text{maxit}, \text{maxnf}, \text{tr}, \text{delta}, \text{scaling}, \text{outflag}] = [\text{inf}, \text{nfmax}, 1, -1, 0, 2].
  \]

  Note that \( \text{delta} = -1 \) means that \( \Delta_0 = 1 \). According to our numerical results, CoDoSol was not sensitive for the initial radius; hence, the default was used.

- **STRSCNE** is a solver for constrained nonlinear systems of equations, obtained from

  [http://codosol.de.unifi.it](http://codosol.de.unifi.it).

  It combines Newton method and a trust region procedure, see Bellavia et al. (2004). The option

  \[
  \text{parms} = [\text{maxit}, \text{maxnf}, \text{delta}, \text{outflag}]
  = [\text{inf}, \text{nfmax}, -1, 0]
  \]

  was used. Note that \( \text{delta} = -1 \) means that \( \Delta_0 = 1 \). According to our numerical results, STRSCNE was not sensitive for the initial radius; hence, the default was used for \( \Delta_0 \).

- **NLEQ1** is a damped affine invariant Newton method for nonlinear systems, obtained from

  [http://elib.zib.de/pub/elib/codelib/nleq1_m/nleq1.m](http://elib.zib.de/pub/elib/codelib/nleq1_m/nleq1.m) and suggested by Deuflhard (2011) and written by Nowak and Weimann (1990). The default tuning parameters were used; only \( \text{iop} \text{t.jacgen} = 2 \), \( \text{iop} \text{t.qrankl} = 1 \), and \( \text{wk} \text{f.min} = 1e-50 \) were selected.

- **NLEQ2** is the same as NLEQ1; only \( \text{iop} \text{t.qrankl} = 0 \) was selected.

- **MINLBFGS** is a L-BFGS with line search algorithm, obtained from

  [https://www.tensorlab.net/](https://www.tensorlab.net/)

  and written by Sorber et al. (2012). The default parameters were used, except for \( \text{m} \text{in} \text{fiter} = \text{nfmax} \), \( \text{TolFun} = 1e-50 \), \( \text{TolX} = 1e-50 \). The default parameters were chosen for PlaneSearch and \( M \).

  - **MINLBFGSDL1**, **MINLBFGSDL2**, **MINLBFGSDL3**, and **MINLBFGSDL4** are MINLBFGSDL with
    \[
    \begin{align*}
    \text{(1)} & \quad \text{PlaneSearch} = \text{false} \text{ and } M = \text{min}(30, n), \\
    \text{(2)} & \quad \text{PlaneSearch} = \text{false} \text{ and } M = \text{min}(100, n), \\
    \text{(3)} & \quad \text{PlaneSearch} = \text{true} \text{ and } M = \text{min}(30, n), \\
    \text{(4)} & \quad \text{PlaneSearch} = \text{true} \text{ and } M = \text{min}(100, n).
    \end{align*}
    \]

  According to our results, MINLBFGSDL1 was the best.

- **MINFNCG** is a nonlinear conjugate gradient solver, obtained from

  [https://www.tensorlab.net/](https://www.tensorlab.net/)

  and written by Sorber et al. (2012). We used the following option set

  \[
  \text{MaxIter} = \text{nfmax}; \quad \text{TolFun} = 1e-50; \quad \text{TolX} = 1e-50.
  \]

  The other tuning parameter was \( \text{Beta} \in \{\text{HS}, \text{HSm}, \text{PR}, \text{FR}, \text{PRm}, \text{SD}\} \). MINFNCG1, MINFNCG2, MINFNC3, MINFNCG4, MINFNCG5, and MINFNCG6 are MINFNCG with \( \text{Beta} = \text{HS} \), \( \text{Beta} = \text{HSm} \), \( \text{Beta} = \text{PR} \), \( \text{Beta} = \text{FR} \), \( \text{Beta} = \text{PRm} \), and \( \text{Beta} = \text{SD} \), respectively.

- **NLSQERR** is a global unconstrained Gauss-Newton method with error oriented convergence criterion and adaptive trust region strategy Deuflhard (2011), obtained from

  [http://elib.zib.de/pub/elib/codelib/NewtonLib/index.html](http://elib.zib.de/pub/elib/codelib/NewtonLib/index.html)

  The following options were used

  \[
  \begin{align*}
  \text{iniscalx} &= 0; \quad \text{rescalx} = 0; \quad \text{xthrsh} = \text{ones}(n, 1); \\
  \text{xtol} &= 1.0e-50; \quad \text{ftol} = 1.0e-50; \quad \text{kmmax} = \text{nfmax}; \\
  \text{printmon} &= 2; \quad \text{printsol} = 1; \quad \text{fid} = 1;
  \end{align*}
  \]
• NMPNTR, non-monotone projected Newton trust region method, is a bound constrained solver Kimiaei (2016). NMPNTR1, NMPNTR2, NMPNTR3, and NMPNTR4 are NMPNTR with \( \Delta_0 = 1, \Delta_0 = 10, \Delta_0 = 100, \) and \( \Delta_0 = 500, \) respectively. According to our results, NMPNTR2 was the best.

• NATRN is a non-monotone trust region algorithm Amini et al. (2016) using the full finite difference approximation. The subproblem was solved in the same way as LMLS. NATRN1, NATRN2, NATRN3, and NATRN4 are NATRN with \( \Delta_0 = 1, \Delta_0 = 10, \Delta_0 = 100, \) and \( \Delta_0 = 500, \) respectively. According to our results, NATRN1 had the best performance.

• NATRLS is a non-monotone line search and trust region algorithm Amini et al. (2016) using the full finite difference approximation. The subproblem was solved in the same way as LMLS. NATRLS1, NATRLS2, NATRLS3, and NATRLS4 are NATRLS with \( \Delta_0 = 1, \Delta_0 = 10, \Delta_0 = 100, \) and \( \Delta_0 = 500, \) respectively. According to our results, NATRLS1 had the best performance.

• LSQNONLIN1, obtained from the MATLAB Optimization Toolbox at

\[ \text{https://de.mathworks.com/help/optim/ug/lsqlin.html} \]

is a nonlinear least squares solver with the following options:

\[
\text{options} = \text{optimoptions}(\text{@lsqlin}, \text{Algorithm}, \text{‘levenberg-marquardt’}, \text{‘FiniteDifferenceType’,’forward’, \text{‘MaxIter’}, \text{Inf}, \text{MaxFunEvals’}, \text{nfmax’}, \text{‘TolX’}, 0, \text{’Specify ObjectiveGradient’,’false’}).
\]

• LSQNONLIN2, obtained from the MATLAB Optimization Toolbox at

\[ \text{https://de.mathworks.com/help/optim/ug/lsqlin.html} \]

is a nonlinear least squares solver with the following options:

\[
\text{options} = \text{optimoptions}(\text{@lsqlin}, \text{Algorithm}, \text{‘trust-region reflective’}, \text{‘FiniteDifferenceType’,’forward’, \text{‘MaxIter’}, \text{Inf}, \text{MaxFunEvals’}, \text{nfmax’}, \text{‘TolX’}, 0, \text{’Specify ObjectiveGradient’,’false’}).
\]

- DOGLEG is Powell’s dogleg method for least squares problems, which is the best algorithm from the toolbox of immoptibox.zip Nielsen (2012), available at

\[ \text{http://www2.imm.dtu.dk/projects/immoptibox/} \]

The following option was used

\[
\text{opts} = \text{[}\Delta_0, \text{tolg, tolx, tolr, maxeval]}\]

\[
\Delta_0, \text{1e}–50, \text{1e}–50, \text{1e}–50, \text{nfmax}].
\]

• FMINUNC, obtained from the MATLAB Optimization Toolbox at

\[ \text{https://ch.mathworks.com/help/optim/ug/fminunc.html} \]

is a standard quasi-Newton algorithm. We used FMINUNC with

\[
\text{opts} = \text{optimoptions}(\text{@fminunc}, \text{‘Algorithm’,’quasi-newton’, ‘Display’,’iter’, ‘Max Iter’,Inf,’MaxFunEvals’, nfmax, ‘TolX’, 0, \text{’TolFun’,0,’ObjectiveLimit’,’-1e-50’).}
\]

• FMINUNC1 is FMINUNC with the limited memory quasi-Newton approximation by Liu and Nocedal (1989). It were added to FMINUNC by the present authors. The option set for it was used the same as FMINUNC; only the memory \( m = 10 \) was added to the option set.

3.2 Default for tuning parameters of LMLS

The tuning parameters for our new method (LMLS) are chosen as

\[
\begin{align*}
m_{\text{min}} &= 10; \ m_{\text{ts}} = 10; \ \gamma_r = 10^{-30}; \ \gamma_p = 5 \varepsilon_m; \\
\gamma_s = \sqrt{\varepsilon_m}; \ \gamma_t = 0.1; \ \sigma_1 = 0.5; \ \lambda^0 = 1; \\
\sigma_2 = 1; \ \gamma^\text{init} = 10^{-8}; \ \gamma_m = 10^{-30}; \ \Delta = 10^{-4}; \\
\Delta_0 = 10; \ \Delta_\text{min} = 10^{-6}, \ \lambda = 10^5.
\end{align*}
\]

The remaining tuning parameter \( m_{\text{sn}} \) is varied in the experiment: LMLS1, LMLS2, LMLS3, and LMLS4 are LMLS with \( m_{\text{sn}} = 3, \ m_{\text{sn}} = \text{min}(10, n), \ m_{\text{sn}} = \text{min}(30, n), \) and \( m_{\text{sn}} = \text{min}(100, n), \) respectively.
3.3 Test problems used, initial point, and stopping tests

Test problems suggested by Lukšan et al. (2018) are classified in Table 1 according to whether they are overdetermined \((r > n)\) or not \((r = n)\). A shifted point for these problems is done like Kimiaei and Neumaier (2020) as \(\phi_i := (-1)^{i-1/2}/(2 + i)\) for all \(i = 1, \ldots, n\). This means that the initial point is chosen by \(x_0^i = \phi_i\) for all \(i = 1, \ldots, n\) and the initial function value is computed by \(f_0 := f(x_0)\), while the other function values are computed by \(f^\ell := f(x^\ell + \phi)\) for all \(\ell \geq 1\).

We denote \(n\) and \(msec\) as the number of function evaluations and time in milliseconds, respectively. \(n\) and \(msec\) are the upper bounds for them, chosen as

\[
\begin{align*}
nf &\in \begin{cases} 
{10n, 50n, 100n, 500n} & \text{if } 1 \leq n \leq 100, \\
{10n, 50n, 100n} & \text{if } 101 \leq n \leq 1000 \\
{10n, 100n} & \text{if } 1001 \leq n \leq 10000
\end{cases}, \\
msec &\in \begin{cases} 
300 & \text{if } 1 \leq n \leq 100, \\
800 & \text{if } 101 \leq n \leq 10000.
\end{cases}
\end{align*}
\]

Denote by \(f^0\) the function value of the starting point (common to all solvers), by \(f^{\text{sol}}\) the best point found by the solver so, and by \(f^{\text{opt}}\) the best point known to us. Then, if the target accuracy satisfies

\[
q^{\text{sol}} := \frac{(f^{\text{sol}} - f^{\text{opt}})/(f^0 - f^{\text{opt}})}{\begin{cases} 
10^{-8} & \text{if } 1 \leq n \leq 100, \\
10^{-3} & \text{if } 101 \leq n \leq 10000,
\end{cases}}
\]

then the problem is solved by the solver so. Otherwise, the problem is unsolved by it; either \(n\) or \(msec\) is exceeded, or the solver fails.

3.4 The efficiency and robustness of a solver

For a given collection \(S\) of solvers, the strength of a solver \(so \in S\)—relative to an ideal solver that matches on each problem the best solver—is measured, for any given cost measure \(c_s\) by the number, \(e_{so}\), defined by

\[
e_{so} := \begin{cases} 
\min_{s \in S} \frac{c_s}{c_{so}} & \text{if the solver } so \text{ solves the problem}, \\
0, & \text{otherwise},
\end{cases}
\]

called the efficiency of the solver so with respect to this cost measure. Two cost measures \(n\) and \(msec\) are used.

The robustness of a solver is how many test problems it can solve. Efficiency and robustness are two adequate tools to determine which solver is competitive. In fact, the robustness of a solver is more important than its efficiency. We use two different performance plots in terms of the robustness and efficiency of solvers:

- The first performance plot is the data profile by Moré and Wild (2009) for \(n\)\/% of solved problems and \(msec\) as well; but it is the percentage of problems solved within the number of function evaluations and time in milliseconds.
- The second performance plot is the performance profile by Dolan and Moré (2002) for \(n\)\/% of solved problems and \(msec\); the percentage of problems solved within a factor \(r\) of the best solvers.

All tables and data/performance profiles are given in Sects. 4.2–4.4. In Sects. 3.5–3.7, we summarize them as two new performance plots.

3.5 Small scale: \(n \in \{1, 100\}\)

A comparison among \textsc{LMLS1}, \textsc{LMLS2}, \textsc{LMLS3}, \textsc{LMLS4}, and solvers using quasi-Newton is shown in Subfigures (a) and (b) of Fig. 1, so that

- \textsc{LMLS4} using the full estimated Jacobian matrix is the best in terms of the number of solved problems and the \(n\)\/% efficiency;
- \textsc{LMLS3}, \textsc{LMLS2}, and \textsc{LMLS1} are more efficient than solvers using quasi-Newton approximation (\textsc{FMINUNC}, \textsc{FMINUNC1}, \textsc{MINFLBFGS1}, and \textsc{MINFLBFGSDL1}) in terms of the \(n\)\/% efficiency;
- \textsc{FMINUNC} and \textsc{MINFLBFGSDL1} are comparable with \textsc{LMLS3}—only for very large budget—in terms of the number of solved problems but \textsc{LMLS3}, \textsc{LMLS2}, and \textsc{LMLS1} are more efficient than \textsc{FMINUNC} and \textsc{MINFLBFGSDL1} in terms of the \(n\)\/% efficiency not only for very large budget but also for small up to large budgets.

To determine whether our new non-monotone and adaptive radius strategies are effective or not, we compare \textsc{LMLS4} with solvers using other non-monotone and adaptive radius strategies, shown in Subfigures (c) and (d) of Fig. 1. All solvers use the full Jacobian matrix and the trust region subproblems are solved by the same algorithm. As can be seen, \textsc{LMLS4} is much more efficient and robust than \textsc{NATRLS1}, \textsc{NMPGTR2}, and \textsc{NATRN1} in terms of the number of solved problems and the \(n\)\/% efficiency.

We compare \textsc{LMLS4} with four famous solvers \textsc{LSQN}, \textsc{LIN}, \textsc{CoDoSo1}, \textsc{NLEQ1}, and \textsc{DOGLEG1} shown in Subfigures (e) and (f) of Fig. 1. It is seen that \textsc{LMLS4} and \textsc{CoDoSo1} are the two best solvers in terms of the \(n\)\/%
Table 1 A classification of test problems

| Dimensions n | 3   | 5   | 10  | 16  | 30  | 50  | 100 | 300 | 500 | 1000 | 5000 | 10,000 |
|---------------|-----|-----|-----|-----|-----|-----|-----|-----|-----|------|------|--------|
| Total number of least squares problems \((r \geq n)\) | 49  | 69  | 76  | 83  | 83  | 83  | 83  | 83  | 83  | 83   | 83   | 83     |
| Number of square problems \((r = n)\) | 43  | 53  | 55  | 62  | 62  | 62  | 62  | 62  | 62  | 62   | 62   | 62     |
| Number of least squares problems with \(r > n\) | 6   | 16  | 21  | 21  | 21  | 21  | 21  | 21  | 21  | 21   | 21   | 21     |

efficiency while **LMLS4** and **DOGLEG1** are the two best solvers in terms of the number of solved problems.

Another comparison is among **LMLS3**, **LMLS2**, and **LMLS1** using the Jacobian matrix in an adaptive subspace basis indices set and **LSQNONLIN1** and **NLEQ1** using the full Jacobian matrix. We conclude from Subfigures (g) and (h) of Fig. 1 that

- **LMLS3** is the best in terms of the number of solved problems and the \(n_f\) efficiency;
- **LMLS2** is the second best solver in terms of the number of solved problems and the \(n_f\) efficiency for medium, large, and very large budgets;
- **LMLS1** with lowest subspace dimension is more efficient than **LSQNONLIN1** in terms of the number of solved problems and the \(n_f\) efficiency; even it is more efficient than **NLEQ1** for very large budget in terms of the \(n_f\) efficiency.

As a result, **LMLS** is competitive for small-scale problems in comparison with the state-of-the-art solvers.

### 3.6 Medium scale: \(n \in [101, 1000]\)

In this subsection, we compare **LMLS1**, **LMLS2**, **LMLS3**, and **LMLS4** using the estimated Jacobian matrices in a subspace with random basis indices set and **FMINUNC** using standard BFGS approximations and **FMINUNC1** using limited memory BFGS ones (Fig. 2).

From Subfigures (a) and (b) of Fig. 1, we conclude that

- **LMLS4**, **LMLS3**, and **LMLS2** are the three best solvers in terms of the \(n_f\) efficiency, respectively;
- **LMLS4** is the best solver in terms of the number of solved problems; only **FMINUNC** is the best for large budget.

In terms of the \(n_f\) efficiency and the number of solved problems, Subfigures (a) and (b) of Fig. 3 result in the fact that

- **LMLS4**, **LMLS3**, and **LMLS2** are the three best solvers, respectively;
- **LMLS1** with lowest subspace dimension is more efficient than **FMINUNC1** for small budget while **FMINUNC1** is more efficient than **LMLS1** for large budget.

### 4 Additional material for LMLS

#### 4.1 Summarizing tables

In all tables, efficiencies are given as percentages, rounded (towards zero) to integers. Larger efficiencies imply a better average behavior, while a zero efficiency indicates failure.

\#100 is the total number of problems for which the solver \(so\) was best with respect to \(n_f\) \((e_{so} = 1 = 100\%)\). \#100 is the total number of problems solved for which the solver \(so\) was better than all other solvers with respect to \(n_f\).

We denote the time in seconds without the setup time for the objective function by \(sec\). In tables, a sign

- \(n\) indicates that \(n_f \geq n_f^{max}\) was reached.
- \(t\) indicates that \(sec \geq sec^{max}\) was reached.
- \(f\) indicates that the algorithm failed for other reasons.

\(T_{mean}\) is the mean of the time in seconds needed by a solver to solve the test problems chosen from the list of test problems \(P\), ignoring the times for unsolved problems. It can be a good measure when solvers have approximately the same number of solved problems.
Fig. 1 Performance plots for small-scale problems. a-b: A comparison of limited memory solvers, c-d: A comparison among LMLS in a full subspace with random basis indices and solvers using other non-monotone and adaptive radius techniques, e-f: A comparison among LMLS in a full subspace with random basis indices and other famous solvers, g-h: A comparison among low-dimensional LMLS1, LMLS2, LMLS3, and NLEQ1 and LSQNONLIN1 using full estimated Jacobian.
Fig. 2  a–b: Performance plots for medium-scale problems

Fig. 3  a–b: Performance plots for large-scale problems
4.2 Tables and data/performance profiles for \(1 \leq n \leq 100\)

This section contains Tables 2, 3, 4, 5 and Figs. 4, 5, 6, 7, summaries of which were discussed in Sect 3.5.

Table 2 Results for small-scale and small budget

| Solver   | Solved | #100 | #100 | #mean | # of anomalies | Mean efficiency in % |
|----------|--------|------|------|-------|---------------|---------------------|
| LMLS4    | 304    | 204  | 15   | 52    | 222           | 53                  |
| CoDoSol1 | 300    | 197  | 30   | 56    | 219           | 52                  |
| NATRLS1  | 284    | 63   | 8    | 67    | 242           | 40                  |
| DOGLEG1  | 280    | 70   | 11   | 98    | 246           | 38                  |
| NMPGTR2  | 254    | 172  | 8    | 55    | 272           | 43                  |
| LMLS3    | 252    | 156  | 4    | 53    | 274           | 42                  |
| NLEQ1    | 201    | 42   | 30   | 77    | 145           | 33                  |
| LMLS2    | 184    | 91   | 7    | 75    | 342           | 29                  |
| NTRLN1   | 179    | 54   | 11   | 89    | 347           | 26                  |
| LMLS1    | 110    | 47   | 20   | 112   | 416           | 16                  |
| STRSCLNE1| 70     | 69   | 0    | 28    | 107           | 13                  |
| FMINUNC1 | 69     | 4    | 0    | 123   | 455           | 6                   |
| FMINUNC  | 58     | 2    | 0    | 139   | 468           | 5                   |
| MINFLBFSG1| 23      | 0    | 0    | 227   | 449           | 1                   |
| LSQNONLIN1| 1    | 1    | 1    | 50    | 525           | 0                   |
| MINFLBFSGD1| 1      | 0    | 0    | 30    | 525           | 0                   |
| MINFNCG1 | 0      | 0    | 0    | –     | 514           | 0                   |

Table 3 Results for small-scale and medium budget

| Solver   | Solved | #100 | #100 | #mean | # of anomalies | Mean efficiency in % |
|----------|--------|------|------|-------|---------------|---------------------|
| DOGLEG1  | 419    | 80   | 17   | 148   | 107           | 55                  |
| LMLS4    | 416    | 228  | 17   | 108   | 110           | 66                  |
| CoDoSol1 | 398    | 226  | 52   | 90    | 107           | 65                  |
| NATRLS1  | 364    | 79   | 21   | 81    | 162           | 51                  |
| LMLS3    | 359    | 177  | 8    | 124   | 167           | 55                  |
| NMPGTR2  | 343    | 191  | 22   | 79    | 183           | 53                  |
| LMLS2    | 282    | 111  | 16   | 197   | 244           | 40                  |
| NTRLN1   | 222    | 63   | 17   | 117   | 304           | 31                  |
| NLEQ1    | 222    | 42   | 30   | 77    | 122           | 35                  |
| FMINUNC  | 219    | 4    | 3    | 167   | 292           | 13                  |
| LMLS1    | 206    | 49   | 21   | 253   | 320           | 24                  |
| FMINUNC1 | 179    | 8    | 7    | 177   | 345           | 13                  |
| MINFLBFSGD1| 168    | 0    | 0    | 286   | 358           | 5                   |
| MINFLBFSG1| 133    | 0    | 0    | 200   | 339           | 5                   |
| LSQNONLIN1| 131    | 1    | 1    | 314   | 395           | 5                   |
| STRSCLNE1| 72     | 69   | 0    | 28    | 1             | 13                  |
| MINFNCG1 | 47     | 0    | 0    | 698   | 458           | 1                   |
Table 4  Results for small-scale and large budget

Stopping test: $q^{\text{so}} \leq 1\text{e-08}$, sec $\leq 300$, nf $\leq 100\times n$

| dim $\in [1,100]$ | Solved | #100 | !100 | $T_{\text{mean}}$ | # of anomalies | Mean efficiency in % |
|-------------------|--------|-------|------|----------------|----------------|---------------------|
|                   |        |       |      |                | #n #t #f       |                     |
|                   |        |       |      |                | nf msec        |                     |
| DOGLEG1           | dogleg1| 434   | 82   | 19            | 167           | 92 0 0 56 31        |
| LMLS4             | lmtr4  | 424   | 229  | 17            | 113           | 102 0 0 67 59       |
| CoDoSol1          | codo1  | 408   | 227  | 53            | 103           | 88 0 30 66 54       |
| NATRLS1           | natr1  | 375   | 87   | 27            | 104           | 151 0 0 53 39       |
| LMLS3             | lmtr3  | 374   | 177  | 9             | 150           | 152 0 0 56 48       |
| NMPGTR2           | nmpg2  | 365   | 196  | 25            | 103           | 160 0 1 54 40       |
| FMINUNC           | func   | 313   | 6    | 4             | 351           | 170 0 43 16 14      |
| LMLS2             | lmtr2  | 306   | 106  | 12            | 257           | 220 0 0 41 31       |
| MINFLBFGSDL1      | minlbgs1| 272  | 2    | 2             | 453           | 244 0 10 9 8        |
| LMLS1             | lmtr1  | 231   | 53   | 24            | 437           | 295 0 0 25 17       |
| MINFLBFGS1        | lbfgs1 | 230   | 0    | 0             | 349           | 232 0 64 8 6        |
| NATRN1            | natrn1 | 225   | 65   | 19            | 121           | 301 0 0 32 22       |
| NLEQ1             | nleq1  | 223   | 43   | 30            | 79            | 121 0 182 35 17     |
| FMINUNC1          | func1  | 209   | 8    | 6             | 327           | 297 0 20 14 12      |
| LSQNONLIN1        | lsqn1  | 155   | 1    | 1             | 374           | 371 0 0 6 4         |
| MINFNC1           | MINFNC1| 136   | 0    | 0             | 524           | 365 0 25 2 2        |
| STRSCNE1          | strs1  | 72    | 69   | 0             | 28            | 1 0 453 13 7        |

Table 5  Results for small-scale and very large budget

Stopping test: $q^{\text{so}} \leq 1\text{e-08}$, sec $\leq 300$, nf $\leq 500\times n$

| dim $\in [1,100]$ | Solved | #100 | !100 | $T_{\text{mean}}$ | # of anomalies | Mean efficiency in % |
|-------------------|--------|-------|------|----------------|----------------|---------------------|
|                   |        |       |      |                | #n #t #f       |                     |
|                   |        |       |      |                | nf msec        |                     |
| DOGLEG1           | dogleg1| 444   | 79   | 18            | 220           | 82 0 0 57 31        |
| LMLS4             | lmtr4  | 437   | 229  | 18            | 159           | 89 0 43 68 58       |
| CoDoSol1          | codo1  | 422   | 226  | 54            | 172           | 61 0 47 67 55       |
| FMINUNC           | func   | 411   | 8    | 7             | 761           | 20 0 95 19 17       |
| NMPGTR2           | nmpg2  | 407   | 194  | 25            | 378           | 110 1 8 56 42       |
| LMLS3             | lmtr3  | 393   | 178  | 9             | 343           | 133 0 0 56 47       |
| MINFLBFGSDL1      | minlbgs1| 385  | 6    | 6             | 723           | 69 0 72 11 12       |
| NATRLS1           | natr1  | 384   | 89   | 31            | 143           | 142 0 0 54 41       |
| LMLS2             | lmtr2  | 353   | 107  | 14            | 998           | 173 0 0 42 31       |
| MINFLBFGS1        | lbfgs1 | 342   | 4    | 4             | 973           | 56 0 128 10 9       |
| LMLS1             | lmtr1  | 300   | 53   | 25            | 1302          | 226 0 0 27 17       |
| MINFNC1           | MINFNC1| 283   | 0    | 0             | 990           | 194 0 49 3 4        |
| FMINUNC1          | func1  | 283   | 8    | 7             | 534           | 205 0 38 15 13      |
| NATRN1            | natrn1 | 230   | 63   | 17            | 178           | 296 0 0 32 22       |
| NLEQ1             | nleq1  | 224   | 42   | 30            | 82            | 120 0 182 35 18     |
| LSQNONLIN1        | lsqn1  | 209   | 1    | 1             | 690           | 316 1 0 6 5         |
| STRSCNE1          | strs1  | 72    | 69   | 0             | 29            | 1 0 453 13 6         |
Fig. 4 a and b: Data profiles for \( \text{nf}/(\text{best nf}) \) and \( \text{msec}/(\text{best msec}) \), respectively. \( \rho \) designates the fraction of problems solved within the number of function evaluations and time in milliseconds used by the best solver. Problems solved by no solver are ignored. c-d: Performance profiles for \( \text{nf}/(\text{best nf}) \) and \( \text{msec}/(\text{best msec}) \), respectively. \( \rho \) designates the fraction of problems solved within the number of function evaluations and time in milliseconds used by the best solver. Problems solved by no solver are ignored.
Fig. 5  Details as in Fig. 4
Fig. 6  Details as in Fig. 4
Fig. 7  Details as in Fig. 4
4.3 Tables and data/performance profiles for $101 \leq n \leq 1000$

This section contains Tables 6, 7, 8 and Figs. 8, 9, 10, summaries of which were discussed in Sect 3.6.

Table 6 Results for medium-scale and small budget

| Solver  | Solved | #100 | #!100 | $T_{\text{mean}}$ | # of anomalies | Mean efficiency in % |
|---------|--------|------|-------|-------------------|----------------|----------------------|
|         |        |      |       |                   | $\#_n$ | $\#_t$ | $\#_f$ | nf | msec |
| LMLS4   | 119    | 49   | 48    | 13738            | 129   | 1    | 0    | 37 | 37   |
| FMINUNC1| 114    | 40   | 14    | 11271            | 133   | 2    | 0    | 29 | 35   |
| LMLS3   | 110    | 30   | 28    | 15401            | 138   | 1    | 0    | 34 | 28   |
| LMLS2   | 105    | 22   | 21    | 15006            | 143   | 1    | 0    | 31 | 20   |
| LMLS1   | 97     | 15   | 15    | 18207            | 151   | 1    | 0    | 27 | 21   |
| FMINUNC | 93     | 26   | 0     | 13702            | 154   | 2    | 0    | 22 | 24   |

Table 7 Results for medium-scale and budget

| Solver  | Solved | #100 | #!100 | $T_{\text{mean}}$ | # of anomalies | Mean efficiency in % |
|---------|--------|------|-------|-------------------|----------------|----------------------|
|         |        |      |       |                   | $\#_n$ | $\#_t$ | $\#_f$ | nf | msec |
| LMLS4   | 169    | 58   | 56    | 11582            | 78   | 2    | 0    | 50 | 46   |
| LMLS3   | 168    | 48   | 47    | 12980            | 79   | 2    | 0    | 47 | 37   |
| FMINUNC | 168    | 32   | 5     | 11064            | 78   | 3    | 0    | 36 | 42   |
| FMINUNC1| 165    | 43   | 16    | 10503            | 81   | 3    | 0    | 39 | 50   |
| LMLS2   | 164    | 16   | 15    | 15614            | 84   | 1    | 0    | 43 | 25   |
| LMLS1   | 159    | 20   | 20    | 15399            | 88   | 2    | 0    | 38 | 28   |

Table 8 Results for medium-scale and large budget

| Solver  | Solved | #100 | #!100 | $T_{\text{mean}}$ | # of anomalies | Mean efficiency in % |
|---------|--------|------|-------|-------------------|----------------|----------------------|
|         |        |      |       |                   | $\#_n$ | $\#_t$ | $\#_f$ | nf | msec |
| FMINUNC | 189    | 44   | 18    | 14603            | 55   | 3    | 2    | 41 | 48   |
| FMINUNC1| 178    | 46   | 20    | 12133            | 66   | 3    | 2    | 41 | 52   |
| LMLS3   | 176    | 39   | 37    | 16832            | 71   | 2    | 0    | 48 | 36   |
| LMLS4   | 174    | 53   | 51    | 18938            | 73   | 2    | 0    | 50 | 47   |
| LMLS2   | 167    | 24   | 24    | 18182            | 81   | 1    | 0    | 42 | 24   |
| LMLS1   | 165    | 20   | 20    | 28711            | 84   | 0    | 0    | 38 | 26   |
Fig. 8  Details as in Fig. 4
Fig. 9  Details as in Fig. 4
Fig. 10  Details as in Fig. 4
4.4 Tables and data/performance profiles for \(1001 \leq n \leq 10000\)

This section contains Tables 9, 10 and Figs. 11, 12, summaries of which were discussed in Sect 3.7.

### Table 9 Results for large-scale and small budget

| Solver   | Solved | #100 | #100 | \(T_{\text{mean}}\) | # of anomalies | #n | #t | #l | Mean efficiency in % | For cost measure |
|----------|--------|------|------|-----------------|----------------|----|----|----|-------------------|----------------|
|          |        |      |      |                 |                |    |    |    |                   |                |
| LMLS4 lmtr4 | 101     | 47   | 44   | 265873          | 47             | 17 | 1  |     | 48                | 45             |
| LMLS3 lmtr3 | 97      | 22   | 19   | 242590          | 57             | 10 | 2  |     | 43                | 42             |
| LMLS2 lmtr2 | 94      | 11   | 10   | 262404          | 60             | 8  | 4  |     | 43                | 41             |
| LMLS1 lmtr1 | 86      | 22   | 21   | 302069          | 60             | 14 | 6  |     | 37                | 32             |
| FMINUNC1 func1 | 81     | 36   | 35   | 197750          | 60             | 24 | 1  |     | 34                | 37             |

### Table 10 Results for large-scale and budget

| Solver   | Solved | #100 | #100 | \(T_{\text{mean}}\) | # of anomalies | #n | #t | #l | Mean efficiency in % | For cost measure |
|----------|--------|------|------|-----------------|----------------|----|----|----|-------------------|----------------|
|          |        |      |      |                 |                |    |    |    |                   |                |
| LMLS3 lmtr3 | 128     | 32   | 29   | 313348          | 0              | 38 | 0  |     | 57                | 55             |
| LMLS4 lmtr4 | 128     | 49   | 47   | 325880          | 0              | 38 | 0  |     | 60                | 55             |
| LMLS2 lmtr2 | 120     | 17   | 15   | 306190          | 0              | 46 | 0  |     | 53                | 49             |
| FMINUNC1 func1 | 104    | 37   | 36   | 236336          | 0              | 61 | 1  |     | 42                | 47             |
| LMLS1 lmtr1 | 95      | 18   | 17   | 332592          | 0              | 71 | 0  |     | 39                | 33             |
Fig. 11  Details as in Fig. 4
A new limited memory method for unconstrained nonlinear least squares

Fig. 12 Details as in Fig. 4

Funding Open access funding provided by University of Vienna.

Declarations

Conflict of interest The authors declare that they have no conflict of interest

Human and animal rights This study does not contain any studies with human participants or animals performed by any of the authors.

Open Access This article is licensed under a Creative Commons Attribution 4.0 International License, which permits use, sharing, adaptation, distribution and reproduction in any medium or format, as long as you give appropriate credit to the original author(s) and the source, provide a link to the Creative Commons licence, and indicate if changes were made. The images or other third party material in this article are included in the article’s Creative Commons licence, unless indicated otherwise in a credit line to the material. If material is not included in the article’s Creative Commons licence and your intended use is not permitted by statutory regulation or exceeds the permitted use, you will need to obtain permission directly from the copyright holder. To view a copy of this licence, visit http://creativecommons.org/licenses/by/4.0/.

References

Ahookhosh M, Amini K (2011) An efficient nonmonotone trust-region method for unconstrained optimization. Numer Algo 59:523–540
Ahookhosh M, Esmaeili H, Kimiaei M (2013) An effective trust-region-based approach for symmetric nonlinear systems. Int J Comput Math 90:671–690
Ahookhosh M, Amini K, Kimiaei M (2015) A globally convergent trust-region method for large-scale symmetric nonlinear systems. Number Func Anal Opt 36:830–855
Amini K, Shiker MAK, Kimiaei M (2016) A line search trust-region algorithm with nonmonotone adaptive radius for a system of nonlinear equations. 4OR-Q J Oper Res 14:133–152
Amini K, Esmaeili H, Kimiaei M (2016) A nonmonotone trust-region-approach with nonmonotone adaptive radius for solving nonlinear systems. IJNAO 6
Bellavia S, Macconi M, Morini B (2004) STRSCNE: a scaled trust-region solver for constrained nonlinear equations. Comput Optim Appl 28:31–50
Bellavia S, Macconi M, Pieraccini S (2012) Constrained dogleg methods for nonlinear systems with simple bounds. Comput Optim Appl 53:771–794
Conn AR, Gould NIM, Toint Ph L (2000) Trust region methods. Society for industrial and applied mathematics
Deng NY, Xiao Y, Zhou FJ (1993) Nonmonotonic trust region algorithm. J Optim Theory Appl 76:259–285
Dennis JE Jr, Moré JJ (1977) Quasi-newton methods, motivation and theory. SIAM Rev 19:46–89
Dennis JE Jr, Walker HF (1981) Convergence theorems for least-change secant update methods. SIAM J Numer Anal 18:949–987
Deuflhard P (2011) Newton methods for nonlinear problems. Springer, Berlin Heidelberg
Dolan ED, Moré JJ (2002) Benchmarking optimization software with performance profiles. Math Prog 91:201–213
Esmaeili H, Kimiaei M (2014) An efficient adaptive trust-region method for systems of nonlinear equations. Int J Comput Math 92:151- [object Object]
Esmaeili H, Kimiaei M (2014) A new adaptive trust-region method for system of nonlinear equations. Appl Math Model 38:3003–3015
Esmaeili H, Kimiaei M (2015) A trust-region method with improved adaptive radius for systems of nonlinear equations. Math Meth Oper Res 83:109–125
Fan J (2006) Convergence rate of the trust region method for nonlinear equations under local error bound condition. Comput Optim Appl 34:215–227
Fan J, Pan J (2009) An improved trust region algorithm for nonlinear equations. Comput Optim Appl 48:59–70
Fan J, Pan J (2010) A modified trust region algorithm for nonlinear equations with new updating rule of trust region radius. Int J Comput Math 87:3186–3195
Grippo L, Sciandrone M (2007) Nonmonotone derivative-free methods for nonlinear equations. Comput Optim Appl 37:297–328
Grippo L, Lampariello F, Lucidi S (1986) A nonmonotone line search technique for newton’s method. SIAM J Numer Anal 23:707–716
Kimiaei M (2020) Line search in noisy unconstrained black box optimization. http://www.optimization-online.org/DB_HTML/2020/09/8007.html
Kimiaei M (2016) A new class of nonmonotone adaptive trust-region methods for nonlinear equations with box constraints. Calcolo 54:769–812
Kimiaei M (2017) Nonmonotone self-adaptive levenberg–marquardt approach for solving systems of nonlinear equations. Number Func Anal Opt 39:47–66
Kimiaei M, Neumaier A (2019) Testing and tuning optimization algorithm. Preprint, Vienna University, Fakultät für Mathematik, Universität Wien, Oskar-Morgenstern-Platz 1, A-1090 Wien, Austria
Kimiaei M, Neumaier A (2020) Efficient global unconstrained black box optimization. http://www.optimization-online.org/DB_HTML/2018/08/6783.html
Liu DC, Nocedal J (1989) On the limited memory BFGS method for large scale optimization. Math Program 45:503–528
Lukšan L, Matonoha C, Vlček J (2018) Problems for nonlinear least squares and nonlinear equations. Technical report V-1259, ICS CAS
Moré JJ, Wild SM (2009) Benchmarking derivative-free optimization algorithms. SIAM J Optim 20:172–191
Nazareth L (1979) A relationship between the BFGS and conjugate gradient algorithms and its implications for new algorithms. SIAM J Numer Anal 16:794–800
Nielsen HB (2012) immoptibox – a matlab toolbox for optimization and data fitting. version 2.2
Nocedal J (1992) Theory of algorithms for unconstrained optimization. Acta Numer 1:199–242
Nocedal J, Wright SJ (1999) eds. Numerical optimization, Springer-Verlag
Nowak U, Weimann L (1990) A family of newton codes for systems of highly nonlinear equations. Technical report TR 91–10, Zuse Institute Berlin (ZIB)
Ortega JM, Rheinboldt WC (2000) Iterative solution of nonlinear equations in several variables. Society for industrial and applied mathematics
Schnabel RB (1989) Sequential and parallel methods for unconstrained optimization. In: Iri M, Tanabe K (eds) Mathematical programming, recent developments and applications. Kluwer Academic Publishers, Netherlands, pp 227–261
Sorber L, Barel MV, Lathauwer LD (2012) Unconstrained optimization of real functions in complex variables. SIAM J Optim 22:879–898
Yu Z, Pu D (2008) A new nonmonotone line search technique for unconstrained optimization. J Comput Appl Math 219:134–144
Yuan Y (2011) Recent advances in numerical methods for nonlinear equations and nonlinear least squares. Numer Algebra Control Optim 1:15–34

Publisher’s Note Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.