Soft Computing (2022) 26:465–490
https://doi.org/10.1007/s00500-021-06415-8

FOUNDATIONS

A new limited memory method for unconstrained nonlinear least squares

Morteza Kimiaei[1] · Arnold Neumaier[1]

Accepted: 12 October 2021 / Published online: 13 December 2021

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

$$\begin{array}{ll} \min & f(x):=\frac{1}{2}\|E(x)\|_2^2 \\ \text{s.t.} & x \in \mathbb{R}^n, \end{array} \tag{1}$$

with high-dimensional $x \in \mathbb{R}^n$ and continuously differentiable $E : \mathbb{R}^n \to \mathbb{R}^r$ $(r \geq n)$, 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:

The first author acknowledges the financial support of the Doctoral Program *Vienna Graduate School on Computational Optimization* (*VGSCO*) funded by the Austrian Science Foundation under Project No W1260-N35.

✉ Morteza Kimiaei
kimiaeim83@univie.ac.at
http://www.mat.univie.ac.at/~kimiaei/

Arnold Neumaier
Arnold.Neumaier@univie.ac.at
http://www.mat.univie.ac.at/~neum/

[1] Fakultät für Mathematik, Universität Wien, Oskar-Morgenstern-Platz 1, A-1090 Wien, Austria

- 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 non-singularity 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 Weimann (1990) and **MINFNCG** (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 Kimiaei (2014b, a, 2015); Fan (2006); Fan and Pan (2009, 2010); Kimiaei (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 Sciandrone (2007); Kimiaei (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 surrogate objective function

$$\mathcal{Q}(p) := \frac{1}{2}\|E + Jp\|^2 := f + p^T g + \frac{1}{2}(Jp)^T Jp, \tag{2}$$

where $f := f(x)$, $E := E(x)$, $J := J(x)$, $g := g(x) := J^T E$. We denote by $A_{:k}$ the kth column of a matrix A.

A trust region method finds a minimizer of the constrained problem

$$\begin{array}{l} \min\ \mathcal{Q}(p) \\ \text{s.t}\ \ p \in \mathbb{R}^n \ \text{ and } \ \|p\| \le \Delta, \end{array} \tag{3}$$

whose constraint restricts feasible points by the **trust region radius** $\Delta > 0$. This problem is called the **trust region subproblem**. Given a solution p of (3), we define the **actual reduction** in the objective function by

$$df := f - f(x + p) \tag{4}$$

and the **predicted reduction** in the model function by

$$dq := \mathcal{Q}(0) - \mathcal{Q}(p). \tag{5}$$

What constitutes an agreement between the actual and predicted reduction around the current iterate x must be measured by the **monotone trust region ratio**

$$\rho := \frac{df}{dq}. \tag{6}$$

If such an agreement is good according to a heuristic formula discussed in Sect. 2.2, the iteration is said **successful**, $x + p$ is accepted as a new point, said a **best point**, and the radius is expanded; otherwise, the iteration is said **unsuccessful** and so the radius is reduced.

2.1 A new subspace Gauss–Newton method

In this subsection, we have two goals: estimating the Jacobian matrix and constructing a Gauss-Newton direction in a subspace with random basis indices.

Let $m_{\rm sn}$ be the subspace dimension. The Jacobian matrix in a subspace with random basis indices is estimated by a new **subspace random finite difference** called **SRFD** using the following steps:

(1) At first, an initial subspace basis indices set is a random subset of $\{1, \ldots, n\}$ consisting of $m_{\rm sn}$ members and its complementary is $S^c := \{1, \ldots, n\} \setminus S$.

(2) Next, if the complementary of old subspace basis indices set $S^c_{\rm old}$ is not empty, a new index set $\mathcal{I}$ needs to be identified before a new subspace basis indices set is determined. In this case, if $\mathcal{I}$ consists of at least $m_{\rm sn}$ members, $\mathcal{I}$ is a random subset of $\{1, \ldots, m_{\rm sn}\}$ with the $|S^c_{\rm old}|$ members; otherwise, it is a permutation of $\{1, \ldots, |S^c_{\rm old}|\}$. Then, a new subspace basis indices set is determined by $S := S^c_{\rm old}(\mathcal{I})$ and its complementary is found by $S^c := S^c_{\rm old} \setminus S$. But if $S^c_{\rm old}$ 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.

(3) For any $i \in S$,

- the step size is computed by

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

  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 ith 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

$$(p_{\rm sn})_i := 0 \ \text{ for } i \in S^c \text{ and } \ (p_{\rm sn})_S := -(J^T_{:S} J_{:S})^{-1} J^T_{:S} E. \tag{7}$$

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 Δ 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 before a successful iteration is found. This choice is based on an estimated increase in f defined below which is updated according to whether a decrease in f is found or not. It helps us to generate a somewhat strong non-monotone term when a decrease in f is not found and a somewhat weak non-monotone term otherwise. Somewhat strong non-monotone terms increase the chance of finding a point with better function value or at least a point with a little progress in the function value instead of solving trust region subproblems with high computational costs.

We denote by X a list of best points and by F a list of corresponding function values. Let $m_{\rm rs}$ be the maximum number of good points saved in X. In order to update X and F, we use **updateXF**. Here we describe how to work it. If $m_{\rm rs}$ is not exceeded, points with good function values are saved in X and their function values in F. Otherwise, the worst point and its function value are found and replaced by the best point and its function value, respectively.

Let $\gamma_t \in (0,1)$, $\underline{\gamma} \in (0,1)$, $\gamma^{\rm init} > 0$, and $\overline{\gamma} > 1$ be the tuning parameters and let

$$f^k_{\max} := \max_{i=1:m_{\rm rs}} \{F^k_i\} \quad \text{for all } k = 0,1,2,\cdots. \tag{8}$$

Before a new non-monotone term is constructed, an estimated increase in f needs to be estimated by

$$\delta^k_f := \begin{cases} \gamma^{\rm init}|f^0| & \text{if } k=0,\ f^0 \in (0,\infty),\\ 1 & \text{if } k=0,\ f^0 \in \{-\infty,0,\infty\},\\ \dfrac{1}{\overline{\gamma}}(f^{k-1} - f(x^{k-1}+p^{k-1})) & \text{if } k \ge 1,\ f(x^{k-1}+p^{k-1}) < f^{k-1},\\ \max(\overline{\gamma}\delta^{k-1}_f, \underline{\gamma}(|f(x^{k-1}+p^{k-1})| + |f^k_{\max}|)) & \text{if } k \ge 1,\ f(x^{k-1}+p^{k-1}) \ge f^{k-1}. \end{cases} \tag{9}$$

Accordingly, the new non-monotone formula is defined by

$$f^k_{\rm nm} := \begin{cases} f^0 & \text{if } k=0,\\ f^k + \delta^k_f & \text{if } k \ge 1 \end{cases} \tag{10}$$

and the new adaptive radius is constructed by

$$\Delta^k_{\rm nm} := \lambda^k \sqrt{f^k_{\rm nm}}, \tag{11}$$

where $\Delta^0_{\rm nm} > 0$ is a tuning parameter and λ^k is updated according to

$$\lambda^k := \begin{cases} \sigma_1 \lambda^{k-1}, & \text{if } \rho^{k-1}_{\rm nm} < \gamma_t,\\ \min(\overline{\lambda}, \max(\sigma_2\lambda^{k-1}, \underline{\lambda})), & \text{otherwise.} \end{cases} \tag{12}$$

Here $\lambda^0 > 0$, $0 < \sigma_1 < 1 < \sigma_2$, and $\overline{\lambda} > \underline{\lambda} > 0$ are the tuning parameters and the new non-monotone trust region ratio is defined by

$$\rho^{k-1}_{\rm nm} := \frac{f^{k-1}_{\rm nm} - f(x^{k-1}+p^{k-1})}{\widetilde{\mathcal{Q}}^{k-1}(0) - \widetilde{\mathcal{Q}}^{k-1}(p^{k-1})}, \tag{13}$$

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

$$\begin{aligned} \min\ \widetilde{\mathcal{Q}}^{k-1}(p) &:= \frac{1}{2}\|E^{k-1} + J^{k-1}_{:S}p\|^2 := f^{k-1} + p^T g^{k-1}_S \\ &\quad + \frac{1}{2}(J^{k-1}_{:S}p)^T J^{k-1}_{:S} p \\ \text{s.t}\ \ p \in \mathbb{R}^{m_{\rm sn}} &\ \text{ and }\ \|p\| \le \Delta^{k-1}_{\rm nm} \end{aligned} \tag{14}$$

with $f^{k-1} := f(x^{k-1})$, $E^{k-1} := E(x^{k-1})$, $J^{k-1}_{:S} := J_{:S}(x^{k-1})$, and $g^{k-1}_S := (J^{k-1}_{:S})^T E^{k-1}$.

2.3 A subspace dogleg algorithm

We define the **Cauchy step** by

$$\begin{aligned} p_{\rm c} &:= -t^* g_S,\\ t^* &:= \operatorname{argmin}\{\widetilde{\mathcal{Q}}(-tg_S) \mid t \ge 0,\ \|tg_S\| \le \Delta_{\rm nm}\}. \end{aligned} \tag{15}$$

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

$$\|p\| \le \Delta_{\rm nm} \ \text{ and } \ \widetilde{\mathcal{Q}}(p) \le \widetilde{\mathcal{Q}}(p_{\rm c}) \tag{16}$$

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 $\widetilde{\mathcal{Q}}$ is reduced by (15) if $dq_{\rm sn} := \widetilde{\mathcal{Q}}(0) - \widetilde{\mathcal{Q}}(p_{\rm sn}) > 0$. **subDogleg** first identifies whether $dq_{\rm sn} > 0$ or not. Then we have one of the following cases:

CASE 1. If $dq_{\rm sn} > 0$, the **scaled steepest descent step**

$$p_{\rm sd} := -\frac{g_S^T g_S}{(J_S g_S)^T (J_S g_S)} g_S \tag{17}$$

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

$$p_{\text{dg}}{:=}p_{\text{sd}} + t(p_{\text{sn}} - p_{\text{sd}}); \tag{18}$$

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

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

Otherwise, t is computed by

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

e.g., see Nielsen 2012.

Case 2. If $dq_{\text{sn}} \le 0$, the model function $\widetilde{\mathcal{Q}}$ 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 $p{:=}\Delta_{\text{nm}}(p_{\text{sd}}/\|p_{\text{sd}}\|)$ 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 $x^1, \ldots, x^{m_{\text{rs}}}$ be the m_{rs} best point stored in X. Then a point in the affine space spanned by such points has the following form

$$x_z{:=}Xz, \quad z \in \mathbb{R}^{m_{\text{rs}}}, \quad e^T z = 1, \tag{21}$$

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

$$\begin{aligned} &\min \ \frac{1}{2}\|Bz\|_2^2 \\ &\text{s.t.} \ \ e^T z = 1. \end{aligned} \tag{22}$$

This is a quality constrained convex quadratic problem in m_{rs} 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 $z{:=}Zy$, define $a^T{:=}e^T Z$, and obtain the m_{rs}-dimensional minimal norm linear feasibility problem

$$\begin{aligned} &\min \ \frac{1}{2}\|y\|_2^2 \\ &\text{s.t.} \ \ a^T y = 1 \end{aligned} \tag{23}$$

whose solution is $y{:=}a/\|a\|_2$. Hence, a new trial point for the next algorithm is

$$x^{\text{trial}}{:=}x_z{:=}XR^{-1}a/\|a\|_2. \tag{24}$$

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, δ_f, and f_{nm} as input and uses the following tuning parameter:

$\underline{\gamma} \in (0, 1)$ (tiny factor for adjusting δ_f),
$\overline{\gamma} > 1$ (parameter for expanding δ_f),
γ_s (tiny parameter for the finite difference step size),
m_{rs} (memory for affine space),
$0 < \gamma_r < 1$ (tiny parameter for adjusting the scaled random directions).

It returns a new value of δ_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

```
   QR factorization of the matrix $B$
1: Make a QR factorization for the matrix $B$ and get the square upper
   triangular matrix $R$.
   Computation of a new trial point
2: Compute the new trial point $x^{\text{trial}}$ by (24).
3: Compute $E^{\text{trial}}{:=}E(x^{\text{trial}})$ and set $f^{\text{trial}}{:=}0.5\|E^{\text{trial}}\|^2$.
   Check whether the function value is improved or not?
4: if $f^{\text{trial}} \le f_{\text{nm}}$ then            ▷ 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 updateXF.
7:     Compute $S$ and $J_S{:=}J_{:S}(x)$ by SRFD.
8: end if
9: Update $\delta_f$ by (9) and compute $f_{\text{nm}}$ by (10).
```

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 $\rho_{\rm nm} \ge \gamma_t$ 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 (`nfmax`). It uses the following tuning parameters:

$m_{\rm sn}$ (subspace dimension),
$\gamma_t \in (0, 1)$ (parameter for trust region),
$0 < \sigma_1 < 1 < \sigma_2$ (parameters for updating λ),
$\gamma^{\rm init}$ (parameter for updating the initial δ_f),
$\underline{\gamma} \in (0, 1)$ (tiny factor for adjusting δ_f),
$\overline{\gamma} > 1$ (parameter for expanding δ_f),
$\underline{\lambda}$ (lower bound for λ),
$\overline{\lambda}$ (upper bound for λ),
γ_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 $x^{\rm best}$ 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 $\lambda^0 \in (\underline{\lambda}, \overline{\lambda})$.
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 $g_S^0{:=}J_{:S}^T(x^0)E^0$ and $\delta_f^0$ by (9).
5: for $\ell = 0, 1, 2, \cdots$ do
     Compute the $\ell$th search direction $p^\ell$
6:   Compute the subspace Gauss-Newton direction $p^\ell{:=}p_{\rm sn}^\ell$ by (7).
7:   if $\|p_{\rm sn}^\ell\| > \Delta_{\rm nm}^\ell$ then
8:     Obtain an approximated solution $p^\ell$ of (14) by subDogleg.
         subDogleg is a variant of dogleg.m, available at
             http://www.math.ubc.ca/~loew/m604/mfiles.htm
         but with the difference that
         • $t$ is recomputed by (20) when $t$ computed by (19) is not positive,
         • it can handle problems in high dimensions,
         • the concavity of $\widetilde{Q}$ is ignored,
         • it is restricted to the subspace with random basis indices.
9:   end if
     Enforcing the descent condition $(g^\ell)^T p^\ell < 0$ if it needs
10:  if $(g^\ell)^T p^\ell \ge 0$ then                    ▷ Due to rounding errors
11:    Find ind:=$\{i \mid g_i^\ell p_i^\ell > 0\}$.
12:    Set $p_i^\ell{:=} - p_i^\ell$ for $i \in$ ind.
13:  end if
     Computation of a new trial point and its function value
14: Compute $x^{\rm trial}{:=}x^\ell+p^\ell$, $E^{\rm trial}{:=}E(x^{\rm trial})$, and $f^{\rm trial}{:=}0.5\|E^{\rm trial}\|^2$.
     Stopping test
15:  if nfmax is exceeded then
16:    LMLS ends, resulting $x^{\rm best} = x^\ell$.
17:  end if
     Identify whether the $\ell$th iteration is successful or not
18:  Compute $\rho_{\rm nm}^\ell$ by (13).
19:  if $\rho_{\rm nm}^\ell < \gamma_t$ then                ▷ Unsuccessful iteration
20:    Call BroydenLike with the goal of satisfying $f^{\rm trial} \le f_{\rm nm}$.
21:    if $f^{\rm trial} > f_{\rm nm}$ then     ▷ BroydenLike cannot decrease $f$
22:      Update $\delta_f^{\ell+1}$ by (9) and $f_{\rm nm}^{\ell+1}$ by (10).
23:      Reduce $\lambda^{\ell+1}$ to (12) and the radius $\Delta_{\rm nm}^{\ell+1}$ to (11).
24:    end if
25:  else                                          ▷ Successful iteration
26:    Set $x^{\ell+1}{:=}x^{\rm trial}$, $f^{\ell+1}{:=}f^{\rm trial}$, and $E^{\ell+1}{:=}E^{\rm trial}$.
27:    Compute the new indices set $S$ and $J_{:S}(x^{\ell+1})$ by SRFD.
28:    Compute $g_S^{\ell+1}{:=}J_{:S}(x^{\ell+1})^T E^{\ell+1}$.
29:    Update $\delta_f^{\ell+1}$ by (9) and $f_{\rm nm}^{\ell+1}$ by (10).
30:    Expand $\lambda^{\ell+1}$ to (12) and $\Delta_{\rm nm}^{\ell+1}$ to (11).
31:    Store $x^{\ell+1}$ in $X$ and $f^{\ell+1}$ in $F$ by updateXF.
32:  end if
33: end for
```

3 Numerical results

We updated the test environment constructed by Kimiaei and Neumaier (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.

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

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

Note that $\texttt{delta} = -1$ 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.

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

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

was used. Note that $\texttt{delta} = -1$ means that $\Delta_0 = 1$. According to our numerical results, **STRSCNE** was not sensitive for the initial radius; hence, the default was used for Δ_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
  and suggested by Deuflhard (2011) and written by Nowak and Weimann (1990). The default tuning parameters were used; only $\texttt{iopt.jacgen} = 2$, $\texttt{iopt.qrank1} = 1$, and $\texttt{wk.fmin} = \texttt{1e} - \texttt{50}$ were selected.
- **NLEQ2** is the same as **NLEQ1**; only $\texttt{iopt.qrank1} = 0$ was selected.

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

  https://www.tensorlab.net/

  and written by Sorber et al. (2012). The default parameters were used, except for m. **MINLBFGS1** and **MINLBFGS2** are **MINLBFGS** with $m = \min(n, 30)$ and $m = \min(n, 100)$, respectively.
- **MINLBFGSDL** is a L-BFGS with dogleg trust region algorithm with the option set $\texttt{MaxIter} = \texttt{nfmax}$, $\texttt{TolFun} = \texttt{1e} - \texttt{50}$, $\texttt{TolX} = \texttt{1e} - \texttt{50}$. The default parameters were chosen for $\texttt{PlaneSearch}$ and M. **MINLBFGSDL1**, **MINLBFGSDL2**, **MINLBFGSDL3**, and **MINLBFGSDL4** are **MINLBFGSDL** with

(1) $\texttt{PlaneSearch} = \texttt{false}$ and $M = \min(30, n)$,
(2) $\texttt{PlaneSearch} = \texttt{false}$ and $M = \min(100, n)$,
(3) $\texttt{PlaneSearch} = \texttt{true}$ and $M = \min(30, n)$,
(4) $\texttt{PlaneSearch} = \texttt{true}$ and $M = \min(100, n)$.

According to our results, **MINLBFGSDL1** was the best.

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

  https://www.tensorlab.net/

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

$$\texttt{MaxIter} = \texttt{nfmax};\ \ \texttt{TolFun} = \texttt{1e} - \texttt{50};\ \ \texttt{TolX} = \texttt{1e} - \texttt{50}.$$

  The other tuning parameter was $\texttt{Beta} \in \{\texttt{HS}, \texttt{HSm}, \texttt{PR}, \texttt{FR}, \texttt{PRm}, \texttt{SD}\}$. **MINFNCG1**, **MINFNCG2**, **MINFNCG3**, **MINFNCG4**, **MINFNCG5**, and **MINFNCG6** are **MINFNCG** with $\texttt{Beta} = \texttt{HS}$, $\texttt{Beta} = \texttt{HSm}$, $\texttt{Beta} = \texttt{PR}$, $\texttt{Beta} = \texttt{FR}$, $\texttt{Beta} = \texttt{PRm}$, and $\texttt{Beta} = \texttt{SD}$, 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

  The following options were used

$$\texttt{iniscalx} = 0;\ \ \texttt{rescalx} = 0;\ \ \texttt{xthrsh} = \text{ones}(n, 1);\ \ \texttt{xtol} = \texttt{1.e} - \texttt{50};\ \ \texttt{ftol} = \texttt{1.e} - \texttt{50};\ \ \texttt{kmax} = \texttt{nfmax};\ \ \texttt{printmon} = 2;\ \ \texttt{printsol} = 1;\ \ \texttt{fid} = 1;$$

`numdif` $= 1$;
`lambda0` $=$ `eps`; `lambdamin` $=$ `1e` $-$ `50`;
`ftol` $=$ `1.e` $-$ `50`.

- **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,

  https://de.mathworks.com/help/optim/ug/lsqnonlin.html

  is a nonlinear least squares solver with the following options:

  `options = optimoptions(@lsqnonlin,'Algorithm', 'levenberg-marquardt', 'FiniteDifferenceType','forward', 'MaxIter', Inf,'MaxFunEvals', nfmax, 'TolX', 0,'Specify ObjectiveGradient','false').`
- **LSQNONLIN2**, obtained from the MATLAB Optimization Toolbox at,

  https://de.mathworks.com/help/optim/ug/lsqnonlin.html

  is a nonlinear least squares solver with the following options:

  `options = optimoptions(@lsqnonlin,'Algorithm', 'trust-region reflective', 'FiniteDifferenceType','forward', 'MaxIter', Inf,'MaxFunEvals', nfmax, 'TolX', 0,'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

  http://www2.imm.dtu.dk/projects/immoptibox/

  The following option was used

  $$\begin{aligned}\texttt{opts} &= [\Delta_0, \texttt{tolg}, \texttt{tolx}, \texttt{tolr}, \texttt{maxeval}]\\ &= [\Delta_0, \texttt{1e}-\texttt{50}, \texttt{1e}-\texttt{50}, \texttt{1e}-\texttt{50}, \texttt{nfmax}].\end{aligned}$$

  DOGLEG1, **DOGLEG2**, **DOGLEG3**, and **DOGLEG4** are **DOGLEG** with $\Delta_0 = 1$, $\Delta_0 = 10$, $\Delta_0 = 100$, and $\Delta_0 = 500$, respectively. The best version was **DOGLEG1**.

Unconstrained solvers:

- **FMINUNC**, obtained from the MATLAB Optimization Toolbox at

  https://ch.mathworks.com/help/optim/ug/fminunc.html,

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

  `opts = optimoptions(@fminunc),'Algorithm','quasi-newton', 'Display', 'Iter', 'MaxIter',Inf,'MaxFunEvals', nfmax, 'TolX', 0, '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

$$\boxed{\begin{array}{llll} m_{\rm nm} = 10; & m_{\rm rs} = 10; & \gamma_r = 10^{-30}; & \gamma_p = 5\varepsilon_m;\\ \gamma_s = \sqrt{\varepsilon_m}; & \gamma_t = 0.1; & \sigma_1 = 0.5; & \lambda^0 = 1;\\ \sigma_2 = 1; & \gamma^{\rm init} = 10^{-8}; & \gamma_m = 10^{-30}; & \underline{\lambda} = 10^{-4};\\ \Delta^0 = 10; & \Delta^{\min} = 10^{-6}; & \overline{\lambda} = 10^5. & \end{array}}$$

The remaining tuning parameter $m_{\rm sn}$ is varied in the experiment: **LMLS1**, **LMLS2**, **LMLS3**, and **LMLS4** are **LMLS** with $m_{\rm sn} = 3$, $m_{\rm sn} = \min(10, n)$, $m_{\rm sn} = \min(30, n)$, and $m_{\rm sn} = \min(100, n)$, 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 $\chi_i := (-1)^{i-1} 2/(2+i)$ for all $i = 1, \ldots, n$. This means that the initial point is chosen by $x_i^0 = \chi_i$ 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 $f^\ell := f(x^\ell + \chi)$ for all $\ell \geq 1$.

We denote `nf` and `msec` as the number of function evaluations and time in milliseconds, respectively. `nfmax` and `secmax` are the upper bounds for them, chosen as

$$\texttt{nfmax} \in \begin{cases} \{10n, 50n, 100n, 500n\} & \text{if } 1 \leq n \leq 100, \\ \{10n, 50n, 100n\} & \text{if } 101 \leq n \leq 1000 \\ \{10n, 100n\} & \text{if } 1001 \leq n \leq 10000 \end{cases}$$

and

$$\texttt{secmax} := \begin{cases} 300 & \text{if } 1 \leq n \leq 100, \\ 800 & \text{if } 101 \leq n \leq 10000. \end{cases}$$

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

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

then the problem is solved by the solver so. Otherwise, the problem is unsolved by it; either `nfmax` or `secmax` 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

$$e_{so} := \begin{cases} \dfrac{\min\limits_{s \in S} c_s}{c_{so}}, & \text{if the solver } so \text{ solves the problem,} \\ 0, & \text{otherwise,} \end{cases}$$

called the **efficiency** of the solver so with respect to this cost measure. Two cost measures `nf` 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 `nf/(best nf)` and `msec/(best 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 `nf/(best nf)` and `msec/(best msec)`; the percentage of problems solved within a factor τ 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 **LMLS1**, **LMLS2**, **LMLS3**, **LMLS4**, and solvers using quasi-Newton is shown in Subfigures (a) and (b) of Fig. 1, so that

- **LMLS4** using the full estimated Jacobian matrix is the best in terms of the number of solved problems and the `nf` efficiency;
- **LMLS3**, **LMLS2**, and **LMLS1** are more efficient than solvers using quasi-Newton approximation (**FMINUNC**, **FMINUNC1**, **MINFLBFGS1**, and **MINFLBFGSDL1**) in terms of the `nf` efficiency;
- **FMINUNC** and **MINFLBFGSDL1** are comparable with **LMLS3**—only for very large budget—in terms of the number of solved problems but **LMLS3**, **LMLS2**, and **LMLS1** are more efficient than **FMINUNC** and **MINFLBFGSDL1** in terms of the `nf` 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 **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, **LMLS4** is much more efficient and robust than **NATRLS1**, **NMPGTR2**, and **NATRN1** in terms of the number of solved problems and the `nf` efficiency.

We compare **LMLS4** with four famous solvers **LSQNONLIN1**, **CoDoSol1**, **NLEQ1**, and **DOGLEG1** shown in Subfigures (e) and (f) of Fig. 1. It is seen that **LMLS4** and **CoDoSol1** are the two best solvers in terms of the `nf`

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 `nf` efficiency;
- **LMLS2** is the second best solver in terms of the number of solved problems and the `nf` 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 `nf` efficiency; even it is more efficient than **NLEQ1** for very large budget in terms of the `nf` 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 with **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 `nf` efficiency, respectively;
- **LMLS4** is the best solver in terms of the number of solved problems; only **FMINUNC** is the best for large budget.

3.7 Large scale: $n \in [1001, 10000]$

In this subsection, we compare **LMLS1**, **LMLS2**, **LMLS3**, **LMLS4** using the estimated Jacobian matrices in a subspace with random basis indices with **FMINUNC1** using limited memory BFGS approximations.

In terms of the `nf` 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 `nf` ($e_{so} = 1 = 100\%$). !100 is the total number of problems solved for which the solver so was better than all other solvers with respect to `nf`.

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

- n indicates that `nf` $\geq$ `nfmax` was reached.
- t indicates that `sec` $\geq$ `secmax` 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 $\mathcal{P}$, ignoring the times for unsolved problems. It can be a good measure when solvers have approximately the same number of solved problems.

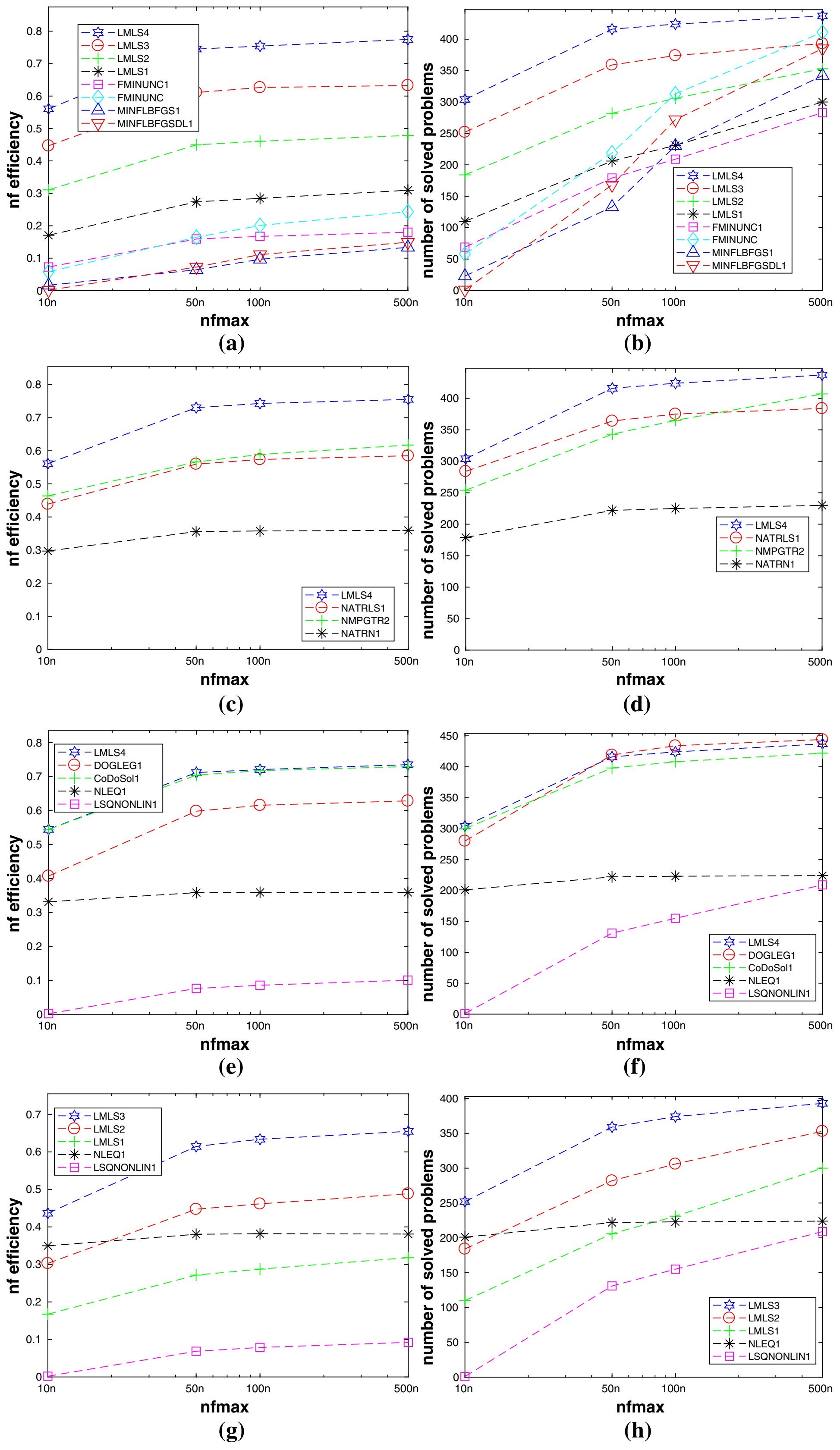

Fig. 1 Performance plots for small-scale problems. **a**–**b**: A comparison of limited memory solvers, **c**–**d**: A comparison among **LMLS** in a full subspace with random basis indices and solvers using other non-monotone and adaptive radius techniques, **e**–**f**: A comparison among **LMLS** in 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

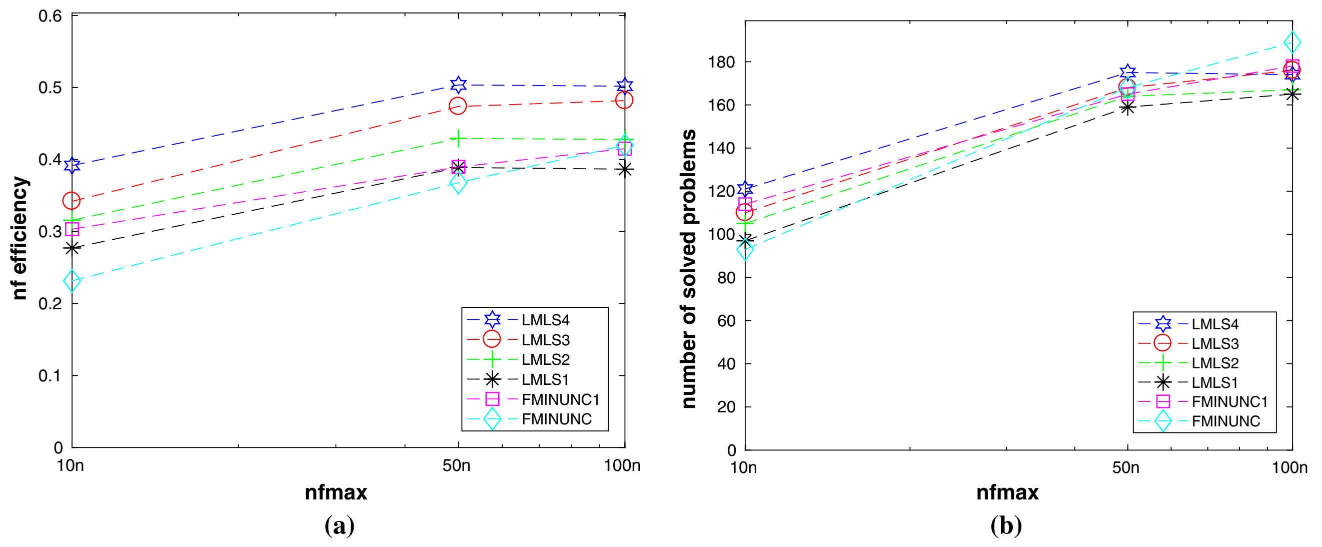

Fig. 2 **a**–**b**: Performance plots for medium-scale problems

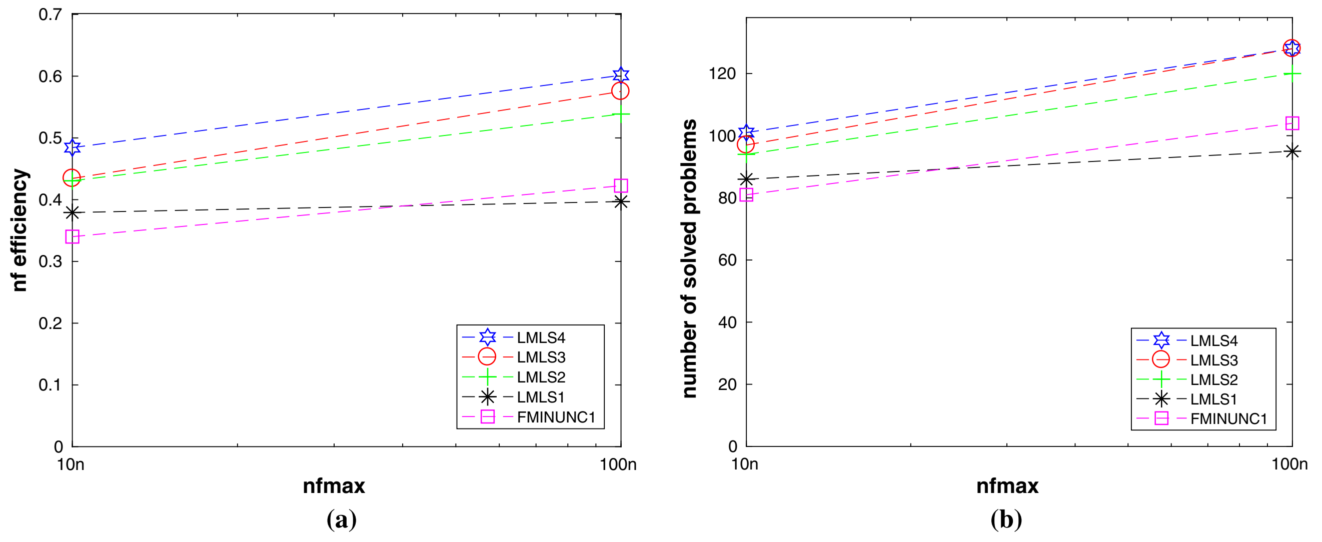

Fig. 3 **a**–**b**: Performance plots for large-scale problems

4.2 Tables and data/performance profiles for $1 \le n \le 100$

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

Stopping test: $q^{so} \le$ `1e-08`, `sec` $\le$ `300`, `nf` $\le$ `10*n`										
367 of 526 problems without bounds solved									Mean efficiency in %	
dim∈[1,100]						# of anomalies			For cost measure	
Solver		Solved	#100	!100	T_{mean}	#n	#t	#f	nf	msec
LMLS4	**lmtr4**	**304**	**204**	15	52	222	0	0	**53**	**46**
CoDoSol1	**codo1**	300	197	**30**	56	219	0	7	52	40
NATRLS1	**natrs1**	284	63	8	67	242	0	0	40	30
DOGLEG1	**dogleg1**	280	70	11	98	246	0	0	38	22
NMPGTR2	**nmpg2**	254	172	8	55	272	0	0	43	31
LMLS3	**lmtr3**	252	156	4	53	274	0	0	42	35
NLEQ1	**nleq1**	201	42	**30**	77	145	0	180	33	16
LMLS2	**lmtr2**	184	91	7	75	342	0	0	29	22
NATRN1	**natrn1**	179	54	11	89	347	0	0	26	18
LMLS1	**lmtr1**	110	47	20	112	416	0	0	16	10
STRSCNE1	**strs1**	70	69	0	**28**	107	0	349	13	7
FMINUNC1	**func1**	69	2	0	123	455	0	2	6	4
FMINUNC	**func**	58	2	0	139	468	0	0	5	4
MINFLBFGS1	**lbfgs1**	23	0	0	227	449	0	54	1	0
LSQNONLIN1	**lsqn1**	1	1	1	50	525	0	0	0	0
MINFLBFGSDL1	**minlbfgs1**	1	0	0	30	525	0	0	0	0
MINFNCG1	**MINFNCG1**	0	0	0	–	514	0	12	0	0

Table 3 Results for small-scale and medium budget

Stopping test: $q^{so} \le$ `1e-08`, `sec` $\le$ `300`, `nf` $\le$ `50*n`										
474 of 526 problems without bounds solved									Mean efficiency in %	
dim∈[1,100]						# of anomalies			For cost measure	
Solver		Solved	#100	!100	T_{mean}	#n	#t	#f	nf	msec
DOGLEG1	**dogleg1**	**419**	80	17	148	107	0	0	55	30
LMLS4	**lmtr4**	416	**228**	17	108	110	0	0	**66**	**59**
CoDoSol1	**codo1**	398	226	**52**	90	107	0	21	65	54
NATRLS1	**natrs1**	364	79	21	81	162	0	0	51	39
LMLS3	**lmtr3**	359	177	8	124	167	0	0	55	46
NMPGTR2	**nmpg2**	343	191	22	79	183	0	0	53	40
LMLS2	**lmtr2**	282	111	16	197	244	0	0	40	29
NATRN1	**natrn1**	222	63	17	117	304	0	0	31	22
NLEQ1	**nleq1**	222	42	30	77	122	0	182	35	18
FMINUNC	**func**	219	4	3	167	292	0	15	13	11
LMLS1	**lmtr1**	206	49	21	253	320	0	0	24	15
FMINUNC1	**func1**	179	8	7	177	345	0	2	13	10
MINFLBFGSDL1	**minlbfgs1**	168	0	0	286	358	0	0	5	5
MINFLBFGS1	**lbfgs1**	133	0	0	200	339	0	54	5	4
LSQNONLIN1	**lsqn1**	131	1	1	314	395	0	0	5	4
STRSCNE1	**strs1**	72	69	0	**28**	1	0	453	13	8
MINFNCG1	**MINFNCG1**	47	0	0	698	458	0	21	1	1

Table 4 Results for small-scale and large budget

Stopping test: $q^{so} \le$ `1e-08`, `sec` $\le$ `300`, `nf` $\le$ `100*n`

493 of 526 problems without bounds solved									Mean efficiency in %	
dim∈[1,100]						# of anomalies			For cost measure	
Solver		Solved	#100	!100	T_{mean}	#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	**natrs1**	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	**minlbfgs1**	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
MINFNCG1	**MINFNCG1**	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: $q^{so} \le$ `1e-08`, `sec` $\le$ `300`, `nf` $\le$ `500*n`

509 of 526 problems without bounds solved									Mean efficiency in %	
dim∈[1,100]						# of anomalies			For cost measure	
Solver		Solved	#100	!100	T_{mean}	#n	#t	#f	nf	msec
DOGLEG1	**dogleg1**	**444**	79	18	220	82	0	0	57	31
LMLS4	**lmtr4**	437	**229**	18	159	89	0	0	**68**	**58**
CoDoSol1	**codo1**	422	226	**54**	172	61	0	43	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	**minlbfgs1**	385	6	6	723	69	0	72	11	12
NATRLS1	**natrs1**	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
MINFNCG1	**MINFNCG1**	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

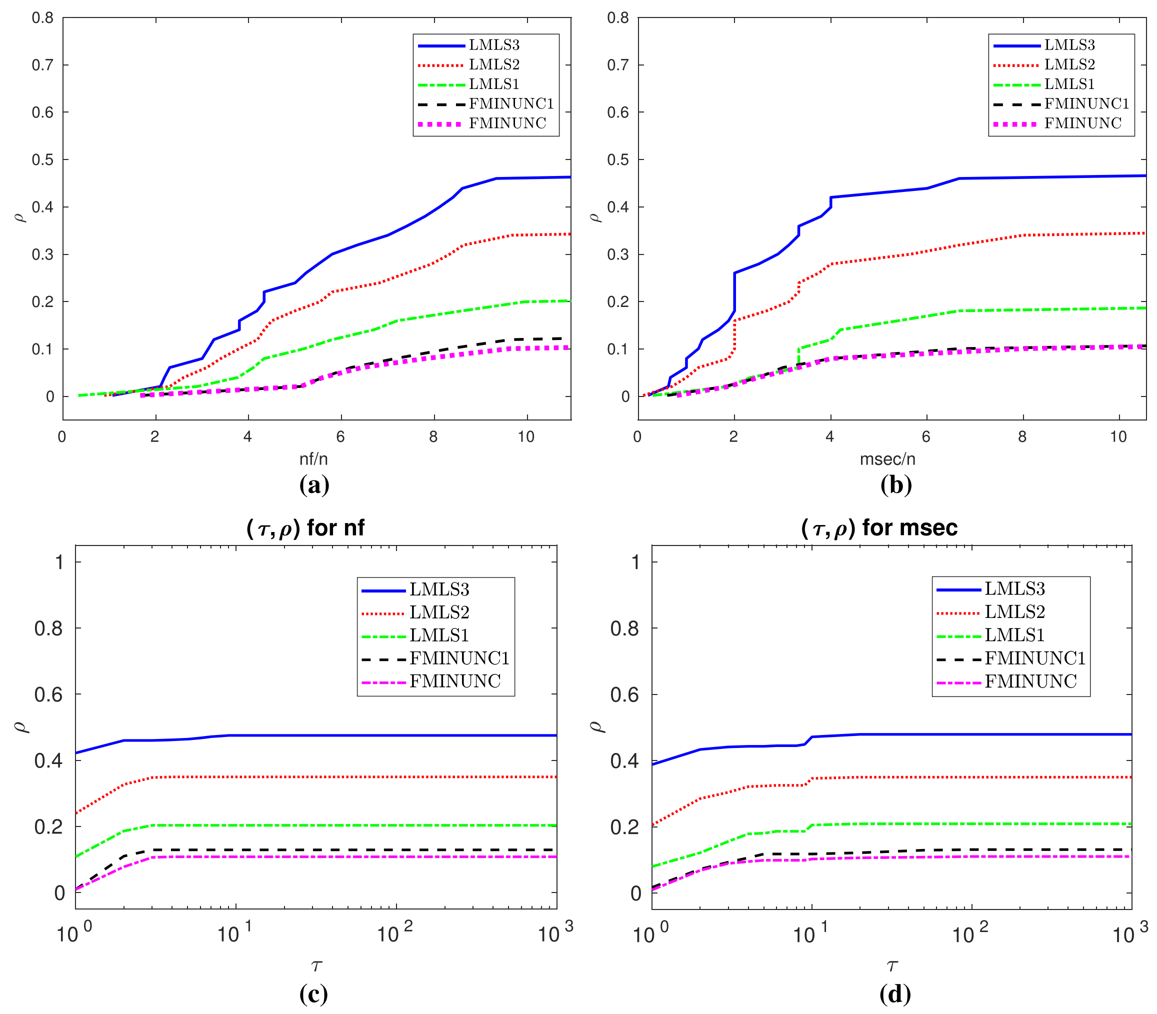

Fig. 4 **a** and **b**: Data profiles for nf/(best nf) and msec/(best msec), respectively. ρ designates the fraction of problems solved within the number of function evaluations and time in milliseconds used by the best solver. Problems solved by no solver are ignored. **c**–**d**: Performance profiles for nf/(best nf) and msec/(best msec), respectively. ρ 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

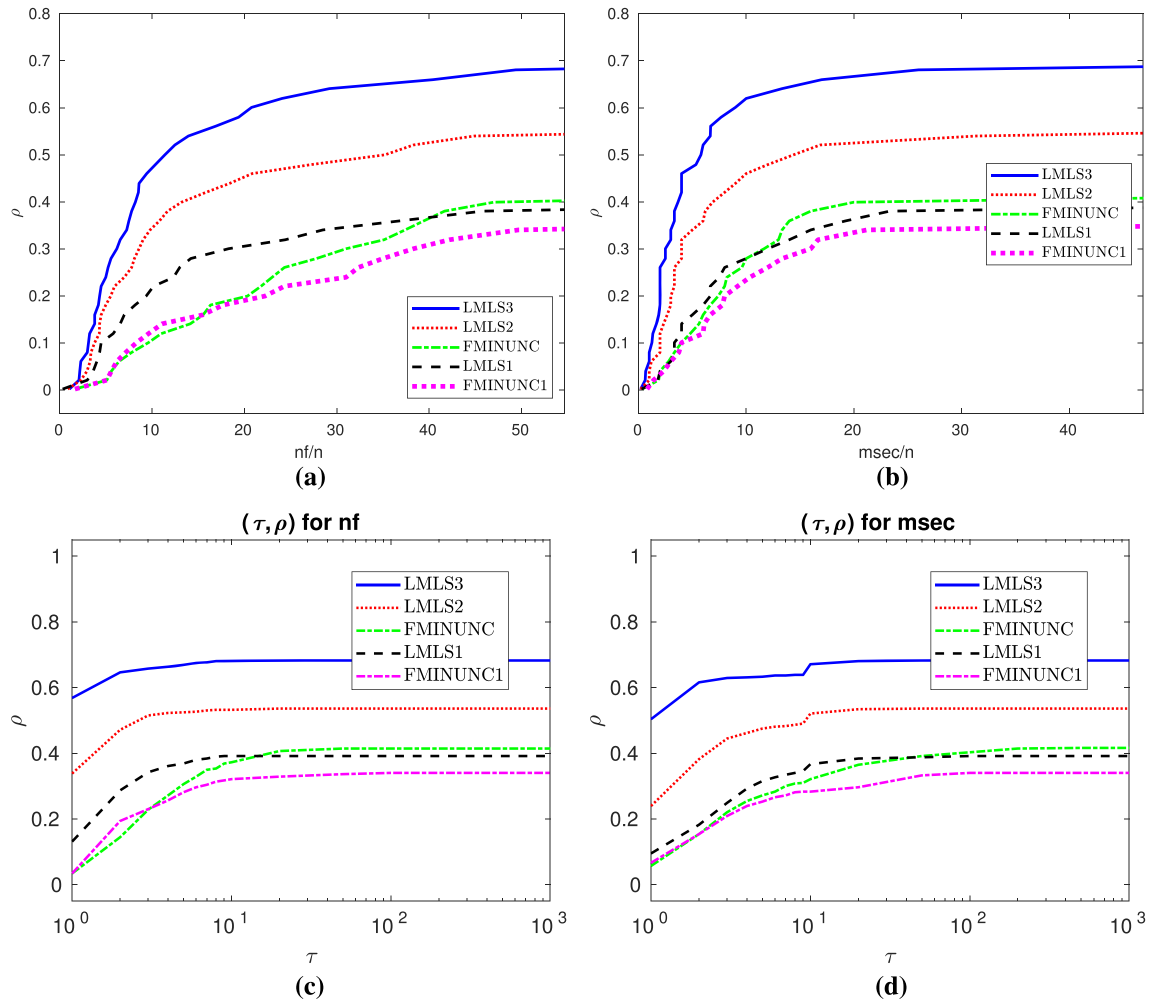

Fig. 5 Details as in Fig. 4

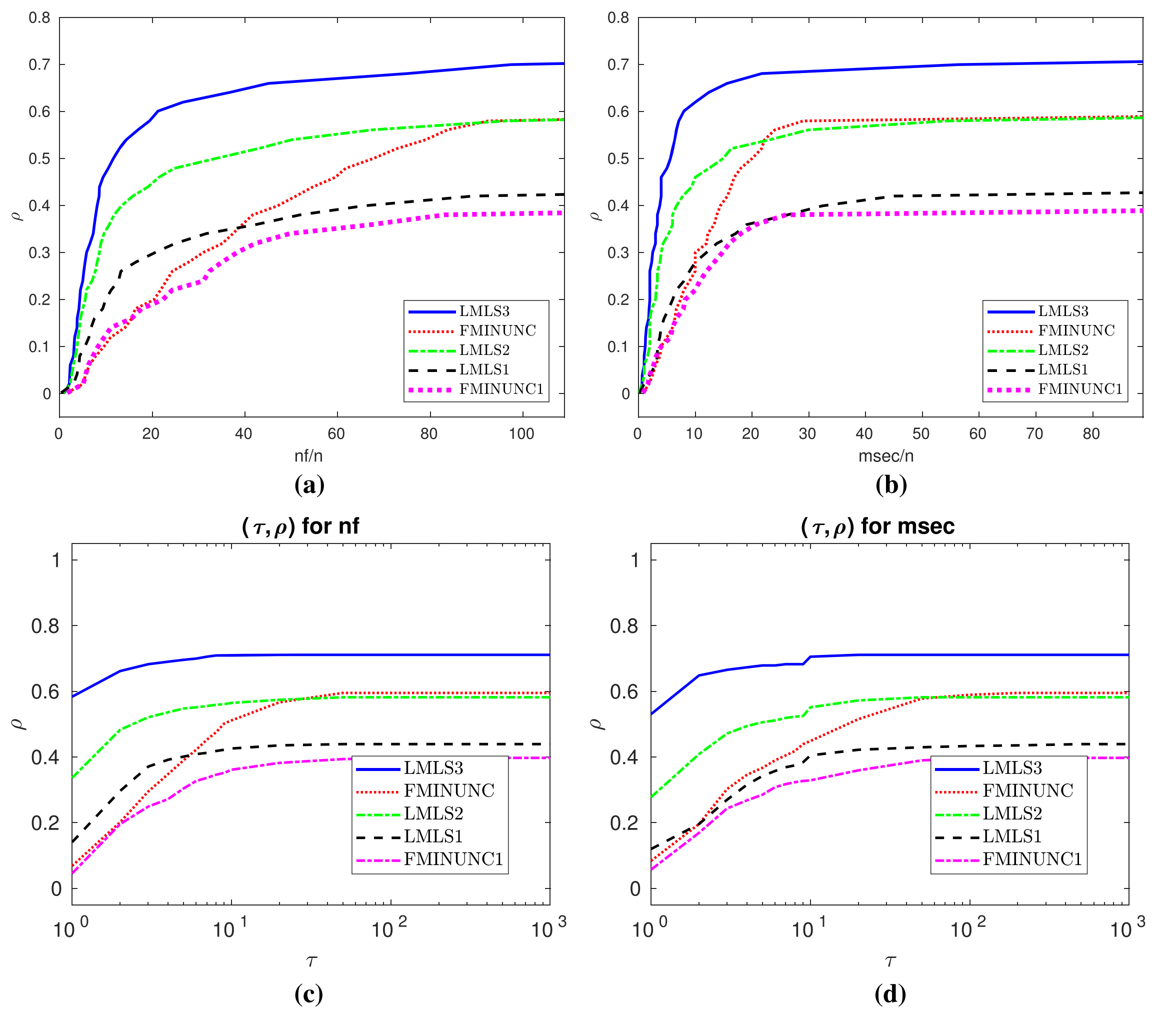

Fig. 6 Details as in Fig. 4

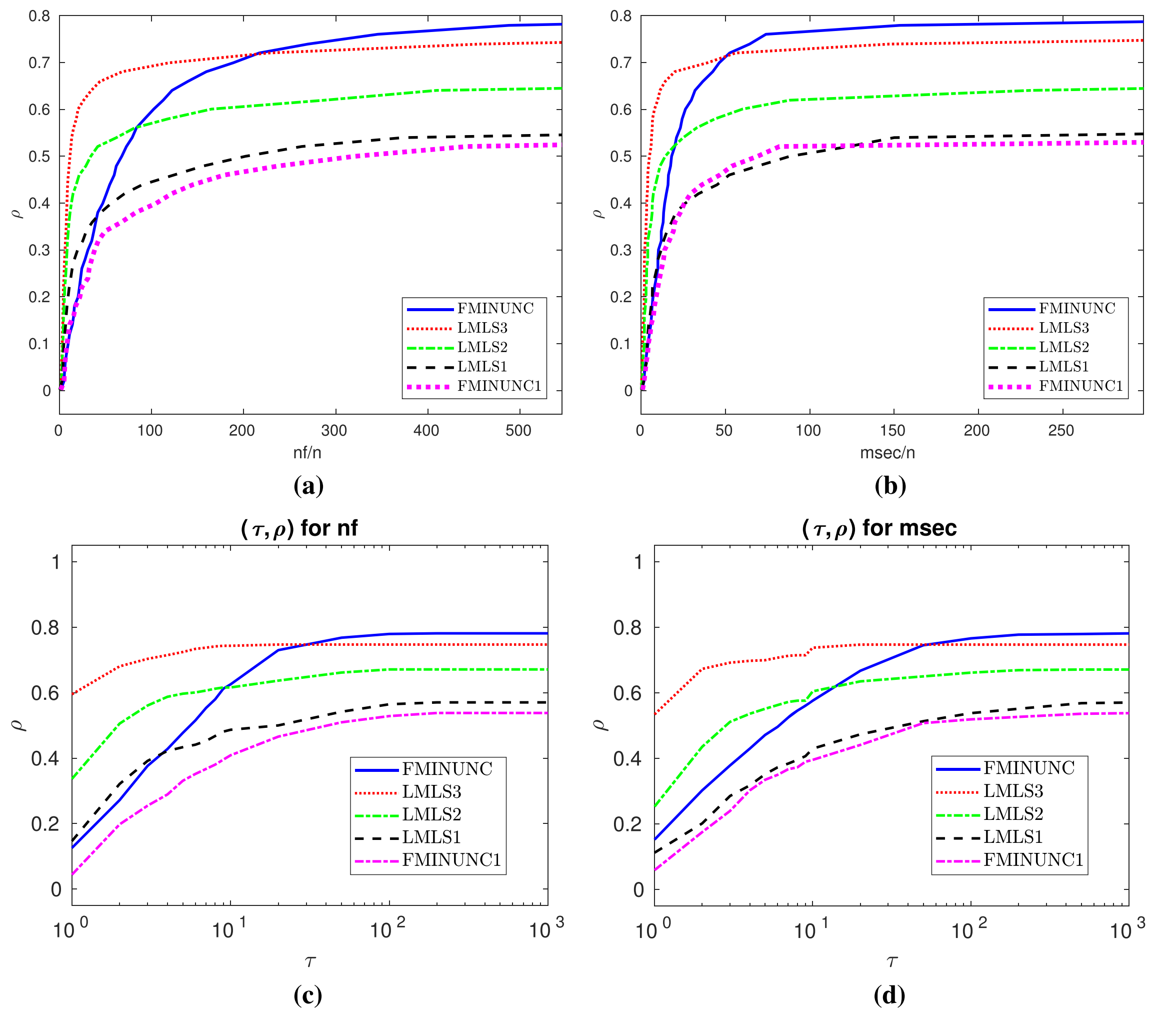

Fig. 7 Details as in Fig. 4

4.3 Tables and data/performance profiles for $101 \le n \le 1000$

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

Stopping test: $q^{so} \le$ `0.001`, `sec` $\le$ `800`, `nf` $\le$ `10*n`										
154 of 249 problems without bounds solved									Mean efficiency in %	
dim∈[101,1000]						# of anomalies			For cost measure	
Solver		Solved	#100	!100	T_{mean}	#n	#t	#f	nf	msec
LMLS4	**lmls4**	**119**	**49**	**48**	13738	129	1	0	**37**	**37**
FMINUNC1	**func1**	114	40	14	**11271**	133	2	0	29	35
LMLS3	**lmtr3**	110	30	28	15401	138	1	0	34	28
LMLS2	**lmtr2**	105	22	21	15006	143	1	0	31	20
LMLS1	**lmtr1**	97	15	15	18207	151	1	0	27	21
FMINUNC	**func**	93	26	0	13702	154	2	0	22	24

Table 7 Results for medium-scale and budget

Stopping test: $q^{so} \le$ `0.001`, `sec` $\le$ `800`, `nf` $\le$ `50*n`										
188 of 249 problems without bounds solved									Mean efficiency in %	
dim∈[101,1000]						# of anomalies			For cost measure	
Solver		Solved	#100	!100	T_{mean}	#n	#t	#f	nf	msec
LMLS4	**lmtr4**	**169**	**58**	**56**	11582	78	2	0	**50**	46
LMLS3	**lmtr3**	168	48	47	12980	79	2	0	47	37
FMINUNC	**func**	168	32	5	11064	78	3	0	36	42
FMINUNC1	**func1**	165	43	16	**10503**	81	3	0	39	**50**
LMLS2	**lmtr2**	164	16	15	15614	84	1	0	43	25
LMLS1	**lmtr1**	159	20	20	15399	88	2	0	38	28

Table 8 Results for medium-scale and large budget

Stopping test: $q^{so} \le$ `0.001`, `sec` $\le$ `800`, `nf` $\le$ `100*n`										
198 of 249 problems without bounds solved									Mean efficiency in %	
dim∈[101,1000]						# of anomalies			For cost measure	
Solver		Solved	#100	!100	T_{mean}	#n	#t	#f	nf	msec
FMINUNC	**func**	**189**	44	18	14603	55	3	2	41	48
FMINUNC1	**func1**	178	46	20	**12133**	66	3	2	41	**52**
LMLS3	**lmtr3**	176	39	37	16832	71	2	0	48	36
LMLS4	**lmtr4**	174	**53**	**51**	18938	73	2	0	**50**	47
LMLS2	**lmtr2**	167	24	24	18182	81	1	0	42	24
LMLS1	**lmtr1**	165	20	20	28711	84	0	0	38	26

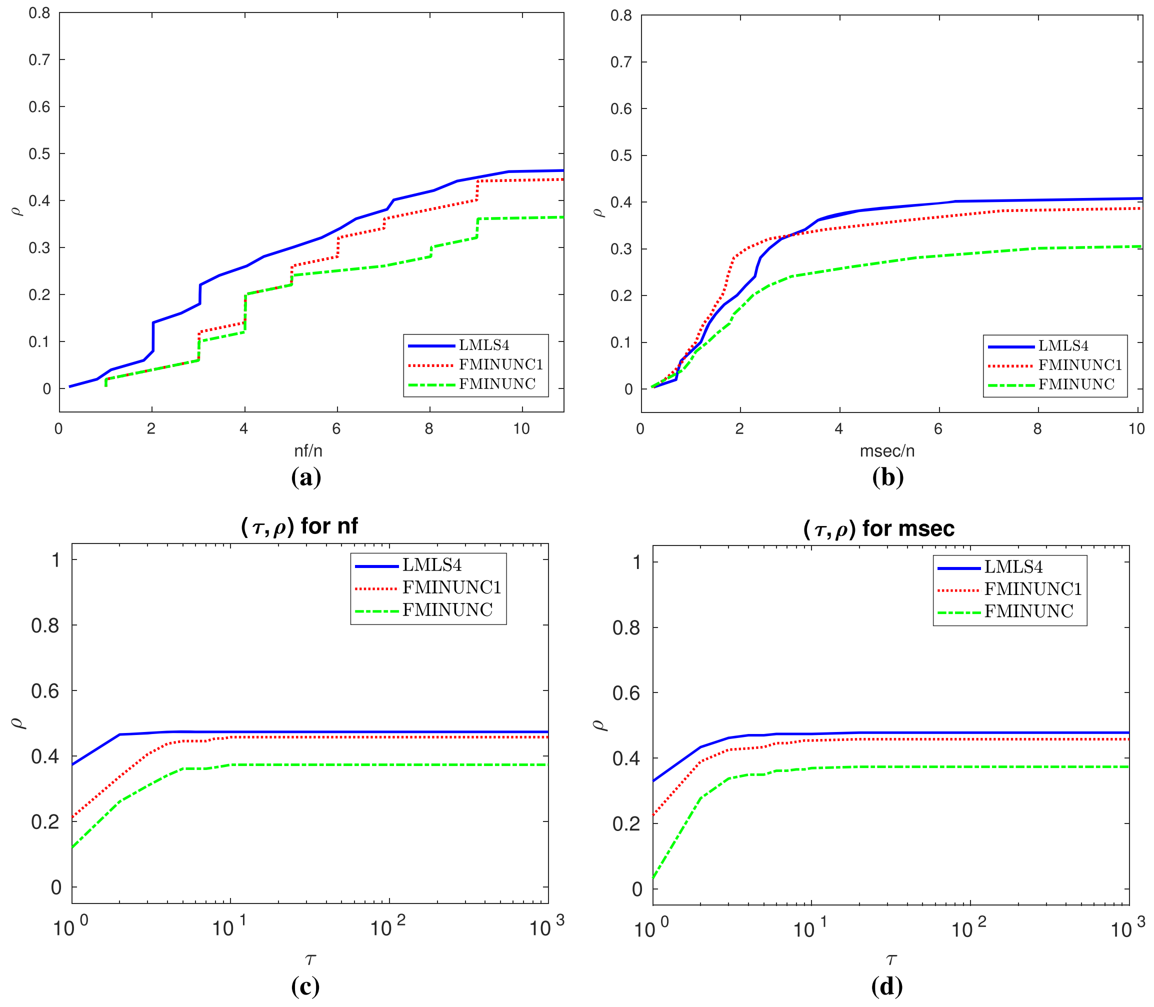

Fig. 8 Details as in Fig. 4

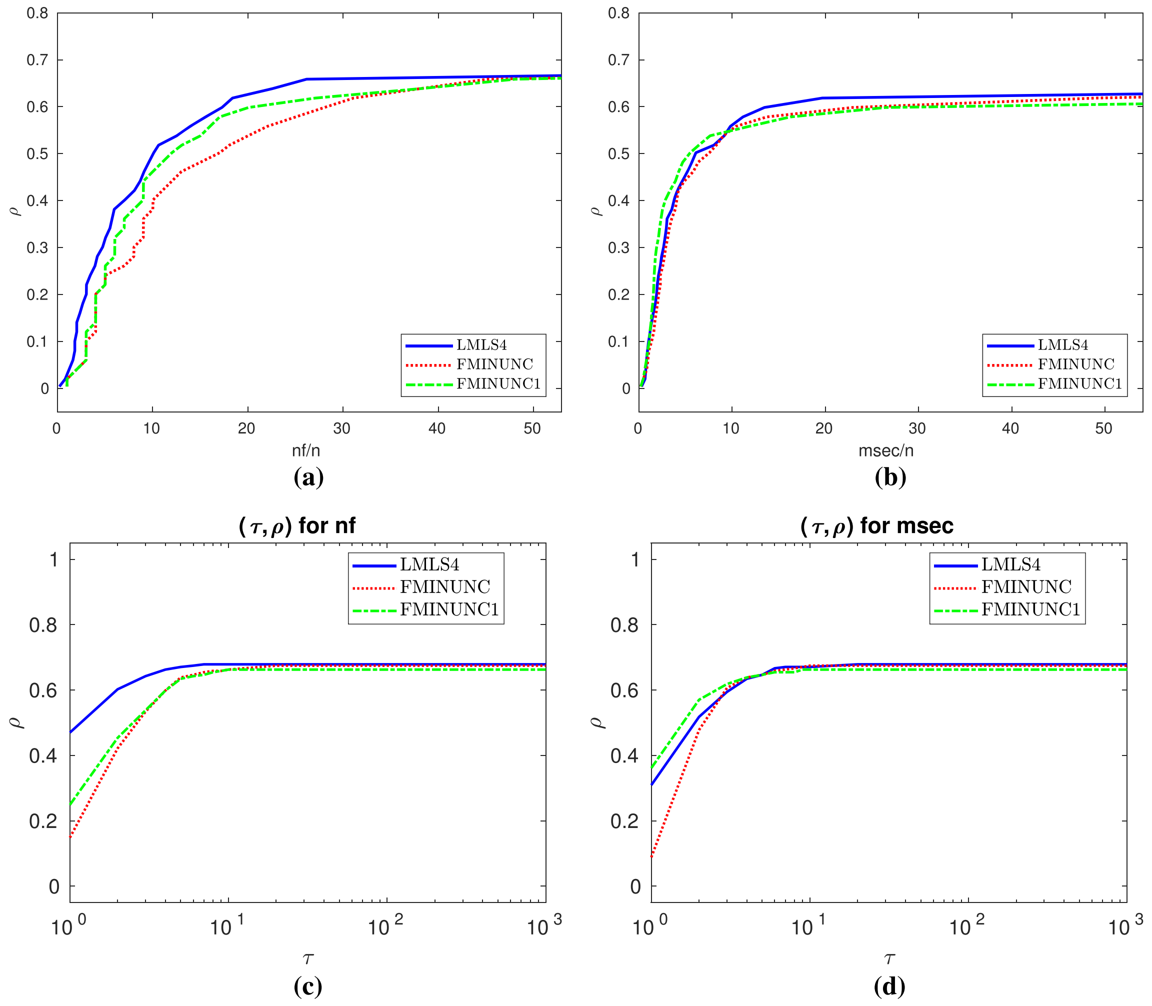

Fig. 9 Details as in Fig. 4

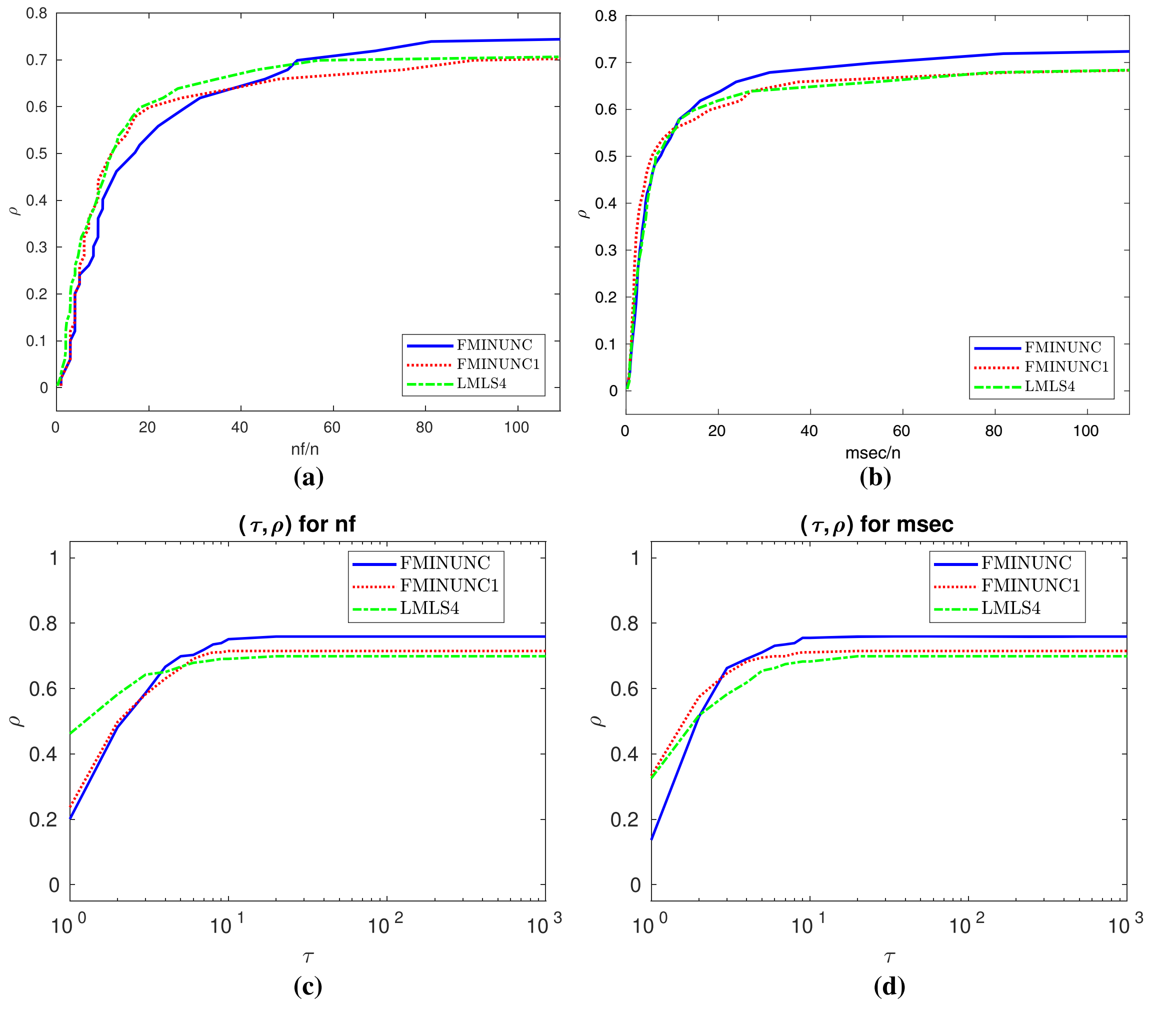

Fig. 10 Details as in Fig. 4

4.4 Tables and data/performance profiles for $1001 \le n \le 10000$

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

Stopping test:	$q^{so} \le$ `0.001`,	`sec` $\le$ `800`,	`nf` $\le$ `10*n`							
132 of 166 problems without bounds solved									Mean efficiency in %	
dim∈[1001,10000]					# of anomalies				For cost measure	
Solver		Solved	#100	!100	T_{mean}	#n	#t	#f	nf	msec
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

Stopping test:	$q^{so} \le$ `0.001`,	`sec` $\le$ `800`,	`nf` $\le$ `100*n`							
147 of 166 problems without bounds solved									Mean efficiency in %	
dim∈[1001,10000]					# of anomalies				For cost measure	
Solver		Solved	#100	!100	T_{mean}	#n	#t	#f	nf	msec
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

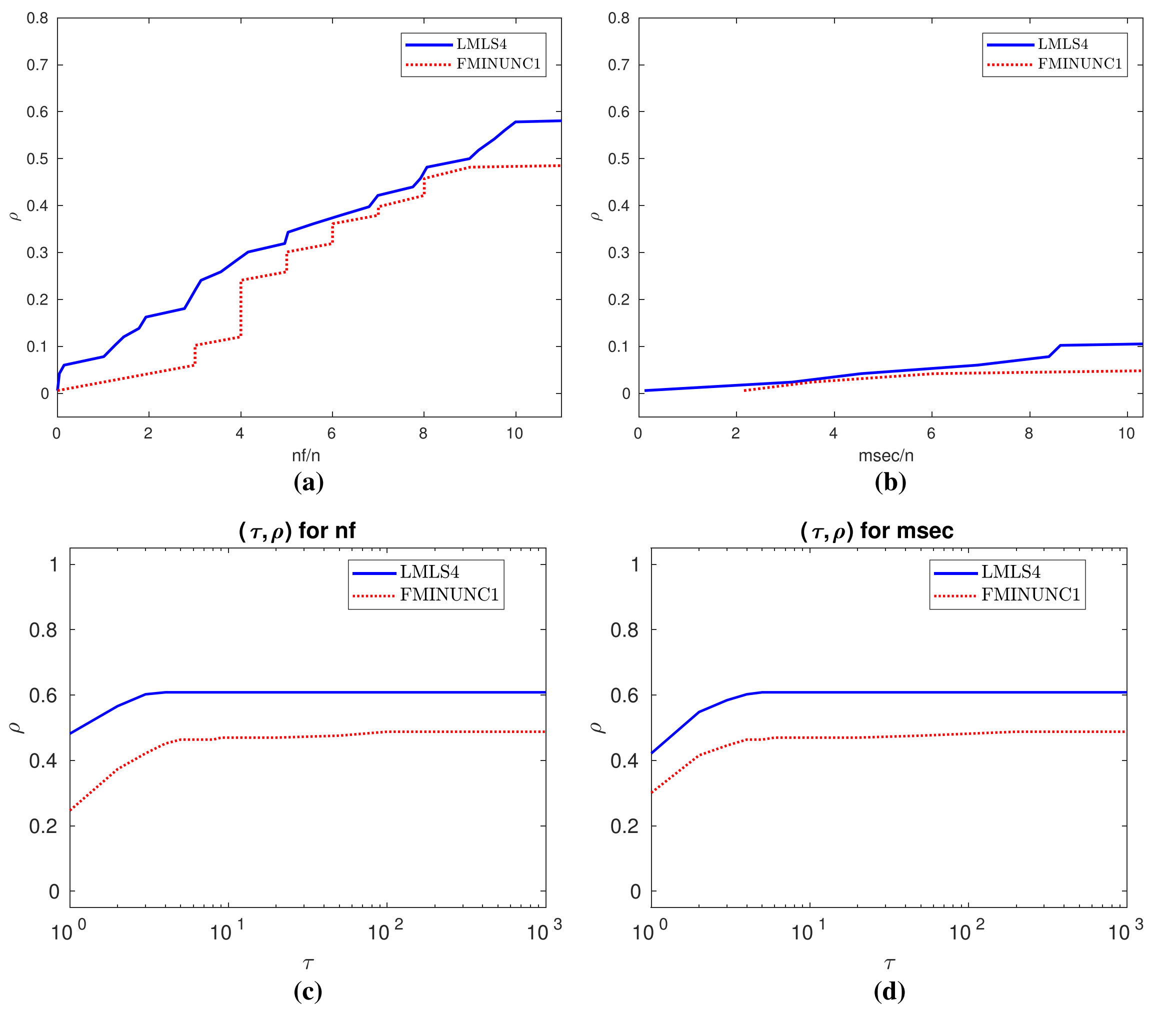

Fig. 11 Details as in Fig. 4

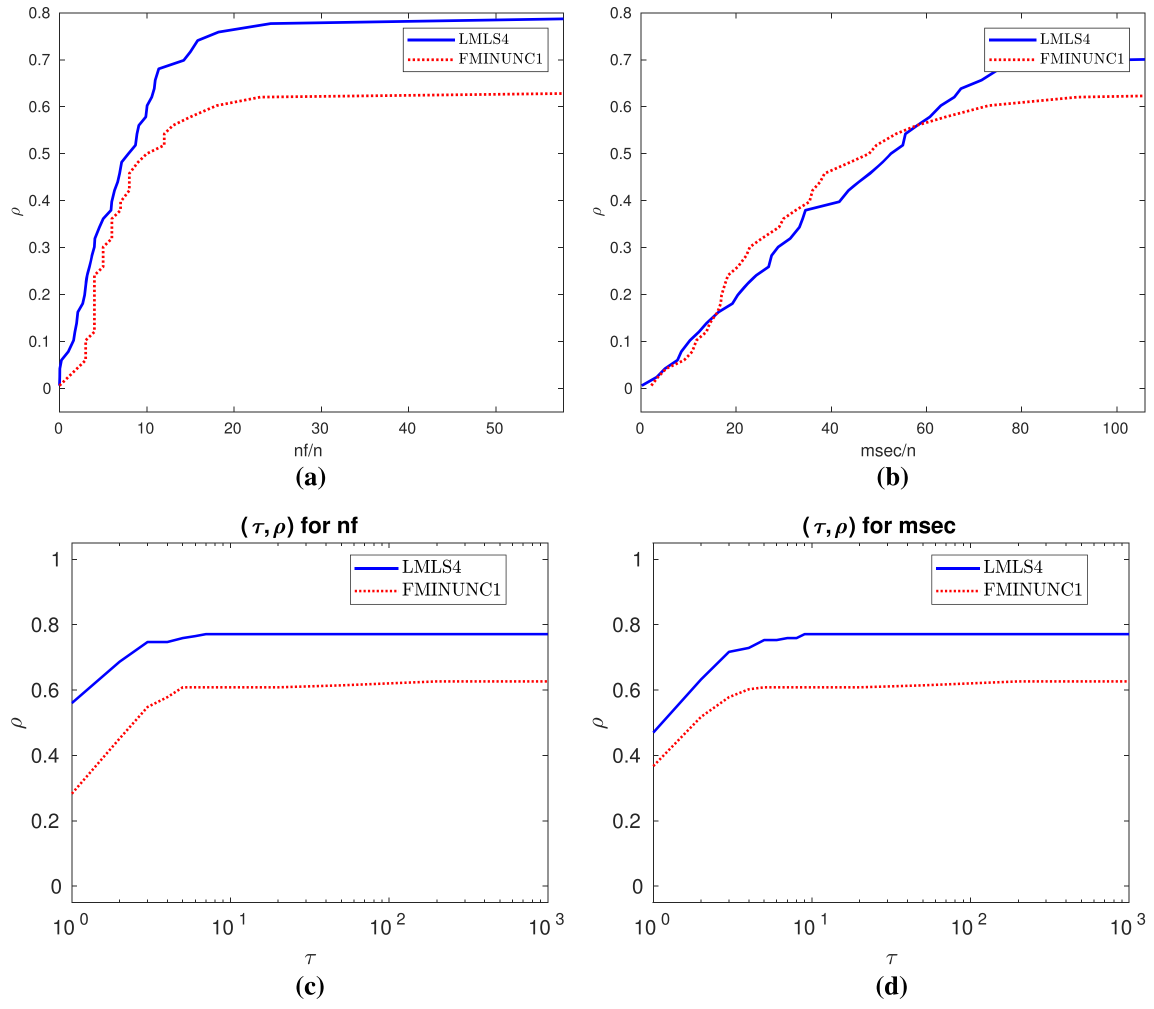

Fig. 12 Details as in Fig. 4

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.

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