J. Approx. Softw., Vol. 2, No. 1 (2025) 3

Journal of Approximation Software

doi: 10.13135/3103-1935/11898 – ISSN: 3103–1935

The MNGNREG toolbox for the regularized solution of nonlinear least-squares problems

F. Pes [1,⋆] and G. Rodriguez [1]

[1] *Dipartimento di Matematica e Informatica, Università degli Studi di Cagliari*

Received: 04/06/2025 – Published: 15/09/2025
Communicated by: M. Vianello (from the SA2025 workshop)

Abstract

This paper describes a Matlab toolbox designed to solve nonlinear least-squares problems, with a particular focus on ill-posed cases lacking unique solution, allowing to obtain the minimal-norm solution. The algorithm is based on the Gauss–Newton method, in which the iteration is modified introducing a projection term onto the null space of the Jacobian of the nonlinear function. To address the severe ill-conditioning often encountered in real-world applications, the toolbox also includes some regularization techniques.

Keywords: nonlinear least-squares, Gauss–Newton method, regularization (MSC2020: 65F22, 65H10)

1 Introduction

Let us consider the function $F(\boldsymbol{x}) = [F_1(\boldsymbol{x}), \ldots, F_m(\boldsymbol{x})]^T$, nonlinear and twice continuously Fréchet-differentiable, mapping from $\mathbb{R}^n$ to $\mathbb{R}^m$. We examine the problem of fitting a data vector $\boldsymbol{b} \in \mathbb{R}^m$ by nonlinear least-squares. This involves solving the minimization problem

$$\min_{\boldsymbol{x}\in\mathbb{R}^n} \|\boldsymbol{r}(\boldsymbol{x})\|^2, \quad \text{where} \quad \boldsymbol{r}(\boldsymbol{x}) = F(\boldsymbol{x}) - \boldsymbol{b}, \tag{1}$$

with $\|\cdot\|$ representing the Euclidean norm and $\boldsymbol{r}(\boldsymbol{x}) = [r_1(\boldsymbol{x}), \ldots, r_m(\boldsymbol{x})]^T$ the residual between the model output $F(\boldsymbol{x})$ and the observed data $\boldsymbol{b}$.

A common approach to tackle such problems is through Newton-based algorithms, the Gauss–Newton method being a prominent variant [2, 20, 21]. The Gauss–Newton method is based on the construction of a sequence of linear approximations of $\boldsymbol{r}(\boldsymbol{x})$. Chosen an initial guess $\boldsymbol{x}^{(0)}$ and denoting by $\boldsymbol{x}^{(k)}$ the current approximation, the iterative scheme generates new approximations

$$\boldsymbol{x}^{(k+1)} = \boldsymbol{x}^{(k)} + \alpha_k \boldsymbol{s}^{(k)}, \qquad k = 0, 1, 2, \ldots, \tag{2}$$

⋆ Corresponding author: federica.pes@unica.it

where the step $\boldsymbol{s}^{(k)}$ is the solution to the linear least-squares problem

$$\min_{\boldsymbol{s}\in\mathbb{R}^n} \|J(\boldsymbol{x}^{(k)})\boldsymbol{s} + \boldsymbol{r}(\boldsymbol{x}^{(k)})\|^2, \tag{3}$$

$J(\boldsymbol{x}) \in \mathbb{R}^{m\times n}$ represents the Jacobian matrix of the function $\boldsymbol{r}(\boldsymbol{x})$, and $\alpha_k \in (0,1]$ is a damping parameter employed in the line search strategy to improve convergence. If the scalar α_k is chosen too small convergence is slow, if it is too large the iteration may diverge. We estimate the step length α_k by the Armijo–Goldstein principle [1, 2, 17]: it is chosen as the largest number in the sequence $1/2^i$, $i = 0,1,\ldots$, for which the following inequality holds

$$\|\boldsymbol{r}(\boldsymbol{x}^{(k)})\|^2 - \|\boldsymbol{r}(\boldsymbol{x}^{(k)} + \alpha_k \boldsymbol{s}^{(k)})\|^2 \geq \frac{1}{2}\alpha_k \|J(\boldsymbol{x}^{(k)})\boldsymbol{s}^{(k)}\|^2. \tag{4}$$

We are interested in underdetermined problems, that is, problems where the solution is not unique. This happens when $J(\boldsymbol{x}^{(k)})$ lacks full column rank, a common situation when $m < n$. To address this, we solve a minimal-norm least-squares problem [15, 16, 22, 23, 24]

$$\begin{cases} \min\limits_{\boldsymbol{x}\in\mathbb{R}^n} \|\boldsymbol{x} - \bar{\boldsymbol{x}}\|^2 \\ \boldsymbol{x} \in \left\{ \arg\min\limits_{\boldsymbol{x}\in\mathbb{R}^n} \|F(\boldsymbol{x}) - \boldsymbol{b}\|^2 \right\}, \end{cases} \tag{5}$$

where a model profile $\bar{\boldsymbol{x}}$ is included to leverage prior knowledge about the solution. Solution methods of such problems are based on variants of the Gauss–Newton method.

To increase flexibility, the seminorm $\|L(\boldsymbol{x} - \bar{\boldsymbol{x}})\|$ is often used in place of $\|\boldsymbol{x} - \bar{\boldsymbol{x}}\|$ in (5), where $L \in \mathbb{R}^{p\times n}$ encodes prior knowledge about the solution. Common choices for L are, for instance, the matrices

$$D_1 = \begin{bmatrix} -1 & 1 & & \\ & \ddots & \ddots & \\ & & -1 & 1 \end{bmatrix} \in \mathbb{R}^{(n-1)\times n}, \qquad D_2 = \begin{bmatrix} 1 & -2 & 1 & & \\ & \ddots & \ddots & \ddots & \\ & & 1 & -2 & 1 \end{bmatrix} \in \mathbb{R}^{(n-2)\times n},$$

which represent discrete approximations to first and second derivative operators, respectively.

Incorporating L and considering an iterative method of the type (2)-(3), the linear approximation of (5) is

$$\begin{cases} \min\limits_{\boldsymbol{s}\in\mathbb{R}^n} \|L(\boldsymbol{x}^{(k)} - \bar{\boldsymbol{x}} + \alpha_k \boldsymbol{s})\|^2 \\ \boldsymbol{s} \in \left\{ \arg\min\limits_{\boldsymbol{s}\in\mathbb{R}^n} \|J_k \boldsymbol{s} + \boldsymbol{r}_k\|^2 \right\}, \end{cases} \tag{6}$$

where $J_k = J(\boldsymbol{x}^{(k)})$ and $\boldsymbol{r}_k = \boldsymbol{r}(\boldsymbol{x}^{(k)})$. We denote this as the *minimal-L-norm Gauss–Newton* method (MLNGN) or simply MNGN if L is the identity matrix.

In [23], we proved that, when $\bar{\boldsymbol{x}} = \boldsymbol{0}$, the iteration of the MNGN method is obtained by subtracting from (2) the projection of the current approximation onto the null space of J_k, as in

$$\boldsymbol{x}^{(k+1)} = \boldsymbol{x}^{(k)} + \alpha_k \boldsymbol{s}^{(k)} - \mathscr{P}_{\mathscr{N}(J_k)} \boldsymbol{x}^{(k)}, \qquad k = 0,1,2,\ldots, \tag{7}$$

where $\mathscr{P}_{\mathscr{N}(J_k)}$ denotes the projector onto the null space of J_k. In [24], by a second-order analysis of the function $\frac{1}{2}\|\boldsymbol{r}(\boldsymbol{x})\|^2$, we showed that the MNGN method is not guaranteed to converge. In fact, both the Gauss–Newton search direction and the projection step must be damped to ensure that the residual decreases as the iteration progresses. To avoid these convergence issues, we introduced the MNGN2 iterative method

$$\boldsymbol{x}^{(k+1)} = \boldsymbol{x}^{(k)} + \alpha_k \boldsymbol{s}^{(k)} - \beta_k \mathscr{P}_{\mathscr{N}(J_k)} \boldsymbol{x}^{(k)}, \qquad k = 0,1,2,\ldots, \tag{8}$$

where the parameter β_k prevents the projection term to cause an increase in the residual. In the same paper, some techniques to estimate such parameter were introduced and discussed in detail.

We stress that the MNGN2 method was introduced in [24] to tackle well-conditioned nonlinear least-squares problems. For ill-conditioned problems, a regularization approach is essential. In [22], a regularized version of the MNGN2 iteration (8), the truncated MNGN2 method, was introduce to compute regularized solutions of nonlinear least-squares problems.

The paper is organized as follows: Section 2 recalls the definition and the principal properties of the singular value decomposition and its generalized extension. We revise the MNGN2 algorithm in Section 3, and two regularized versions of the same method in Section 4. A description of the toolbox that accompanies this paper is provided in Section 5. Section 6 presents a few numerical examples that illustrate the performance of our methods. Section 7 contains concluding remarks.

2 Basic notions on SVD and GSVD

The singular value decomposition (SVD) is a matrix decomposition of the form

$$J = U\Sigma V^T. \tag{9}$$

The diagonal matrix $\Sigma \in \mathbb{R}^{m\times n}$ contains the *singular values*

$$\sigma_1 \geq \sigma_2 \geq \cdots \geq \sigma_r > 0,$$

where $r = \operatorname{rank}(J) \leq \min(m,n)$, and the matrices $U = [\boldsymbol{u}_1, \ldots, \boldsymbol{u}_m] \in \mathbb{R}^{m\times m}$ and $V = [\boldsymbol{v}_1, \ldots, \boldsymbol{v}_n] \in \mathbb{R}^{n\times n}$ are orthogonal.

Let $J \in \mathbb{R}^{m\times n}$ and $L \in \mathbb{R}^{p\times n}$ ($p \leq n$) be matrices with $\operatorname{rank}(J) = r$ and $\operatorname{rank}(L) = p$. Assume that $m + p \geq n$ and

$$\operatorname{rank}\left(\begin{bmatrix} J \\ L \end{bmatrix}\right) = n, \quad \text{that is,} \quad \mathcal{N}(J) \cap \mathcal{N}(L) = \{0\}.$$

The generalized singular value decomposition (GSVD) of the matrix pair (J,L) is defined by the factorizations

$$J = U\Sigma_J W^{-1}, \qquad L = V\Sigma_L W^{-1}, \tag{10}$$

where $U \in \mathbb{R}^{m\times m}$ and $V \in \mathbb{R}^{p\times p}$ are matrices with orthonormal columns $\boldsymbol{u}_i$ and $\boldsymbol{v}_i$, respectively, $W \in \mathbb{R}^{n\times n}$ is nonsingular, $\Sigma_J \in \mathbb{R}^{m\times n}$ and $\Sigma_L \in \mathbb{R}^{p\times n}$ are diagonal matrices.

If $m \geq n \geq r$, the two diagonal matrices have the following structure

$$\Sigma_J = \left[\begin{array}{ccc} O_{n-r} & & \\ & C & \\ \hline & & I_d \\ & O_{(m-n)\times n} & \end{array}\right], \qquad \Sigma_L = \left[\begin{array}{ccc|c} I_{p-r+d} & & & \\ & & S & O_{p\times d} \end{array}\right],$$

where $d = n - p$ and the blocks C and S are given by

$$\begin{aligned} C &= \operatorname{diag}(c_1, \ldots, c_{r-d}), & 0 < c_1 \leq c_2 \leq \cdots \leq c_{r-d} < 1, \\ S &= \operatorname{diag}(s_1, \ldots, s_{r-d}), & 1 > s_1 \geq s_2 \geq \cdots \geq s_{r-d} > 0, \end{aligned} \tag{11}$$

with $c_i^2 + s_i^2 = 1$, for $i = 1, \ldots, r-d$. The identity matrix of size k is represented as I_k, whereas O_k and $O_{k\times\ell}$ denote zero matrices of sizes k and $k \times \ell$, respectively. The identity or zero blocks must be ignored if either of their dimensions is zero.

If $r \leq m < n$, the matrix Σ_J has the form

$$\Sigma_J = \left[\begin{array}{c|ccc} O_{m\times(n-m)} & O_{m-r} & & \\ & & C & \\ & & & I_d \end{array}\right],$$

where the blocks are defined as above, and Σ_L has the same structure as in the previous case.

In the software presented in this paper, we use the Matlab routines `svd` and `gsvd` to compute the SVD of the matrix J and the GSVD of the matrix pair (J, L), respectively.

3 A review of the numerical method

The main properties of iteration (7) for approximating the minimal-norm solution of (5) are contained in the following theorem.

Theorem 3.1. *Let $\boldsymbol{x}^{(k)} \in \mathbb{R}^n$ and let $\widetilde{\boldsymbol{x}}^{(k+1)} = \boldsymbol{x}^{(k)} + \alpha_k \widetilde{\boldsymbol{s}}^{(k)}$ be the Gauss–Newton iteration for (1), where the step $\widetilde{\boldsymbol{s}}^{(k)}$ is determined by solving (3) and the step length α_k is estimated by the Armijo–Goldstein principle. Then, the iteration defined by (6) is explicitly represented by*

$$\boldsymbol{x}^{(k+1)} = \widetilde{\boldsymbol{x}}^{(k+1)} - \mathscr{P}_{\mathscr{N}(J_k)}\left(\boldsymbol{x}^{(k)} - \bar{\boldsymbol{x}}\right), \qquad k = 0, 1, 2, \ldots, \tag{12}$$

where $\mathscr{P}_{\mathscr{N}(J_k)}$ denotes the projector onto the null space of the Jacobian $J_k = J(\boldsymbol{x}^{(k)})$.

If $L = I_n$, given the SVD (9) of J_k, then $\mathscr{P}_{\mathscr{N}(J_k)} = V_2 V_2^T$ is the orthogonal projector onto the null space of J_k, where $V_2 = [\boldsymbol{v}_{r_k+1}, \ldots, \boldsymbol{v}_n]$ and $r_k = \operatorname{rank}(J_k)$.

If $L \neq I_n$, given the GSVD (10) of (J_k, L), then $\mathscr{P}_{\mathscr{N}(J_k)} = W_1 \widehat{W}_1$ is an oblique projector onto $\mathscr{N}(J_k)$, where $W_1 \in \mathbb{R}^{n\times(n-r_k)}$ consists of the first $n - r_k$ columns $\boldsymbol{w}_i$ of W, and $\widehat{W}_1 \in \mathbb{R}^{(n-r_k)\times n}$ contains the first $n - r_k$ rows $\widehat{\boldsymbol{w}}^i$ of W^{-1}.

Proof. See [23, Theorem 3.1] for the case $L = I_n$. See [23, Lemma 4.1 and Theorem 4.2] for the case $L \neq I_n$. □

In [24], we introduced in Equation (12) a second relaxation parameter β_k, to control the step length of the minimal-norm correction

$$\boldsymbol{t}^{(k)} = \mathscr{P}_{\mathscr{N}(J_k)}\left(\boldsymbol{x}^{(k)} - \bar{\boldsymbol{x}}\right). \tag{13}$$

This leads to a new iterative method, called *doubly relaxed MNGN* and denoted MNGN2, which is represented by

$$\boldsymbol{x}^{(k+1)} = \boldsymbol{x}^{(k)} + \alpha_k \widetilde{\boldsymbol{s}}^{(k)} - \beta_k \boldsymbol{t}^{(k)}, \tag{14}$$

where $\widetilde{\boldsymbol{s}}^{(k)}$ is the step vector generated by the Gauss–Newton method and $\boldsymbol{t}^{(k)}$ is the projection vector which makes the norm of $\boldsymbol{x}^{(k+1)}$ minimal, without changing the linearized residual.

The MNGN2 method locally converges if α_k and β_k are appropriately chosen, but it will recover the minimal-norm solution only if $\beta_k \simeq 1$ as k approaches convergence, as the projection onto $\mathscr{N}(J_k)$ is effective only for $\beta_k = 1$. Computational methods for dynamically estimating α_k and β_k are described in [24].

3.1 Estimating the rank of J_k

To effectively compute the minimal-norm solution by iterations (12) or (14), the rank of the Jacobian matrix J_k must be known beforehand. Since the rank may change during iteration, we set $r_k = \text{rank}(J_k)$. However, the exact value of r_k for each iteration $k = 0, 1, \ldots$ is generally unknown. It is therefore essential to estimate the value of r_k at each step, to prevent algorithmic breakdowns or convergence issues.

In such cases, the concept of numerical rank $r_{\varepsilon,k}$, often called the ε-rank, becomes useful. Here, ε represents a chosen tolerance. If $L = I_n$, the numerical rank is determined using the singular values $\sigma_i^{(k)}$ of J_k, and it is defined as the integer $r_{\varepsilon,k}$ satisfying the condition

$$\sigma_{r_{\varepsilon,k}}^{(k)} > \varepsilon \geq \sigma_{r_{\varepsilon,k}+1}^{(k)}.$$

Theorem 3.1 can be adapted to this setting, by replacing the rank r_k with the numerical rank $r_{\varepsilon,k}$ at each iteration. Estimating $r_{\varepsilon,k}$ is challenging for discrete ill-posed problems, for which singular values decay monotonically to zero.

The numerical rank is determined identifying a clear gap between $\sigma_{r_{\varepsilon,k}}^{(k)}$ and $\sigma_{r_{\varepsilon,k}+1}^{(k)}$. To detect such a gap, we adopt a heuristic strategy similar to the one employed in [7] in a different setting. At each step, we compute the ratios

$$\rho_i^{(k)} = \frac{\sigma_i^{(k)}}{\sigma_{i+1}^{(k)}}, \qquad i = 1, 2, \ldots, q-1,$$

where $q = \min(m, n)$. We then consider the index set

$$\mathscr{I}_k = \left\{ i \in \{1, 2, \ldots, q-1\} : \rho_i^{(k)} > R \text{ and } \sigma_i^{(k)} > \tau \right\},$$

for given constants R and τ. In our numerical experiments, we set $R = 10^2$ and $\tau = 10^{-8}$. An index i belongs to $\mathscr{I}_k$ if there is a significant "jump" between $\sigma_i^{(k)}$ and $\sigma_{i+1}^{(k)}$, and $\sigma_i^{(k)}$ is numerically nonzero. If the set $\mathscr{I}_k$ is empty, we set $r_{\varepsilon,k} = q$. Otherwise, we consider

$$\rho_j^{(k)} = \max_{i \in \mathscr{I}_k} \rho_i^{(k)}, \tag{15}$$

and we define $r_{\varepsilon,k} = j$. This approach selects the largest gap between "large" and "small" singular values.

In case of $L \neq I_n$, the rank of the Jacobian is estimated at each step by applying the same procedure described above to the diagonal elements $c_i^{(k)}$ of the GSVD factor Σ_J of J_k; see Equation (11). In this case, at each step, we compute the ratios

$$\rho_i^{(k)} = \frac{c_{i+1}^{(k)}}{c_i^{(k)}}, \qquad i = 1, 2, \ldots, q-d-1,$$

where $q = \min(m, n)$, and apply the same procedure as above to estimate the numerical rank.

3.2 The algorithm

Our numerical tests showed the importance of adaptively selecting both α_k and β_k along the iterations. A simple strategy is to let $\beta_k = \alpha_k$ and estimate α_k using the Armijo–Goldstein principle (4), with $\boldsymbol{s}^{(k)} = \widetilde{\boldsymbol{s}}^{(k)} - \boldsymbol{t}^{(k)}$. While this approach is effective in computing the minimal-norm solution, the convergence is often slow. To accelerate the iterations, we propose an adaptive procedure to optimize the choice of β_k.

Algorithm 1 Outline of the MNGN2 method.

Require: nonlinear function F, data vector $\boldsymbol{b}$,
Require: initial solution $\boldsymbol{x}^{(0)}$, model profile $\bar{\boldsymbol{x}}$, tolerance η for residual increase
Ensure: approximation $\boldsymbol{x}^{(k+1)}$ of minimal-norm least-squares solution
1: $k = 0$, $\beta = 1$
2: **repeat**
3: $k = k + 1$
4: estimate $r_{\varepsilon,k} = \operatorname{rank}(J(\boldsymbol{x}^{(k)}))$ by (15)
5: compute $\widetilde{\boldsymbol{s}}^{(k)}$ by the Gauss–Newton method (3)
6: compute α_k by the Armijo–Goldstein principle (4)
7: compute $\boldsymbol{t}^{(k)}$ by (13)
8: **if** $\beta < 1$ **then**
9: $\beta = 2\beta$
10: **end if**
11: $\widetilde{\boldsymbol{x}}^{(k+1)} = \boldsymbol{x}^{(k)} + \alpha_k \widetilde{\boldsymbol{s}}^{(k)}$
12: $\widetilde{\rho}_{k+1} = \|F(\widetilde{\boldsymbol{x}}^{(k+1)}) - \boldsymbol{b}\| + \varepsilon_M$
13: $\boldsymbol{x}^{(k+1)} = \widetilde{\boldsymbol{x}}^{(k+1)} - \beta \boldsymbol{t}^{(k)}$
14: $\rho_{k+1} = \|F(\boldsymbol{x}^{(k+1)}) - \boldsymbol{b}\|$
15: **while** $(\rho_{k+1} > \widetilde{\rho}_{k+1} + \delta(\widetilde{\rho}_{k+1}, \eta))$ **and** $(\beta > 10^{-8})$ **do**
16: $\beta = \beta/2$
17: $\boldsymbol{x}^{(k+1)} = \widetilde{\boldsymbol{x}}^{(k+1)} - \beta \boldsymbol{t}^{(k)}$
18: $\rho_{k+1} = \|F(\boldsymbol{x}^{(k+1)}) - \boldsymbol{b}\|$
19: **end while**
20: $\beta_k = \beta$
21: **until** convergence

This procedure is outlined in Algorithm 1. We start with $\beta = 1$. At each iteration, we compute the residual at the Gauss–Newton step $\widetilde{\boldsymbol{x}}^{(k+1)}$ and at the tentative iteration $\boldsymbol{x}^{(k+1)} = \widetilde{\boldsymbol{x}}^{(k+1)} - \beta \boldsymbol{t}^{(k)}$ (lines 12 and 14 in Algorithm 1). Subtraction of the vector $\beta \boldsymbol{t}^{(k)}$ could cause an increase in the residual, which if not too large, is useful to reach the minimal-norm solution. So we accept such an increase if

$$\|\boldsymbol{r}(\boldsymbol{x}^{(k+1)})\| \le \|\boldsymbol{r}(\widetilde{\boldsymbol{x}}^{(k+1)})\| + \delta\big(\|\boldsymbol{r}(\widetilde{\boldsymbol{x}}^{(k+1)})\|, \eta\big), \tag{16}$$

where $\delta(\rho, \eta)$ is a function determining the maximal allowed increase for the residual $\rho = \|\boldsymbol{r}(\widetilde{\boldsymbol{x}}^{(k+1)})\|$, and $\eta > 0$ is a chosen tolerance.

If this condition is not satisfied, β is halved (line 16 of Algorithm 1), and the residual is recomputed until (16) is met or β becomes too small. To allow β to increase, we attempt to double it at each iteration (line 9 in Algorithm 1) before applying the residual-checking procedure. At line 12 of the algorithm we add the machine epsilon ε_M to the actual residual $\widetilde{\rho}_{k+1}$ to prevent $\delta(\widetilde{\rho}_{k+1}, \eta)$ from becoming zero.

A possible choice for the value of the residual increase is $\delta(\rho,\eta) = \eta\rho$, with η carefully chosen. Our experiments showed that it is possible to find, by chance, a value of η which produces good results, but its choice is strongly dependent on the particular example. We also noticed that, in cases where the residual stagnates, allowing a large increase in the residual may lead to nonconvergence. Specifically, when the residual is large, only a small increase should be accepted, whereas for very small residuals, relatively larger increases might be tolerable. In such situations, a fixed multiple of the residual is not well suited to model its increase.

To address these challenges, we consider $\delta(\rho,\eta) = \rho^\eta$, and adaptively select η at each iteration by the procedure in Algorithm 2. After at least k_{res} iterations, we compute the linear polynomial which fits the logarithm of the most recent k_{res} residuals in the least-squares sense (line 3 in Algorithm 2). To detect residual stagnation or growth, we check if the slope M of the regression line exceeds -10^{-2}. If this condition is met, the value of η is doubled. To recover a sensible decrease in the norm, if at a subsequent step the residual reduction accelerates (e.g., $M < -\frac{1}{2}$), the value of η is halved. In our experiments, we initialize η to $\frac{1}{8}$ and set $k_{\text{res}} = 5$.

Algorithm 2 Adaptive determination of the residual increase $\delta(\rho,\eta)$.

Require: actual residual $\rho = \|\boldsymbol{r}(\widetilde{\boldsymbol{x}}^{(k+1)})\|$, starting tolerance η
Require: iteration index k, residuals $\theta_j = \|\boldsymbol{r}(\widetilde{\boldsymbol{x}}^{(k-k_{\text{res}}+j)})\|$, $j = 1,\dots,k_{\text{res}}$
Ensure: residual increase $\delta(\rho,\eta)$
1: $M_{\min} = -10^{-2}$, $M_{\max} = -\frac{1}{2}$
2: **if** $k \geq k_{\text{res}}$ **then**
3: compute regression line $p_1(t) = Mt + N$ of $(j, \log(\theta_j))$, $j = 1,\dots,k_{\text{res}}$
4: **if** $M > M_{\min}$ **then**
5: $\eta = 2\eta$
6: **else if** $M < M_{\max}$ **then**
7: $\eta = \eta/2$
8: **end if**
9: **end if**
10: $\delta(\rho,\eta) = \rho^\eta$

To detect convergence of the MNGN2 method, we interrupt the iteration as soon as

$$\|\boldsymbol{x}^{(k+1)} - \boldsymbol{x}^{(k)}\| < \tau\|\boldsymbol{x}^{(k+1)}\| \qquad \text{or} \qquad \|\alpha_k \widetilde{\boldsymbol{s}}^{(k)}\| < \tau, \tag{17}$$

or when a fixed number of iterations $N_{\max}$ is exceeded. The default value of the stop tolerance is $\tau = 10^{-8}$, and $N_{\max}$ is set to 100. The second stopping criterion in (17) serves to identify when the relaxed Gauss–Newton iteration algorithm exhibits slow progress, which typically occurs close to the solution.

4 Regularization

A nonlinear function $F(\boldsymbol{x})$ is considered ill-conditioned in a domain $D \subset \mathbb{R}^n$ when the condition number $\kappa(J)$ of the Jacobian $J = J(\boldsymbol{x})$ is very large for any $\boldsymbol{x} \in D$. In this situation, a common approach is to apply a regularization procedure at each step of the Gauss–Newton method. The ill-conditioned problem is replaced by a nearby better conditioned problem, whose solution is less sensitive to the error in the right-hand side and to round-off errors introduced during the solution process. For an in-depth discussion of regularization of inverse problems, the reader is referred to [14, 18].

The software presented in this paper computes regularized solutions to problem (5) by applying to its linearized version (6) either the truncated (generalized) singular value decomposition or Tikhonov's method.

4.1 Truncated minimal-norm solution

Let $U\Sigma V^T$ be the SVD of the matrix J_k and $r_k = \text{rank}(J_k)$, and choose a value for the truncation parameter $1 \le \ell \le r_k$. When the nonlinear function F is ill-conditioned and $L = I_n$, the truncated MNGN2 method solves (6) substituting to J_k its the best rank-ℓ approximation. Such approximation is produced by the so-called truncated SVD (TSVD). Then, the iterate of the truncated MNGN2 is of the form

$$\boldsymbol{x}_\ell^{(k+1)} = \boldsymbol{x}_\ell^{(k)} - \alpha_k \sum_{i=1}^{\ell} \frac{g_i^{(k)}}{\sigma_i} \boldsymbol{v}_i - \beta_k \sum_{i=\ell+1}^{n} (\boldsymbol{v}_i^T (\boldsymbol{x}_\ell^{(k)} - \bar{\boldsymbol{x}})) \boldsymbol{v}_i, \qquad k = 0, 1, 2, \ldots, \tag{18}$$

where $\boldsymbol{g}^{(k)} = U^T \boldsymbol{r}(\boldsymbol{x}_\ell^{(k)})$ and ℓ plays the role of a regularization parameter, which has to be carefully chosen. We remark that the SVD of the matrix J_k changes at each step. To simplify notation, we write U, $\boldsymbol{v}_i$, and σ_i without specifying the dependence on k.

Similarly, if $L \neq I_n$, the truncated MLNGN2 consists of choosing an integer $0 \le \ell \le p - n + r_k = r_k - d$ and computing at each step the GSVD of (J_k, L). The iteration is given by

$$\boldsymbol{x}_\ell^{(k+1)} = \boldsymbol{x}_\ell^{(k)} - \alpha_k \sum_{i=p-\ell+1}^{p} \frac{g_{i-N}^{(k)}}{c_{i-n+r_k}} \boldsymbol{w}_i - \alpha_k \sum_{i=p+1}^{n} g_{i-N}^{(k)} \boldsymbol{w}_i - \beta_k \sum_{i=1}^{p-\ell} (\widehat{\boldsymbol{w}}^i (\boldsymbol{x}_\ell^{(k)} - \bar{\boldsymbol{x}})) \boldsymbol{w}_i, \tag{19}$$

where $\boldsymbol{g}^{(k)} = U^T \boldsymbol{r}(\boldsymbol{x}_\ell^{(k)})$, $\boldsymbol{w}_i$ are the column vectors of the matrix W of the GSVD, and $\widehat{\boldsymbol{w}}^i$ are the row vectors of the matrix W^{-1}; see Equation (10). The integer $N = \max(n - m, 0)$ appearing in (19) allows us to condense in a single formula both the overdetermined ($m \ge n$) and underdetermined ($m < n$) cases. As in the previous formulation, the GSVD of (J_k, L) must be recomputed at each iteration.

In formulas (18) and (19), the solution at convergence will be denoted by $\boldsymbol{x}_\ell$.

4.2 Minimal-norm Tikhonov solution

Another classical approach is Tikhonov regularization, in which the minimization problem (6) is replaced by

$$\min_{\boldsymbol{s} \in \mathbb{R}^n} \left\{ \|J_k \boldsymbol{s} + \boldsymbol{r}_k\|^2 + \lambda^2 \|L(\boldsymbol{x}^{(k)} - \bar{\boldsymbol{x}} + \alpha_k \boldsymbol{s})\|^2 \right\}, \tag{20}$$

for a fixed value of the parameter $\lambda > 0$. The regularization parameter λ controls the balance between the two terms of the functional, i.e., the weights attributed to the residual term and the regularization term. As before, $L \in \mathbb{R}^{p \times n}$ is a regularization matrix.

The following theorems are extensions of results reported in [23]. They appear here for the first time. We first analyze the case $L = I_n$.

Theorem 4.1. *Let* $\text{rank}(J_k) = r_k$ *and let* $\alpha_k \in \mathbb{R}$ *be the step length determined by any strategy which ensures convergence. The iteration for* (20)*, when* $L = I_n$*, is given by*

$$\boldsymbol{x}_\lambda^{(k+1)} = \boldsymbol{x}_\lambda^{(k)} - \alpha_k \sum_{i=1}^{r_k} \frac{\sigma_i g_i^{(k)} + \alpha_k \lambda^2 z_i^{(k)}}{\sigma_i^2 + \alpha_k^2 \lambda^2} \boldsymbol{v}_i - V_2 V_2^T (\boldsymbol{x}_\lambda^{(k)} - \bar{\boldsymbol{x}}), \tag{21}$$

where $\boldsymbol{g}^{(k)} = U^T \boldsymbol{r}(\boldsymbol{x}_\lambda^{(k)})$, $\boldsymbol{z}^{(k)} = V^T(\boldsymbol{x}_\lambda^{(k)} - \bar{\boldsymbol{x}})$, *and* $V_2 = [\boldsymbol{v}_{r_k+1}, \ldots, \boldsymbol{v}_n]$ *is defined as in Theorem 3.1.*

Proof. Computing the gradient of the function (20) with $L = I_n$ yields the normal equations associated to the penalized least-squares problem

$$(J_k^T J_k + \alpha_k^2 \lambda^2 I_n)\boldsymbol{s} = -J_k^T \boldsymbol{r}_k - \alpha_k \lambda^2 (\boldsymbol{x}_\lambda^{(k)} - \bar{\boldsymbol{x}}). \tag{22}$$

By employing the singular value decomposition $J_k = U\Sigma V^T$, the normal equations (22) become

$$(\Sigma^T \Sigma + \alpha_k^2 \lambda^2 I_n)\boldsymbol{y} = -\Sigma^T \boldsymbol{g}^{(k)} - \alpha_k \lambda^2 \boldsymbol{z}^{(k)}, \tag{23}$$

with $\boldsymbol{y} = V^T \boldsymbol{s}$, $\boldsymbol{g}^{(k)} = U^T \boldsymbol{r}(\boldsymbol{x}_\lambda^{(k)})$, and $\boldsymbol{z}^{(k)} = V^T(\boldsymbol{x}_\lambda^{(k)} - \bar{\boldsymbol{x}})$. The solution to the diagonal normal equations (23)

$$y_i = \begin{cases} -\dfrac{\sigma_i g_i^{(k)} + \alpha_k \lambda^2 z_i^{(k)}}{\sigma_i^2 + \alpha_k^2 \lambda^2}, & i = 1, \ldots, r_k, \\ -\dfrac{1}{\alpha_k} z_i^{(k)}, & i = r_k + 1, \ldots, n, \end{cases}$$

leads to the Tikhonov–Gauss–Newton (TikGN) iterate, which solves (20),

$$\boldsymbol{x}_\lambda^{(k+1)} = \boldsymbol{x}_\lambda^{(k)} + \alpha_k \boldsymbol{s} = \boldsymbol{x}_\lambda^{(k)} - \alpha_k \sum_{i=1}^{r_k} \frac{\sigma_i g_i^{(k)} + \alpha_k \lambda^2 z_i^{(k)}}{\sigma_i^2 + \alpha_k^2 \lambda^2} \boldsymbol{v}_i - \sum_{i=r_k+1}^{n} z_i^{(k)} \boldsymbol{v}_i.$$

The last summation can be rewritten in matrix form as $V_2 V_2^T (\boldsymbol{x}_\lambda^{(k)} - \bar{\boldsymbol{x}})$, where $V_2 = [\boldsymbol{v}_{r_k+1}, \ldots, \boldsymbol{v}_n]$. □

We now turn to the case $L \neq I_n$. We denote the resulting method by TikLGN.

Theorem 4.2. *Let* $\operatorname{rank}(J_k) = r_k$ *and let* $\alpha_k \in \mathbb{R}$ *be the step length determined by any strategy which ensures convergence. The iteration for the TikLGN approach* (20) *is*

$$\boldsymbol{x}_\lambda^{(k+1)} = \boldsymbol{x}_\lambda^{(k)} - \alpha_k \sum_{i=n-r_k+1}^{p} \xi_i \boldsymbol{w}_i - \alpha_k \sum_{i=p+1}^{n} g_{i-N}^{(k)} \boldsymbol{w}_i - W_1 \widehat{W}_1 (\boldsymbol{x}_\lambda^{(k)} - \bar{\boldsymbol{x}}), \tag{24}$$

with

$$\xi_i = \frac{c_{i-n+r_k} g_{i-N}^{(k)} + \alpha_k \lambda^2 s_{i-n+r_k}^2 z_i^{(k)}}{c_{i-n+r_k}^2 + \alpha_k^2 \lambda^2 s_{i-n+r_k}^2}, \qquad i = n - r_k + 1, \ldots, p,$$

where $N = \max(n-m, 0)$ *and* W_1 *and* $\widehat{W}_1$ *are defined as in Theorem 3.1.*

Proof. Let us consider the GSVD of the matrix pair (J_k, L). We initially assume that $m \geq n \geq r_k$, and that L has full rank, i.e., $\operatorname{rank}(L) = p$. Substituting the GSVD in the normal equations associated to (20)

$$(J_k^T J_k + \alpha_k^2 \lambda^2 L^T L)\boldsymbol{s} = -J_k^T \boldsymbol{r}_k - \alpha_k \lambda^2 L^T L (\boldsymbol{x}_\lambda^{(k)} - \bar{\boldsymbol{x}}),$$

one obtains

$$(D + \alpha_k^2 \lambda^2 H)\boldsymbol{y} = -\Sigma_J^T \boldsymbol{g}^{(k)} - \alpha_k \lambda^2 H \boldsymbol{z}^{(k)}, \tag{25}$$

where

$$D = \begin{bmatrix} O_{n-r_k} & & \\ & C^2 & \\ & & I_{n-p} \end{bmatrix}, \qquad H = \begin{bmatrix} I_{n-r_k} & & \\ & S^2 & \\ & & O_{n-p} \end{bmatrix},$$

$\boldsymbol{y} = W^{-1}\boldsymbol{s}$, $\boldsymbol{g}^{(k)} = U^T \boldsymbol{r}(\boldsymbol{x}_\lambda^{(k)})$, and $\boldsymbol{z}^{(k)} = W^{-1}(\boldsymbol{x}_\lambda^{(k)} - \bar{\boldsymbol{x}})$. The diagonal system (25) yields the iterate

$$\boldsymbol{x}_\lambda^{(k+1)} = \boldsymbol{x}_\lambda^{(k)} + \alpha_k \boldsymbol{s} = \boldsymbol{x}_\lambda^{(k)} - \alpha_k \sum_{i=n-r_k+1}^{p} \xi_i \boldsymbol{w}_i - \alpha_k \sum_{i=p+1}^{n} g_i^{(k)} \boldsymbol{w}_i - W_1 \widehat{W}_1 (\boldsymbol{x}_\lambda^{(k)} - \bar{\boldsymbol{x}}), \tag{26}$$

where

$$\xi_i = \frac{c_{i-n+r_k} g_i^{(k)} + \alpha_k \lambda^2 s_{i-n+r_k}^2 z_i^{(k)}}{c_{i-n+r_k}^2 + \alpha_k^2 \lambda^2 s_{i-n+r_k}^2}, \qquad i = n - r_k + 1, \ldots, p.$$

Similarly, when $r_k \leq m < n$, the TikLGN approach leads to the iterate

$$\boldsymbol{x}_\lambda^{(k+1)} = \boldsymbol{x}_\lambda^{(k)} - \alpha_k \sum_{i=n-r_k+1}^{p} \xi_i' \boldsymbol{w}_i - \alpha_k \sum_{i=p+1}^{n} g_{i-n+m}^{(k)} \boldsymbol{w}_i - W_1 \widehat{W}_1 (\boldsymbol{x}_\lambda^{(k)} - \bar{\boldsymbol{x}}), \tag{27}$$

with

$$\xi_i' = \frac{c_{i-n+r_k} g_{i-n+m}^{(k)} + \alpha_k \lambda^2 s_{i-n+r_k}^2 z_i^{(k)}}{c_{i-n+r_k}^2 + \alpha_k^2 \lambda^2 s_{i-n+r_k}^2}, \qquad i = n - r_k + 1, \ldots, p.$$

Introducing $N = n - m$ if $m < n$ and zero otherwise, the overdetermined (26) and the underdetermined (27) cases may be condensed into the single expression (24), and this completes the proof. □

In formulas (21) and (24), the solution at convergence will be denoted by $\boldsymbol{x}_\lambda$.

To ensure convergence of the above iterations, it is indispensable to control the step length for the projection term by introducing a second relaxation parameter β_k. If $L = I_n$, the new iterative method is denoted by TikGN2 and it takes the form

$$\boldsymbol{x}_\lambda^{(k+1)} = \boldsymbol{x}_\lambda^{(k)} - \alpha_k \sum_{i=1}^{r_k} \frac{\sigma_i g_i^{(k)} + \alpha_k \lambda^2 z_i^{(k)}}{\sigma_i^2 + \alpha_k^2 \lambda^2} \boldsymbol{v}_i - \beta_k V_2 V_2^T (\boldsymbol{x}_\lambda^{(k)} - \bar{\boldsymbol{x}}).$$

If a regularization matrix L is involved, the new iterative method is denoted by TikLGN2 and is given by

$$\boldsymbol{x}_\lambda^{(k+1)} = \boldsymbol{x}_\lambda^{(k)} - \alpha_k \sum_{i=n-r_k+1}^{p} \xi_i \boldsymbol{w}_i - \alpha_k \sum_{i=p+1}^{n} g_{i-N}^{(k)} \boldsymbol{w}_i - \beta_k W_1 \widehat{W}_1 (\boldsymbol{x}_\lambda^{(k)} - \bar{\boldsymbol{x}}).$$

5 The Matlab toolbox

In this section, we describe the MNGNREG toolbox, implemented in Matlab and available at https://bugs.unica.it/cana/software/#mngnreg and on GitHub at https://github.com/CaNAgroup/mngnreg. We remark that computational routines are compatible with Octave, while test scripts use some graphical features available only in Matlab. Table 1 contains the list of all files.

The algorithms described in this paper are implemented in the four main computational routines, which can be called by a similar syntax:

```
[x,k,rho,fail,X,Res,Alphas,Betas,ells] = tmngn(fun,b,x0,ell,opts)

[x,k,rho,fail,X,Res,Alphas,Betas,ells] = tmlngn(fun,b,L,x0,ell,opts)

[x,k,rho,fail,X,Res,Alphas,Betas] = tikgn(fun,b,x0,lam,opts)

[x,k,rho,fail,X,Res,Alphas,Betas] = tiklgn(fun,b,L,x0,lam,opts)
```

The input and output parameters are essentially the same: the difference between `tmngn` (`tmlngn`) and `tikgn` (`tiklgn`) is that in the first case the regularization parameter is the integer variable `ell`, while in the second case the variable `lam` is real; moreover, the regularization matrix `L` is present only in `tmlngn` and `tiklgn`.

Input arguments

- `fun` : "handle" of a function which returns the value of F and its Jacobian matrix J,
- `b` : data vector $\boldsymbol{b} \in \mathbb{R}^m$,
- `L` : regularization matrix $L \in \mathbb{R}^{p \times n}$ (in `tmlngn`, in addition to the discretization of the derivatives, the Minimum Gradient Support [11] can be set as the regularization term),
- `x0` : initial point $\boldsymbol{x}^{(0)} \in \mathbb{R}^n$ for the iterative method,
- `ell` : either rank or truncation parameter (in `tmngn` and `tmlngn`),
- `lam` : regularization parameter (in `tikgn` and `tiklgn`),
- `opts` : options (`struct` variable, every option has a default value).

routines	
`tmngn`	minimal-norm Gauss–Newton regularized by TSVD
`tmlngn`	minimal-norm Gauss–Newton regularized by TGSVD
`tikgn`	minimal-norm Gauss–Newton regularized by Tikhonov in standard form
`tiklgn`	minimal-norm Gauss–Newton regularized by Tikhonov in general form
test scripts	
`driverUnder`	test program for `tmngn.m` with beta=1 (Example 6.1)
`driverOver`	test program for `tmlngn.m` with beta=1
`driverMNGN2`	test program for `tmngn.m` (Example 6.2)
nonlinear fun	
`funUnder`	generates a simple underdetermined nonlinear problem (Example 6.1)
`funOver`	generates a simple overdetermined nonlinear problem
`nonlinfun1`	generates an underdetermined nonlinear problem (Example 6.2)
auxiliary routines	
`get_l`	compute discrete derivative operators (from REGUTOOLS)
`jack`	approximate Jacobian matrix by finite differences (from FDEMTOOLS)

Table 1: File list for the MNGNREG toolbox.

Available options are: niter, damped, dampos, findiff, rankflag, tol, alphamin, mnflag, xbar, tau, eta1, eta2, kres; see the `help` page of each function for their meaning. The option `mnflag` controls which algorithm is applied:

- `mnflag=0`, $\boldsymbol{x}^{(k+1)} = \boldsymbol{x}^{(k)} + \alpha_k \boldsymbol{s}^{(k)}$;
- `mnflag=1`, $\boldsymbol{x}^{(k+1)} = \boldsymbol{x}^{(k)} + \alpha_k \widetilde{\boldsymbol{s}}^{(k)} - \boldsymbol{t}^{(k)}$;
- `mnflag=2`, $\boldsymbol{x}^{(k+1)} = \boldsymbol{x}^{(k)} + \alpha_k \left(\widetilde{\boldsymbol{s}}^{(k)} - \boldsymbol{t}^{(k)} \right)$;
- `mnflag=3`, $\boldsymbol{x}^{(k+1)} = \boldsymbol{x}^{(k)} + \alpha_k \widetilde{\boldsymbol{s}}^{(k)} - \beta_k \boldsymbol{t}^{(k)}$, $\delta(\rho, \eta_1) = \eta_1 \rho$;
- `mnflag=4`, $\boldsymbol{x}^{(k+1)} = \boldsymbol{x}^{(k)} + \alpha_k \widetilde{\boldsymbol{s}}^{(k)} - \beta_k \boldsymbol{t}^{(k)}$, $\delta(\rho, \eta_2) = \rho^{\eta_2}$.

The values `mnflag` $= 5$ or 6 select our implementation of the algorithms discussed in [6]; see Equation (28) in Section 6.

Output arguments

- `x` : solution $\boldsymbol{x} \in \mathbb{R}^n$,
- `k` : number of iterations performed,
- `rho` : solution residual $\|F(\boldsymbol{x}) - \boldsymbol{b}\|$,
- `fail` : failure (>1) or success (0 or 1),
- `X` : solution at each iteration $X \in \mathbb{R}^{n \times k}$,
- `Res` : residual $\|F(\boldsymbol{x}^{(k)}) - \boldsymbol{b}\|$ at each iteration,
- `Alphas` : step lengths α_k for damped Gauss–Newton search direction $\widetilde{\boldsymbol{s}}$,
- `Betas` : step lengths β_k for projection vector $\boldsymbol{t}^{(k)}$,
- `ells` : numerical rank at each iteration (in `tm(l)ngn`).

The toolbox allows the use of different iterative algorithms, by setting the values of the option `mnflag`. The default choice consists of applying the MNGN2 method with automatic tuning of the parameters (`mnflag=4`), because according to our experience is the best performing one.

If the problem (5) to be solved is well-conditioned and the rank of the Jacobian is known, it can be passed to `tmngn` (`tmlngn`) via the parameter `ell`. If `ell` is not supplied (or `ell=[]`), then the automatic rank estimation discussed in Section 3.1 is activated, and the estimated rank at each iteration is returned as the output vector `ells`. This choice is the preferred one, because when the iteration approaches convergence the Jacobian often becomes rank-deficient; see [24, Section 3].

When, on the contrary, the nonlinear problem is ill-conditioned, the truncated MNGN2 (MLNGN2) algorithm is activated by assigning to `ell` the value of the regularization parameter ℓ. Under the same assumption, Tikhonov's method (20) can be applied by calling the `tikgn` or `tiklgn` functions with the parameter `lam` set to the value of λ.

The truncated MLNGN2 algorithm has been included in FDEMTOOLS, a Matlab package for direct and inverse modeling of frequency domain electromagnetic data, intended for applied geophysics applications [8, 9].

The MNGNREG package includes a copy of the routine `get_l.m` from the REGULARIZATION TOOLS by P. C. Hansen [19], and the routine `jack.m` from the above mentioned package FDEMTOOLS.

Some simple nonlinear test problems can be generated by `funUnder.m`, `funOver.m`, and `nonlinfun1.m`. The test scripts `driverUnder.m`, `driverOver.m`, and `driverMNGN2.m` run different numerical experiments, whose results will be illustrated in the next section. The content of the scripts illustrate how the computational routines should be practically used.

6 Numerical experiments

In this section, we present some numerical results obtained by applying the Matlab implementations of the MNGN2 algorithms to both well-conditioned and ill-conditioned problems. The numerical experiments were performed on an Intel Core i7 computer with 16 GB RAM, running Matlab R2024a on the Debian GNU/Linux operating system.

6.1 Well-conditioned ill-posed examples

This section is devoted to analyzing the behavior of all variants of the MNGN2 approach, under the assumption that the problem to be solved is well-conditioned in the sense discussed in Section 4. This means that the Jacobian $J = J(\boldsymbol{x})$ is substantially well-conditioned for any $\boldsymbol{x} \in D$. Since the Jacobian may become rank-deficient at convergence, in the following examples we use the automatic rank estimation provided by the toolbox, by setting `ell=[]`.

To give a geometric representation of experiments, we consider nonlinear functions with a small number of variables. In the first example we compare the MNGN method (`mnflag=1`) with the standard Gauss–Newton (GN) method (`mnflag=0`). In the second example we test the MNGN2 approaches (`mnflag=2,3,4`) and we compare them with another approach developed in [6], which we referred to as CKB, which consists of the iteration

$$\boldsymbol{x}^{(k+1)} = \boldsymbol{x}^{(k)} + \widetilde{\boldsymbol{s}}^{(k)} - \gamma_k \mathscr{P}_{\mathscr{N}(J_k)} \left(\boldsymbol{x}^{(k)} - \overline{\boldsymbol{x}} \right), \tag{28}$$

where the sequence of parameters $\gamma_k \in [0,1]$, for $k = 0, 1, \ldots$ converges to zero. The authors adopt the sequences $\gamma_k = (0.5)^{k+1}$ (`mnflag=5`) and $\gamma_k = (0.5)^{2^k}$ (`mnflag=6`).

Example 6.1. Let $F : \mathbb{R}^2 \to \mathbb{R}$ be the nonlinear function defined by

$$F(\boldsymbol{x}) = \left[p(x_1 - 1)^2 + q(x_2 - 1)^2 - 1 \right]^2, \tag{29}$$

depending upon the parameters $p, q \in \mathbb{R}$, where $\boldsymbol{x} = [x_1, x_2]^T$ and $r(\boldsymbol{x}) = F(\boldsymbol{x}) + 1$ is the residual function. In the toolbox, the function `funUnder.m` allows the user to assign different values to p and q, and thus to operate on different functions. If $p, q > 0$, or $p > 0$ and $q < 0$, then any point on the conic section

$$p(x_1 - 1)^2 + q(x_2 - 1)^2 = 1$$

is a minimum of $f(\boldsymbol{x}) = \frac{1}{2}\|r(\boldsymbol{x})\|^2$. Indeed, this least-squares problem is underdetermined, so it has infinitely many solutions. In the script `driverUnder.m` we solve it by the Gauss–Newton

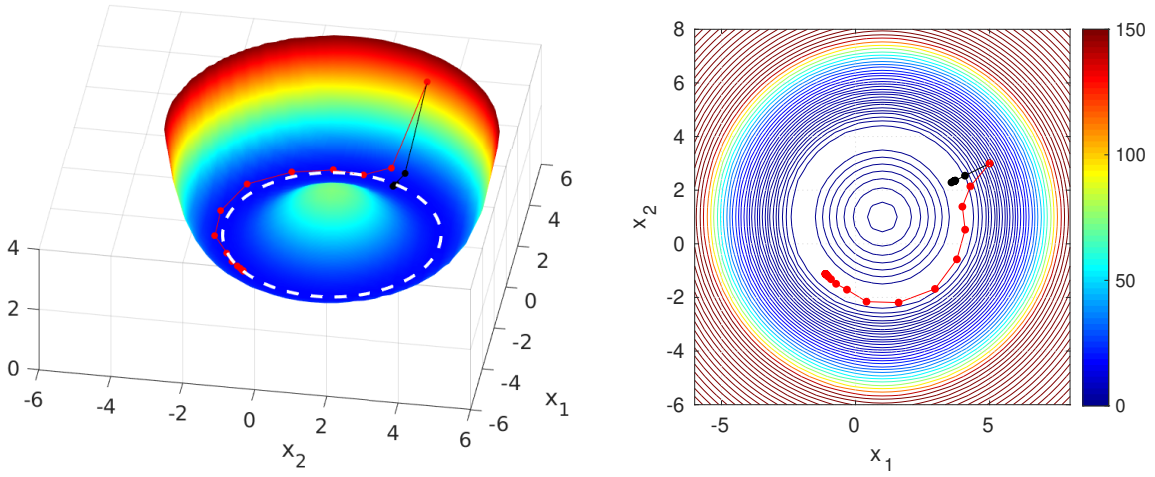

Figure 1: Convergence of problem (29) with $p = q = \frac{1}{9}$ and $\boldsymbol{x}^{(0)} = [5,3]^T$. In the 3D graph on the left, the white dashed line represents the locus of the solutions, the red dots are the iterations of the MNGN method, and the black ones correspond to the GN method. The graph on the right reports the same situation in a contour plot.

method and the minimal-norm Gauss–Newton method, setting `mnflag=0` and `mnflag=1`, respectively.

The objective function $f(\boldsymbol{x})$ can be graphically represented by a surface; see Figure 1. Considering $p = q = \frac{1}{9}$ in Equation (29), the minimal-norm solution is $\boldsymbol{x}^\dagger \approx [-1.12, -1.12]^T$, with $\|\boldsymbol{x}^\dagger\| \approx 1.58$. Figure 1 illustrates the progress of the iterates for both the MNGN and the GN methods, until they reach convergence.

We see that MNGN takes longer to converge as it must "travel" across the solutions locus to reach the minimal-norm solution. On the contrary, GN converges to the solution closer to the initial point. This fact is even clearer in the contour plot on the right of Figure 1.

Example 6.2. Let $F : \mathbb{R}^n \to \mathbb{R}^m$ be a nonlinear function such that each component is

$$F_i(\boldsymbol{x}) = S(\boldsymbol{x})\,(x_i - c_i)\,, \qquad i = 1,\dots,m, \tag{30}$$

where

$$S(\boldsymbol{x}) = \sum_{j=1}^{n} \left(\frac{x_j - c_j}{a_j}\right)^2 - 1$$

is the n-ellipsoid with center $\boldsymbol{c} = [c_1,\dots,c_n]^T$ and whose semiaxes are the components of the vector $\boldsymbol{a} = [a_1,\dots,a_n]^T$. In the software, the function `nonlinfun1.m` contains the definition of this function as well as its Jacobian matrix. The description of the Jacobian and the expression of the minimal-norm solution are given in [24] for selected values of the vectors $\boldsymbol{a}$ and $\boldsymbol{c}$.

The locus of the solutions is the union of the n-ellipsoid and the intersection between the planes $x_i = c_i$, $i = 1,\dots,m$. If $\boldsymbol{a} = [1,1,\dots,1]^T$ and $\boldsymbol{c} = [2,0,\dots,0]^T$, the minimal-norm solution is $\boldsymbol{x}^\dagger = [1,0,\dots,0]^T$.

The case $m = 2$, $n = 3$ is displayed in Figure 2, together with the iterations of the algorithms MNGN2 (`mnflag=4`) and CKB (`mnflag=5`). The script `driverMNGN2.m` produces these results. In this test, the latter algorithm converges to a solution of non-minimal norm.

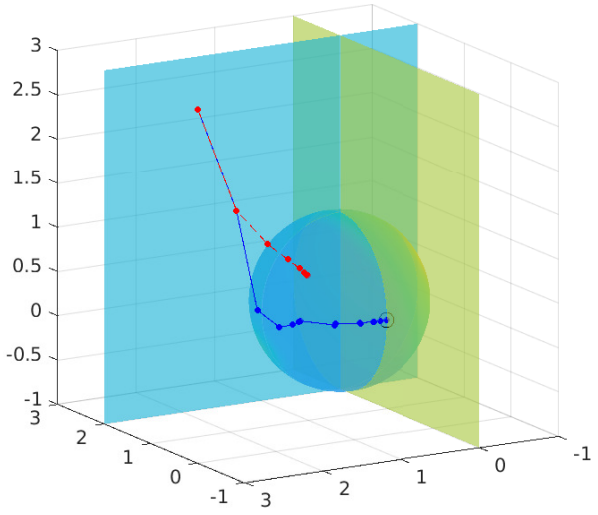

Figure 2: Solution of problem (30) for $m = 2$ and $n = 3$, with $\boldsymbol{a} = [1,1,1]^T$, $\boldsymbol{c} = [2,0,0]^T$, and $\boldsymbol{x}^{(0)} = [0,3,3]^T$. The locus of the solutions is the sphere and the line intersection of the two planes. The blue dots are the iterations of the MNGN2 (`mnflag=4`) method, and the red ones correspond to the CKB (`mnflag=5`) method. The black circle encompasses the minimal-norm solution.

Table 2 illustrates the situation with $m = 8$ and $n = 10$. We repeated the computation 100 times, varying the starting point $\boldsymbol{x}^{(0)}$ by letting its components be uniformly distributed random numbers in $(-5,5)$. We consider a numerical test a "success" if the algorithm converges according to condition (17), with stop tolerance $\tau = 10^{-8}$ and maximum number of iterations $N_{\max} = 500$. A failure is not a serious problem, in general, because non-convergence simply suggests to try a different starting vector. At the same time, a success of a method does not imply that it recovers the minimal-norm solution, as the convergence is only local. So, to give an idea of the performance of the methods, we measure over all the tests the average of both the number of iterations required and the norm of the converged solution $\|\widetilde{\boldsymbol{x}}\|$. We also report the number of successes. The MNGN2 method with `mnflag=4` is the only one which recovers the correct solution; the variant of MNGN2 corresponding to `mnflag=2` determines a reasonable solution, with norm 1.52, but with a very small number of successes. The other methods always converge, but do not recover the minimal-norm solution.

`mnflag`	iterations	$\|\widetilde{\boldsymbol{x}}\|$	#success
2	215	1.5196	12
3 ($\eta = 8$)	11	1.9911	100
4	47	1.0100	100
5	27	2.0346	100
6	11	2.0531	100

Table 2: Results for problem (30) with $m = 8$, $n = 10$, $\boldsymbol{a} = [1,\ldots,1]^T$, and $\boldsymbol{c} = [2,0,\ldots,0]^T$.

6.2 An ill-conditioned example from a real-world application

Here we consider a nonlinear model used for non-destructive soil prospection by devices based on frequency domain electromagnetic induction. The model was described by Wait in [25]. A vast literature studied the application of the model and its numerical inversion; see [3, 4, 5, 10, 11, 12, 13].

In the model, the soil is assumed to have a layered structure with n layers below ground level. Each subsoil layer is characterized by an electrical conductivity σ_k, for $k = 1, \dots, n$. The measuring device generates an electromagnetic primary field, and senses the induced secondary electromagnetic field. The model represents the ratio of the secondary to the primary field. Computing the electrical conductivity of the soil layers requires the inversion of a Hankel transform, whose unknown solution depends on the electromagnetic features of the subsoil through a nonlinear recursive formula. The Jacobian of the transformation was computed in [10, 12].

Modern instruments allow one to generate multiple measurements by selecting different device configurations. We will denote such measurements by b_i, $i = 1, \dots, m$, and the model predictions by $F_i(\boldsymbol{\sigma})$, where $\boldsymbol{\sigma} = [\sigma_1, \dots, \sigma_n]^T$. Then, the problem of data inversion consists of computing the conductivity vector $\boldsymbol{\sigma}$ such that

$$\min_{\boldsymbol{\sigma} \in \mathbb{R}^n} \|\boldsymbol{r}(\boldsymbol{\sigma})\|^2, \qquad \text{with } \boldsymbol{r}(\boldsymbol{\sigma}) = F(\boldsymbol{\sigma}) - \boldsymbol{b}.$$

This is an ill-conditioned problem, in the sense that its Jacobian is strongly ill-conditioned in each point of the investigation domain.

In the following numerical simulation, we fix the following test model for the electrical conductivity as a function of depth,

$$\sigma(z) = \mathrm{e}^{-(z-1.2)^2}. \tag{31}$$

We discretize the soil by $n = 20$ uniformly spaced layers up to the depth of 3.5m and we assign to each layer the conductivity $\sigma_i = \sigma(z_i)$, $i = 1, \dots, n$, with $z_1 = 0$m and $z_n = 3.5$m. The forward model generates a noise-free synthetic data vector $\boldsymbol{b}_{\text{exact}}$, that is perturbed by adding Gaussian noise to simulate experimental errors.

To produce the results we used the FDEMTOOLS Matlab package [8, 9], that contains the definition of the nonlinear model.

In the following, we display the solutions computed by the `tmlngn` (19) and `tiklgn` (24) methods, and compare them to two standard regularization methods, namely, TGSVD and Tikhonov. Since the model function (31) is smooth, we choose $L = D_1$ or D_2 as regularization matrices, favoring a regularizing term based on the approximation of the first or second derivatives. The regularization parameters λ and ℓ are selected by three different criteria. Minimizing the 2-norm error with respect to the exact solution aims at ascertaining the best possible performance of the methods. For simulating the inversion of experimental data, we estimate regularization parameters by the discrepancy principle and by the L-curve criterion.

In the first test, the data set is composed by $m = 20$ measurements, the noise level is $\varepsilon = 10^{-2}$. In many cases, especially when the initial vector used to initialize the iteration is close enough to the solution of the problem, the minimal-norm and the standard approaches produce similar approximations. Anyway, when the initial vector is rather far away from the solution, there are cases in which the minimal-norm methods are significantly more accurate and less sensible to the presence of local minima than the traditional approaches.

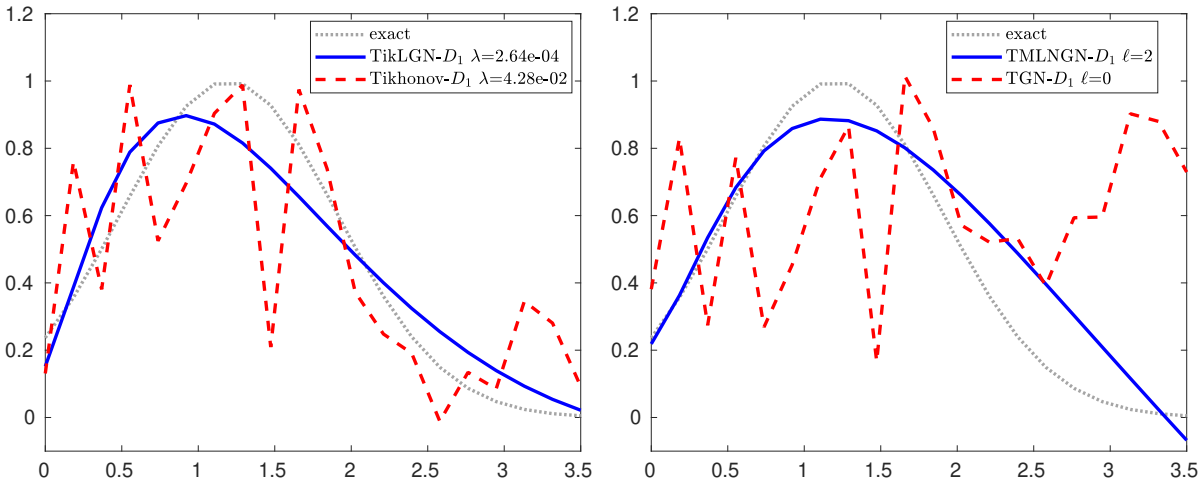

Figure 3: EM data inversion: $m = n = 20$, noise level $\varepsilon = 10^{-2}$, $L = D_1$, $\boldsymbol{\sigma}^{(0)}$ with random components in $(49.5, 50.5)$. The exact solution is compared to the solutions computed by TikLGN and the classical Tikhonov method (on the left) and by TMLNGN and by the Gauss–Newton method regularized by TGSVD, labeled as TGN, (on the right). The parameters λ and ℓ are the best possible.

Figure 3 shows one of these cases. Here the minimal-norm algorithms are compared to the traditional approaches, namely, Tikhonov regularization and the Gauss–Newton method regularized by TGSVD, labeled as TGN. In this test, the initial vector $\boldsymbol{\sigma}^{(0)}$ has random components uniformly distributed in the interval $(49.5, 50.5)$. We observe from both graphs of Figure 3 that the minimal-L-norm approaches, i.e., the `tiklgn` and the `tmlngn` methods, produce acceptable solutions, while the approximations from the standard Tikhonov approach and the TGN method are completely wrong.

In Figure 4 we illustrate the performance of the discrepancy principle and of the L-curve criterion in estimating the regularization parameters λ and ℓ. In this example, we consider $m = 10$ data values, the initial solution is $\boldsymbol{\sigma}^{(0)}$, whose components are uniformly distributed random numbers in the interval $(0.45, 0.55)$, and the noise level $\varepsilon = 10^{-4}$. The graphs in the top row concern the reconstructions obtained by the `tiklgn` and Tikhonov methods. In the bottom row we report the results obtained by the `tmlngn` and the TGN methods. The regularization parameters for the graphs in the column on the left are determined by the discrepancy principle, while the column on the right shows the reconstructions corresponding to the regularization parameters estimated by the L-curve. From all four plots, we can see that the minimal-L-norm solutions are more accurate.

7 Conclusions

In this paper, we propose a Matlab implementation of the MNGN2 method and its regularized versions. The toolbox is completed by some examples of nonlinear problems and scripts that apply the above-mentioned algorithms. We analyze their performance on various test problems: the minimal-norm methods prove to be very effective, compared to other classical approaches.

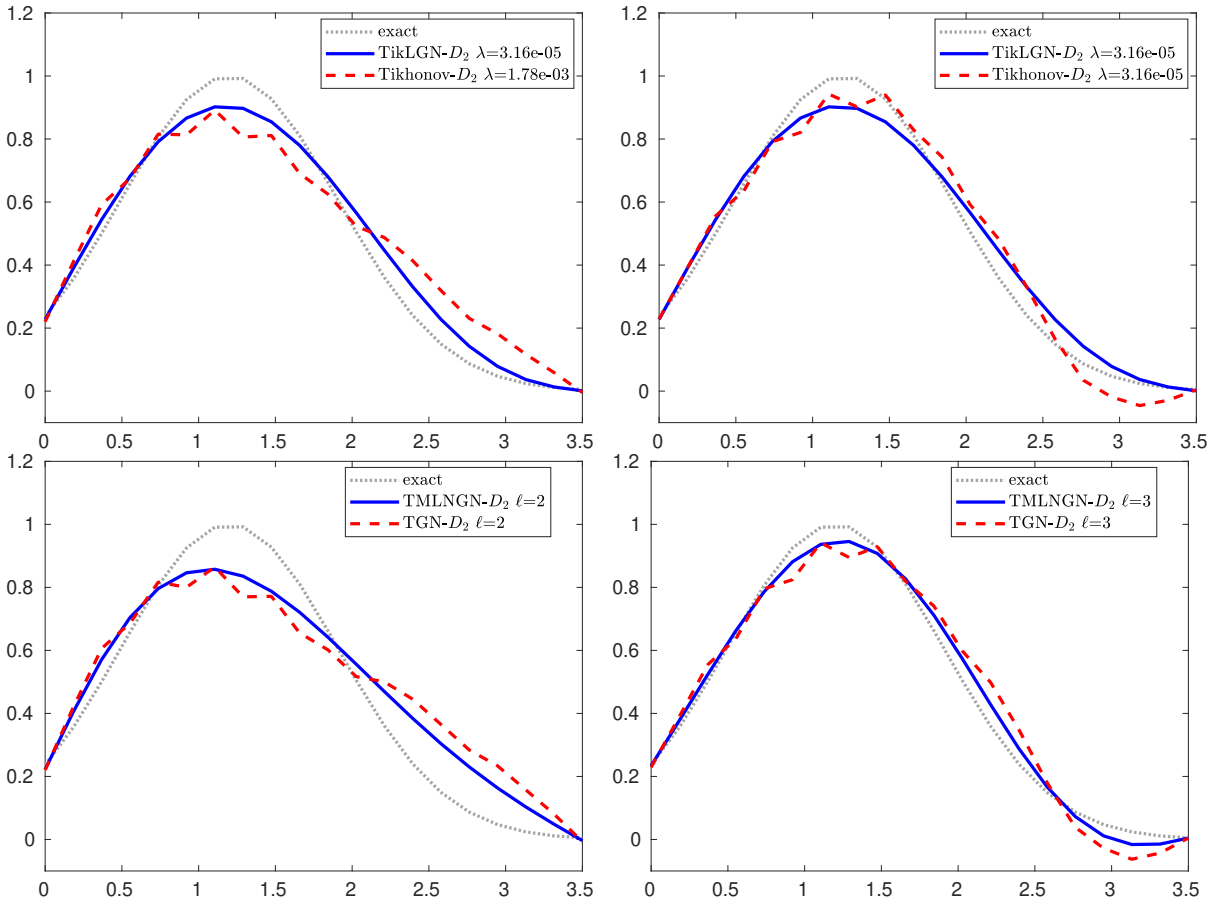

Figure 4: EM data inversion: $m = 10$, $n = 20$, noise level $\varepsilon = 10^{-4}$, $L = D_2$, $\boldsymbol{\sigma}^{(0)}$ with random components in $(0.45, 0.55)$. The exact solution is compared to the solutions computed by `tiklgn` and the classical Tikhonov method (top row) and by `tmlngn`/TGN (bottom row). The parameters λ and ℓ have been chosen by the discrepancy principle (left column) and by the L-curve (right column).

Acknowledgments

This research is partially supported by European Union-Next Generation EU, Mission 4 Component 1, CUP F53D23002700006 through the PRIN 2022 project "Inverse Problems in the Imaging Sciences (IPIS)". GR acknowledges partial support from the PRIN-PNRR 2022 project "AQuAInt - Approximation and Quadrature for Applicative Integral Models" (P20229RMLB). FP is partially supported by the INdAM-GNCS 2025 Project "Tecniche numeriche per problemi di grandi dimensioni" (CUP E53C24001950001) and GR by the INdAM-GNCS 2025 Project "Metodi di approssimazione globale per operatori integrali e applicazioni alle equazioni funzionali" (CUP E53C24001950001). The authors are members of the GNCS group of INdAM. FP gratefully acknowledges OGS and CINECA under HPC-TRES program award number 2024-02.

Declaration of interests

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

References

[1] L. Armijo. *Minimization of functions having Lipschitz continuous first partial derivatives*. Pac. J. Math., vol. 16, 1, (1966), 1–3. URL http://dx.doi.org/10.2140/pjm.1966.16.1

[2] Å. Björck. *Numerical Methods for Least Squares Problems*. Philadelphia: SIAM. URL http://dx.doi.org/10.1137/1.9781611971484

[3] J. Boaga, M. Ghinassi, A. D'Alpaos, G. P. Deidda, G. Rodriguez, and G. Cassiani. *Geophysical investigations unravel the vestiges of ancient meandering channels and their dynamics in tidal landscapes*. Sci. Rep., vol. 8, (2018), 1708 (8 pages). URL http://dx.doi.org/10.1038/s41598-018-20061-5

[4] A. Buccini and P. Díaz de Alba. *A variational non-linear constrained model for the inversion of FDEM data*. Inverse Problems, vol. 38, 1, (2022), 014001. URL http://dx.doi.org/10.1088/1361-6420/ac3c54

[5] A. Buccini, P. Díaz de Alba, and F. Pes. *An Alternating Direction Multiplier Method for the inversion of FDEM data*. J. Sci. Comput., vol. 101, (2024), 14. URL http://dx.doi.org/https://doi.org/10.1007/s10915-024-02652-9

[6] S. L. Campbell, P. Kunkel, and K. Bobinyec. *A minimal norm corrected underdetermined Gauß–Newton procedure*. Appl. Numer. Math., vol. 62, 5, (2012), 592–605. URL http://dx.doi.org/10.1016/j.apnum.2012.01.006

[7] A. Concas, S. Noschese, L. Reichel, and G. Rodriguez. *A spectral method for bipartizing a network and detecting a large anti-community*. J. Comput. Appl. Math., vol. 373, (2020), 112306 (15 pages). URL http://dx.doi.org/10.1016/j.cam.2019.06.022

[8] G. P. Deidda, P. Díaz de Alba, C. Fenu, G. Lovicu, and G. Rodriguez. *FDEMtools: a MATLAB package for FDEM data inversion*. Numer. Algorithms, vol. 84, (2020), 1313–1327. URL http://dx.doi.org/10.1007/s11075-019-00843-2

[9] G. P. Deidda, P. Díaz de Alba, F. Pes, and G. Rodriguez. *Forward Electromagnetic Induction Modelling in a Multilayered Half-Space: An Open-Source Software Tool*. Remote Sens., vol. 15, 7, (2023), 1772. URL http://dx.doi.org/10.3390/rs15071772

[10] G. P. Deidda, P. Díaz de Alba, and G. Rodriguez. *Identifying the magnetic permeability in multi-frequency EM data inversion*. Electron. Trans. Numer. Anal., vol. 47, (2017), 1–17. URL https://www.emis.de/journals/ETNA/volumes/2011-2020/vol47/abstract_vol47_pp1-17.html?vol=47&pages=1-17

[11] G. P. Deidda, P. Díaz de Alba, G. Rodriguez, and G. Vignoli. *Inversion of Multiconfiguration Complex EMI Data with Minimum Gradient Support Regularization: A Case Study*. Math. Geosci., vol. 52, 7, (2020), 945–970. URL http://dx.doi.org/10.1007/s11004-020-09855-4

[12] G. P. Deidda, C. Fenu, and G. Rodriguez. *Regularized solution of a nonlinear problem in electromagnetic sounding*. Inverse Problems, vol. 30, (2014), 125014 (27 pages). URL http://dx.doi.org/10.1088/0266-5611/30/12/125014

[13] G. Dragonetti, A. Comegna, A. Ajeel, G. P. Deidda, N. Lamaddalena, G. Rodriguez, G. Vignoli, and A. Coppola. *Calibrating electromagnetic induction conductivities with time-domain reflectometry measurements*. Hydrol. Earth. Syst. Sc., vol. 22, (2018), 1509–1523. URL http://dx.doi.org/10.5194/hess-22-1509-2018

[14] H. W. Engl, M. Hanke, and A. Neubauer. *Regularization of Inverse Problems*. Dordrecht: Kluwer. ISBN 0-7923-4157-0. URL https://link.springer.com/book/9780792341574

[15] J. Eriksson and P. A. Wedin. *Regularization Methods for Nonlinear Least Squares Problems. Part I: Exactly Rank-deficient Problems*. Tech. Rep., Umeå University, Sweden (1996)

[16] J. Eriksson, P. A. Wedin, M. E. Gulliksson, and I. Söderkvist. *Regularization methods for uniformly rank-deficient nonlinear least-squares problems*. J. Optim. Theory Appl., vol. 127, (2005), 1–26. URL http://dx.doi.org/10.1007/s10957-005-6389-0

[17] A. A. Goldstein. *Constructive Real Analysis*. Harper and Row. ISBN 9780063561847

[18] P. C. Hansen. *Rank–Deficient and Discrete Ill–Posed Problems*. Philadelphia: SIAM. URL http://dx.doi.org/10.1137/1.9780898719697

[19] P. C. Hansen. *Regularization Tools: version 4.0 for Matlab 7.3*. Numer. Algorithms, vol. 46, (2007), 189–194. URL http://dx.doi.org/10.1007/s11075-007-9136-9

[20] P. C. Hansen, V. Pereyra, and G. Scherer. *Least Squares Data Fitting with Applications*. Baltimore: Johns Hopkins University Press. ISBN 9781421408583. URL http://dx.doi.org/10.1353/book.21076

[21] J. M. Ortega and W. C. Rheinboldt. *Iterative Solution of Nonlinear Equations in Several Variables*. Philadelphia: SIAM. URL http://dx.doi.org/10.1137/1.9780898719468

[22] F. Pes. *Truncated Minimal-Norm Gauss–Newton Method Applied to the Inversion of FDEM Data*. O. Gervasi, B. Murgante, A. M. A. C. Rocha, C. Garau, F. Scorza, Y. Karaca, and C. M. Torre (eds.), *Computational Science and Its Applications – ICCSA 2023 Workshops*. Cham: Springer Nature Switzerland. ISBN 978-3-031-37117-2, 641–658. URL http://dx.doi.org/10.1007/978-3-031-37117-2_43

[23] F. Pes and G. Rodriguez. *The minimal-norm Gauss-Newton method and some of its regularized variants*. Electron. Trans. Numer. Anal., vol. 53, (2020), 459–480. URL http://dx.doi.org/10.1553/etna_vol53s459

[24] F. Pes and G. Rodriguez. *A doubly relaxed minimal-norm Gauss–Newton method for underdetermined nonlinear least-squares problems*. Appl. Numer. Math., vol. 171, (2022), 233–248. URL http://dx.doi.org/10.1016/j.apnum.2021.09.002

[25] J. Wait. *Geo-Electromagnetism*. New York: Academic Press. ISBN 978-0-12-730880-7. URL http://dx.doi.org/10.1016/B978-0-12-730880-7.X5001-7