Journal of Scientific Computing manuscript No.
(will be inserted by the editor)

An Efficient Implementation of the Gauss-Newton Method via Generalized Krylov Subspaces

Alessandro Buccini · Patricia Díaz de Alba · Federica Pes · Lothar Reichel

Received: date / Accepted: date

Abstract The solution of nonlinear inverse problems is a challenging task in numerical analysis. In most cases, this kind of problems is solved by iterative procedures that, at each iteration, linearize the problem in a neighborhood of the currently available approximation of the solution. The linearized problem is then solved by a direct or iterative method. Among this class of solution methods, the Gauss-Newton method is one of the most popular ones. We propose an efficient implementation of this method for large-scale problems. Our implementation is based on projecting the nonlinear problem into a sequence of nested subspaces, referred to as Generalized Krylov Subspaces (GKS), whose dimension increases with the number of iterations, except for when restarts are carried out. When the computation of the Jacobian matrix is expensive, we combine our iterative method with secant (Broyden) updates to further reduce the computational cost. We show convergence of the proposed solution methods and provide a few numerical examples that illustrate their performance.

A. Buccini
Department of Mathematics and Computer Science, University of Cagliari,
09124 Cagliari, Italy
E-mail: alessandro.buccini@unica.it

P. Díaz de Alba
Department of Mathematics, University of Salerno,
84084 Fisciano, Italy
E-mail: pdiazdealba@unisa.it

F. Pes
Department of Chemistry and Industrial Chemistry, University of Pisa,
56124 Pisa, Italy
E-mail: federica.pes@dcci.unipi.it

L. Reichel
Department of Mathematical Sciences, Kent State University,
Kent, Ohio 44242, USA
E-mail: reichel@math.kent.edu

Keywords nonlinear inverse problems · Gauss-Newton method · generalized Krylov subspace

Mathematics Subject Classification (2020) 65K10 · 49M15 · 65J15

1 Introduction

This paper is concerned with the solution of nonlinear least-squares problems of the form

$$\min_{\mathbf{x}\in\mathbb{R}^n} \|\mathbf{y} - f(\mathbf{x})\|^2 , \tag{1.1}$$

where $f : \mathbb{R}^n \to \mathbb{R}^m$ is a differentiable nonlinear function, $\mathbf{x} \in \mathbb{R}^n$ denotes the unknown vector that we would like to determine, $\mathbf{y} \in \mathbb{R}^m$ represents measured data, and $\|\cdot\|$ stands for the Euclidean vector norm. This kind of problems arises in several fields of science and engineering, such as in image reconstruction and geophysics. In some applications, the solution of the problem (1.1) is very sensitive to perturbations, e.g., to errors in the data vector $\mathbf{y}$; these errors may, for instance, be caused by measurement inaccuracies. In this paper, we only consider problems that are not overly sensitive to perturbations; if the given problem is very sensitive to perturbations in $\mathbf{y}$, then we assume that the problem has been regularized to make the solution less sensitive. For an example of well-posed problem of the form (1.1) we refer the reader to the so-called Bratu problem; see, e.g., [29] and Section 4. If the problem is ill-posed, a simple approach to regularize it is to use Tikhonov regularization (see, e.g., [21]), i.e., to substitute (1.1) with

$$\min_{\mathbf{x}\in\mathbb{R}^n} \left\{ \|\mathbf{y} - f(\mathbf{x})\|^2 + \mu \|\mathbf{x}\|^2 \right\},$$

for a suitable $\mu > 0$. In the following, we will assume that this substitution, if required, has been performed.

We consider the Gauss-Newton method [20, 34] for the solution of (1.1). This is an iterative method that at each step solves a linear least-squares problem obtained from a first-order Taylor expansion of the nonlinear function $r(\mathbf{x}) = \mathbf{y} - f(\mathbf{x})$. Convergence is assured thanks to the use of the Armijo-Goldstein condition; see below. In case the problem (1.1) is underdetermined, i.e., when n is larger than m, the solution is not unique. Pes and Rodriguez [32, 33] describe a modification of the Gauss-Newton method that can be applied in this situation to determine the minimal-norm solution; also some regularized variants of the Gauss-Newton method are described.

Nevertheless, we would like to briefly discuss the situation when the Jacobian matrix is severely ill-conditioned or even rank-deficient. Since we project a linearization of the (large) problem (1.1) into solution subspaces of fairly small dimensions, the projected problems usually are better conditioned than the original large problem. As it will be illustrated by numerical examples in Section 4, we are able to quite accurately solve ill-conditioned problems, when the data $\mathbf{y}$ are not corrupted by noise, by employing as the only type

of regularization the projection into a small linear subspace. This assumption corresponds to committing an inverse crime. Nevertheless, we illustrate that our solution method is less sensitive to ill-conditioning than the standard Gauss-Newton method, and that, when the problem (1.1) is severely ill-conditioned, the standard Gauss-Newton method may fail to determine an accurate solution even under the scenario of an inverse crime.

The main computational effort of the Gauss-Newton method is the solution of linear least-squares problems of possibly large dimensions at each iteration. We would like to address this issue by using Generalized Krylov Subspaces (GKS). The subspaces we consider were first applied by Lampe et al. [30] to the solution of large linear discrete ill-posed problems by Tikhonov regularization. Subsequently they were used for the solution of nonconvex minimization problems in imaging and statistics; see, e.g., [9, 10, 27, 31]. The approximate solution $\mathbf{x}^{(k)} \in \mathbb{R}^n$ computed at iteration k lives in a generalized Krylov subspace $\mathcal{V}_{k-1}$, and a projected residual error is included in the next generalized Krylov subspace $\mathcal{V}_k$. The solution subspaces generated in this manner are nested and typically contain accurate approximations of the desired solution of (1.1) already when they are of fairly small dimension; in particular, the use of generalized Krylov subspaces performs better than using standard Krylov subspaces; see [7, 31] for illustrations.

Another possible bottleneck of the Gauss-Newton method for large-scale problems is the computation of the Jacobian matrix at each iteration. If this is very time-consuming, then instead of forming the Jacobian by evaluating all of its entries at each iteration, we approximate the Jacobian by using a secant (Broyden) update; see [6, 20]. When the iterates $\mathbf{x}^{(k)}$ and $\mathbf{x}^{(k+1)}$ are close, the secant update allows us to approximate the Jacobian of f at $\mathbf{x}^{(k+1)}$ by the Jacobian (or an approximation thereof) at $\mathbf{x}^{(k)}$ plus a low-rank correction. Therefore, we update the Jacobian for $\tilde{k}$ iterations, and compute the Jacobian "from scratch" only when $k \equiv 0 \bmod \tilde{k}$.

Finally, we observe that, if many iterations of the algorithms are performed, the dimension of the space $\mathcal{V}_k$ may increase significantly and this, in turn, can slow down the computations. To avoid this situation, we propose a restarted variant based on the algorithms proposed in [11]. Every k_{rest} iterations the space $\mathcal{V}_k$ is restarted, i.e., it is replaced by a one-dimensional space; see below. This strategy ensures that the dimension of the space $\mathcal{V}_k$ is bounded regardless of the number of iterations. We will illustrate that this reduces the computing time.

It is the purpose of this paper to show that projection into generalized Krylov subspaces may be beneficial when solving large nonlinear least-squares problems (1.1). This approach is compared to a standard implementation of the Gauss-Newton method. We remark that many approaches to Newton-Krylov methods have been discussed in the literature; among the first papers on this topic are those written by Brown and Saad [4, 5], who consider the application of Arnoldi methods, which requires $m = n$. Darvishi and Shin [13] and Kan et al. [28] provide more recent contributions. However, these methods

do not focus on nonlinear least-squares problems and, moreover, consider the application of standard Krylov subspaces.

This paper is organized as follows. Section 2 briefly recalls the classical Gauss-Newton method. Our new solution approaches are described in Section 3 and there we also show some of properties of these methods. Section 4 presents a few numerical examples that illustrate the performance of our methods, and concluding remarks can be found in Section 5.

2 The Gauss-Newton method

This section reviews the classical Gauss-Newton method applied to the solution of nonlinear least-squares problems (1.1). Let $r(\mathbf{x}) = \mathbf{y} - f(\mathbf{x})$ be the residual function, where $\mathbf{x} \in \mathbb{R}^n$ represents an approximate solution of the problem (1.1), the function $f : \mathbb{R}^n \to \mathbb{R}^m$ is nonlinear and Fréchet differentiable, and the vector $\mathbf{y} \in \mathbb{R}^m$ represents available data. We disregard for the moment that in some applications, the problem (1.1) might be severely ill-conditioned. The Gauss-Newton method computes the solution $\mathbf{x}^*$ of (1.1) by minimizing the Euclidean norm of the residual $r(\mathbf{x})$, i.e., it solves

$$\mathbf{x}^* = \arg\min_{\mathbf{x}\in\mathbb{R}^n} \|r(\mathbf{x})\|^2, \tag{2.1}$$

where, for simplicity, let us assume that the solution $\mathbf{x}^*$ is unique. The Gauss-Newton method is an iterative procedure that, at each iteration, linearizes the problem (1.1) and minimizes the norm of the residual of the linearized problem. Since we assumed that f is Fréchet differentiable, it follows that $r(\mathbf{x})$ satisfies this property as well. Therefore, we use the approximation

$$r\left(\mathbf{x}^{(k+1)}\right) \simeq r\left(\mathbf{x}^{(k)}\right) + J^{(k)}\mathbf{q}^{(k)},$$

where $\mathbf{x}^{(k)}$ is the current approximation of $\mathbf{x}^*$. The vector $\mathbf{q}^{(k)}$ is referred to as the step, and $J^{(k)} = J(\mathbf{x}^{(k)}) \in \mathbb{R}^{m\times n}$ represents the Jacobian matrix of $r(\mathbf{x}) = [r_1(\mathbf{x}), \ldots, r_m(\mathbf{x})]^T$ at $\mathbf{x} = \mathbf{x}^{(k)}$. It is defined by

$$[J(\mathbf{x})]_{ij} = \frac{\partial r_i(\mathbf{x})}{\partial x_j}, \quad i = 1, \ldots, m, \quad j = 1, \ldots, n.$$

Let $\mathbf{x}^{(0)} \in \mathbb{R}^n$ be an initial approximation of the solution $\mathbf{x}^*$ of the nonlinear least-squares problem (2.1). We determine, for $k = 0, 1, 2, \ldots$, the step $\mathbf{q}^{(k)}$ by solving the linear least-squares problem

$$\min_{\mathbf{q}\in\mathbb{R}^n} \left\| r\left(\mathbf{x}^{(k)}\right) + J^{(k)}\mathbf{q} \right\|^2,$$

and we compute the next approximation $\mathbf{x}^{(k+1)}$ of $\mathbf{x}^*$ as

$$\mathbf{x}^{(k+1)} = \mathbf{x}^{(k)} + \alpha^{(k)}\mathbf{q}^{(k)}, \qquad k = 0, 1, 2, \ldots ,$$

where the coefficient $\alpha^{(k)}$ is determined according to the Armijo-Goldstein principle, see below, to ensure convergence to a stationary point of

$$\mathcal{J}(\mathbf{x}) = \|r(\mathbf{x})\|^2 ;$$

see, e.g., [20,25] for discussions. Since the functional $\mathcal{J}$ might not be convex, it is in general not possible to show that the iterates $\mathbf{x}^{(k)}$ converge to a minimizer of $\mathcal{J}$. We remark that if one knows that the solution $\mathbf{x}^*$ is nonnegative, then it is possible to determine $\alpha^{(k)}$ such that $\mathbf{x}^{(k)} \geq 0$ for all k. We will not dwell on this situation here, but we note that it is straightforward to extend our solution method to compute nonnegative solutions.

We say that $\alpha^{(k)}$ satisfies the Armijo-Goldstein condition if

$$\left\| r\left(\mathbf{x}^{(k)}\right) \right\|^2 - \left\| r\left(\mathbf{x}^{(k)} + \alpha^{(k)}\mathbf{q}^{(k)}\right) \right\|^2 \geq \frac{1}{2}\alpha^{(k)} \left\| J^{(k)}\mathbf{q}^{(k)} \right\|^2 . \tag{2.2}$$

To determine a suitable $\alpha^{(k)}$, we apply a line search. Let $\alpha_0^{(k)} > 0$ be large enough. If $\alpha_0^{(k)}$ satisfies the condition (2.2), then we set $\alpha^{(k)} = \alpha_0^{(k)}$, otherwise we let

$$\alpha_1^{(k)} = \frac{\alpha_0^{(k)}}{2}.$$

We iterate in this manner until we find an $\alpha_j^{(k)}$ that satisfies the condition (2.2) and then set $\alpha^{(k)} = \alpha_j^{(k)}$. The Gauss-Newton method with line search based on the Armijo-Goldstein condition yields a converging sequence of iterates $\mathbf{x}^{(k)}$, $k = 0, 1, 2, \ldots$, for any differentiable function f; see [25] for a proof. Algorithm 1 summarizes the computations.

Theorem 1 *Let $f : \mathbb{R}^n \to \mathbb{R}^m$ be a Fréchet differentiable function and let $\mathbf{x}^{(k)}$, $k = 1, 2, \ldots$, denote the iterates generated by Algorithm 1. There exists $\mathbf{x}^*$ such that*

$$\lim_{k\to\infty} \left\| \mathbf{x}^{(k)} - \mathbf{x}^* \right\| = 0,$$

and $\mathbf{x}^*$ is a stationary point of $\mathcal{J}(\mathbf{x}) = \|\mathbf{y} - f(\mathbf{x})\|^2$. If $\mathcal{J}$ is convex, then $\mathbf{x}^*$ is a global minimizer of $\mathcal{J}$. Moreover, if $\mathcal{J}$ is strictly convex, then there is a unique global minimizer that coincides with $\mathbf{x}^*$.

Proof Proofs can be found in [3, 25].

3 The Gauss-Newton method in generalized Krylov subspaces

We would like to reduce the computational cost of the iterations with Algorithm 1 when applied to the solution of large-scale problems (1.1). We achieve this by determining approximate solutions of the problem (2.1) in a sequence of fairly low-dimensional solution subspaces $\mathcal{V}_k$, $k = 1, 2, \ldots$, whose dimensions increase with k. At iteration k, we first compute the approximation $\mathbf{x}^{(k+1)} \in \mathcal{V}_k$ of the solution $\mathbf{x}^*$ of (2.1), and then expand the solution subspace by including

Algorithm 1: The classical Gauss-Newton method

input : Nonlinear function f, data $\mathbf{y}$, initial guess for the approximate solution $\mathbf{x}^{(0)}$, initial guess for the damping parameter α_0, tolerance τ, maximum number of iterations K.

output: Approximate solution $\mathbf{x}^*$.

Define the function $r(\mathbf{x}) = \mathbf{y} - f(\mathbf{x})$;
for $k = 0, 1, 2, \ldots, K$ **do**
  Compute the Jacobian $J^{(k)}$ of f in $\mathbf{x}^{(k)}$;
  $\mathbf{q}^{(k)} = \arg\min_{\mathbf{q}\in\mathbb{R}^n} \left\| r\left(\mathbf{x}^{(k)}\right) + J^{(k)}\mathbf{q} \right\|^2$;
  $\alpha_0^{(k)} = \alpha_0$;
  $j = 0$;
  while $\left\| r\left(\mathbf{x}^{(k)}\right) \right\|^2 - \left\| r\left(\mathbf{x}^{(k)} + \alpha_j^{(k)}\mathbf{q}^{(k)}\right) \right\|^2 \leq \frac{1}{2}\alpha_j^{(k)} \left\| J^{(k)}\mathbf{q}^{(k)} \right\|^2$ **do**
    $\alpha_{j+1}^{(k)} = \frac{\alpha_j^{(k)}}{2}$;
    $j = j + 1$;
  $\alpha^{(k)} = \alpha_j^{(k)}$;
  $\mathbf{x}^{(k+1)} = \mathbf{x}^{(k)} + \alpha^{(k)}\mathbf{q}^{(k)}$;
  if $\left\| \mathbf{x}^{(k)} - \mathbf{x}^{(k+1)} \right\| \leq \tau \left\| \mathbf{x}^{(k)} \right\|$ **then**
    exit;
$\mathbf{x}^* = \mathbf{x}^{(k+1)}$;

an appropriately chosen vector in $\mathcal{V}_k$. This defines the next solution subspace $\mathcal{V}_{k+1}$. Since $d_k = \dim(\mathcal{V}_k) \ll n$, the computation of $\mathbf{x}^{(k+1)}$ is less expensive than computing a new iterate with the standard Gauss-Newton method, which seeks to determine a solution in $\mathbb{R}^n$.

Assume that at the k-th step an approximation $\mathbf{x}^{(k)} \in \mathcal{V}_{k-1} \subset \mathcal{V}_k$ of the solution $\mathbf{x}^*$ is known. Let the columns of the matrix $V_k \in \mathbb{R}^{n\times d_k}$ form an orthonormal basis for the subspace $\mathcal{V}_k$. Then

$$V_k^T V_k = I_{d_k} \quad \text{and} \quad \mathbf{x}^{(k)} = V_k \mathbf{z}^{(k)},$$

where I_{d_k} denotes the identity matrix of order d_k, $\mathbf{z}^{(k)} \in \mathbb{R}^{d_k}$, and the last entry of $\mathbf{z}^{(k)}$ vanishes since $\mathbf{z}^{(k)} \in \mathcal{V}_{k-1}$.

The residual at the k-th step is given by

$$r\left(V_k\mathbf{z}^{(k)}\right) = \mathbf{y} - f\left(\mathbf{x}^{(k)}\right) = \mathbf{y} - f\left(V_k\mathbf{z}^{(k)}\right). \tag{3.1}$$

We determine a new approximate solution $\mathbf{x}^{(k+1)} \in \mathcal{V}_k$ as

$$\mathbf{x}^{(k+1)} = V_k\mathbf{z}^{(k+1)},$$

where the vector $\mathbf{z}^{(k+1)}$ is computed by applying the Gauss-Newton method to (3.1), i.e.,

$$\mathbf{z}^{(k+1)} = \mathbf{z}^{(k)} + \alpha^{(k)}\mathbf{q}^{(k)}, \tag{3.2}$$

and the step $\mathbf{q}^{(k)}$ is obtained by solving the least-squares problem

$$\min_{\mathbf{q}\in\mathbb{R}^{d_k}} \left\| r\left(V_k\mathbf{z}^{(k)}\right) + J^{(k)}\mathbf{q} \right\|^2.$$

With slight abuse of notation, we denote the step by $\mathbf{q}$, similarly as in the previous section, even though the vectors $\mathbf{q}$ are of different dimensions; in this section, the dimension depends on k. The matrix $J^{(k)} \in \mathbb{R}^{m\times d_k}$ represents the Jacobian of r in (3.1) evaluated at the point $\mathbf{z}^{(k)}$. We have

$$J^{(k)} = -J_f\left(V_k\mathbf{z}^{(k)}\right)V_k,$$

where J_f denotes the Jacobian matrix of the nonlinear function f. To ensure convergence, the parameter $\alpha^{(k)}$ in (3.2) is determined by applying the Armijo-Goldstein principle, i.e., $\alpha^{(k)}$ is the largest member of the sequence $\alpha^{(0)}2^{-i}$, $i = 0, 1, \dots$, for which

$$\left\|r\left(V_k\mathbf{z}^{(k)}\right)\right\|^2 - \left\|r\left(V_k\left(\mathbf{z}^{(k)} + \alpha^{(k)}\mathbf{q}^{(k)}\right)\right)\right\|^2 \geq \frac{1}{2}\alpha^{(k)}\left\|J^{(k)}\mathbf{q}^{(k)}\right\|^2.$$

In our numerical experiments we set $\alpha^{(0)} = 1$.

Once the new iterate $\mathbf{x}^{(k+1)}$ has been evaluated, we enlarge the solution subspace $\mathcal{V}_k$ by determining a vector $\mathbf{v}^{(k+1)}$ that is orthogonal to $\mathcal{V}_k$. To this end, let us first compute the Jacobian $J_f\left(\mathbf{x}^{(k+1)}\right)$ of f at $\mathbf{x}^{(k+1)}$. This computation also is required for determining $\mathbf{x}^{(k+2)}$ and, therefore, is carried out only once. Let

$$\mathbf{g}^{(k+1)} = J_f\left(\mathbf{x}^{(k+1)}\right)^T\left(\mathbf{y} - f\left(\mathbf{x}^{(k)}\right)\right).$$

The intuition is that $\mathbf{g}^{(k+1)}$ is the residual of the normal equations associated with the linearized problem. However, in general, this vector is not orthogonal to the subspace $\mathcal{V}_k$. We therefore explicitly orthogonalize it to this subspace, i.e., we compute

$$\tilde{\mathbf{g}}^{(k+1)} = \mathbf{g}^{(k+1)} - V_kV_k^T\mathbf{g}^{(k+1)},$$

and then normalize the vector so obtained,

$$\mathbf{v}^{(k+1)} = \frac{\tilde{\mathbf{g}}^{(k+1)}}{\left\|\tilde{\mathbf{g}}^{(k+1)}\right\|}.$$

We store an orthonormal basis for $\mathcal{V}_{k+1}$ as columns of the matrix $V_{k+1} = \left[V_k, \mathbf{v}^{(k+1)}\right] \in \mathbb{R}^{n\times(d_k+1)}$.

Although it is an extremely rare event when solving problems that arise from a real application, it is possible that $\tilde{\mathbf{g}}^{(k+1)} = \mathbf{0}$, i.e., that $\mathbf{g}^{(k+1)} \in \mathcal{V}_k$. In this case, we simply do not enlarge the solution subspace in this iteration and proceed with the computations. Differently from Krylov methods for the solution of linear systems of equations, such as GMRES, this is not a case of a "lucky breakdown" and one generally cannot terminate the iterations.

The last point to address is how to determine V_0, i.e., the basis for the initial subspace. In our computed examples, we will choose an initial approximate solution $\mathbf{x}^{(0)} \neq \mathbf{0}$ and set

$$V_0 = \frac{\mathbf{x}^{(0)}}{\left\|\mathbf{x}^{(0)}\right\|}.$$

The computations are summarized by Algorithm 2.

Algorithm 2: A Gauss-Newton method using generalized Krylov subspaces

input : Nonlinear function f, data vector $\mathbf{y}$, initial guess for the approximate solution $\mathbf{x}^{(0)}$, initial guess for the damping parameter α_0, tolerance τ, maximum number of iterations K.

output: Approximation $\widetilde{\mathbf{x}}^*$ of the solution $\mathbf{x}^*$.

$V_0 = \mathbf{x}^{(0)} / \left\|\mathbf{x}^{(0)}\right\|$;

$\mathbf{z}^{(0)} = \left\|\mathbf{x}^{(0)}\right\|$;

Compute the Jacobian J_f of f at $\mathbf{x}^{(0)}$;

for $k = 0, 1, 2, \ldots, K$ **do**

  $J^{(k)} = -J_f V_k$;

  $\mathbf{r}^{(k)} = \mathbf{y} - f\left(V_k \mathbf{z}^{(k)}\right)$;

  $\mathbf{q}^{(k)} = \arg \min_{\mathbf{q} \in \mathbb{R}^{d_k}} \left\|\mathbf{r}^{(k)} + J^{(k)}\mathbf{q}\right\|^2$;

  $\alpha_0^{(k)} = \alpha_0$;

  $j = 0$;

  while $\left\|r\left(V_k \mathbf{z}^{(k)}\right)\right\|^2 - \left\|r\left(V_k\left(\mathbf{z}^{(k)} + \alpha_j^{(k)}\mathbf{q}^{(k)}\right)\right)\right\|^2 \leq \frac{1}{2}\alpha_j^{(k)} \left\|J^{(k)}\mathbf{q}^{(k)}\right\|^2$ **do**

    $\alpha_{j+1}^{(k)} = \frac{\alpha_j^{(k)}}{2}$;

    $j = j + 1$;

  $\alpha^{(k)} = \alpha_j^{(k)}$;

  $\mathbf{z}^{(k+1)} = \mathbf{z}^{(k)} + \alpha^{(k)}\mathbf{q}^{(k)}$;

  if $\left\|\mathbf{z}^{(k)} - \mathbf{z}^{(k+1)}\right\| \leq \tau \left\|\mathbf{z}^{(k)}\right\|$ **then**

    exit;

  Compute the Jacobian J_f of f at $V_k \mathbf{z}^{(k+1)}$;

  $\mathbf{g}^{(k+1)} = J_f^T \mathbf{r}^{(k)}$;

  $\tilde{\mathbf{g}}^{(k+1)} = \mathbf{g}^{(k+1)} - V_k V_k^T \mathbf{g}^{(k+1)}$;

  $V_{k+1} = \left[V_k,\ \tilde{\mathbf{g}}^{(k+1)} / \left\|\tilde{\mathbf{g}}^{(k+1)}\right\|\right]$;

  $\mathbf{z}^{(k+1)} = \left[\mathbf{z}^{(k+1)}\ 0\right]^T$;

$\widetilde{\mathbf{x}}^* = V_k \mathbf{z}^{(k+1)}$;

We are now in position to show some theoretical results.

Lemma 1 *Let f be a Fréchet differentiable function. With the notation of Algorithm 2, it holds that for every k, there is a finite positive integer j such that*

$$\left\|r\left(V_k \mathbf{z}^{(k)}\right)\right\|^2 - \left\|r\left(V_k\left(\mathbf{z}^{(k)} + \alpha_j^{(k)}\mathbf{q}^{(k)}\right)\right)\right\|^2 \geq \frac{1}{2}\alpha_j^{(k)} \left\|J^{(k)}\mathbf{q}^{(k)}\right\|^2.$$

Proof This result follows from [25, Proposition 4.1].

Theorem 2 *Let f be Fréchet differentiable and let $\mathbf{z}^{(k)}$, $k = 1, 2, \ldots$, denote the iterates generated by Algorithm 2. Define*

$$\mathbf{x}^{(k)} = V_k \mathbf{z}^{(k)}.$$

Then there is a vector $\mathbf{x}^*$ such that

$$\lim_{k \to \infty} \left\|\mathbf{x}^{(k)} - \mathbf{x}^*\right\| = 0,$$

and $\mathbf{x}^*$ *is a stationary point of* $\mathcal{J}(\mathbf{x}) = \|\mathbf{y} - f(\mathbf{x})\|^2$. *If* $\mathcal{J}$ *is convex, then* $\mathbf{x}^*$ *is a global minimizer of* $\mathcal{J}$. *Moreover, if* $\mathcal{J}$ *is strictly convex, then* $\mathbf{x}^*$ *is the unique global minimizer.*

Proof We first discuss the case when there is an index k_0 such that $\mathbf{g}^{(k)} \in \mathcal{V}_{k_0}$ for all $k > k_0$ and $\dim(\mathcal{V}_{k_0}) < n$. Then, after iteration k_0, all subsequent iterates belong to $\mathcal{V}_{k_0}$. However, since the Armijo-Goldstein principle is satisfied, this means that Algorithm 2 reduces to Algorithm 1 after k_0 iterations and, therefore, the iterates converge to a stationary point of $\mathcal{J}$ by Theorem 1.

On the other hand, if there is no index k_0 such that $\mathbf{g}^{(k)} \in \mathcal{V}_{k_0}$ for all $k > k_0$ and $\dim(\mathcal{V}_{k_0}) < n$, then there is an index k_1 such that $\dim(\mathcal{V}_{k_1}) = n$ and, therefore, $\mathcal{V}_k = \mathbb{R}^n$ for all $k > k_1$. Convergence to a stationary point now follows from Theorem 1.

Remark 1 The proof of Theorem 2 would appear to suggest that one may need n iterations with Algorithm 2 to obtain an accurate approximation of the limit point. However, in problems that stem from real applications, typically only a few iterations are required to satisfy the stopping criterion. This is illustrated by numerical experiments in Section 4.

3.1 Secant acceleration

In some applications the evaluation of the Jacobian J_f of f may be computationally expensive. We therefore discuss the updating strategy proposed by Broyden [6] that makes it possible to avoid the computation of J_f from scratch every iteration. A nice discussion of this updating strategy is provided by Dennis and Schnabel [20, Chapter 8]. Broyden [6] proposed the following secant update of the Jacobian matrix

$$J_f\left(V_{k+1}\mathbf{z}^{(k+1)}\right) \approx J_f\left(V_k\mathbf{z}^{(k)}\right) + H^{(k)}, \tag{3.3}$$

where $H^{(k)}$ generally is a rank-two matrix defined by

$$H^{(k)} = \frac{\Delta\mathbf{r}^{(k)} - J_f\left(V_k\mathbf{z}^{(k)}\right)}{\left\|\Delta\mathbf{x}^{(k)}\right\|^2}\left(\Delta\mathbf{x}^{(k)}\right)^T, \tag{3.4}$$

with $\Delta\mathbf{x}^{(k)} = \mathbf{x}^{(k+1)} - \mathbf{x}^{(k)}$ and $\Delta\mathbf{r}^{(k)} = \mathbf{r}^{(k+1)} - \mathbf{r}^{(k)}$. We then can compute an approximation of $J_f(V_{k+1}\mathbf{z}^{(k+1)})V_{k+1}$ as follows

$$\begin{aligned} J_f(V_{k+1}\mathbf{z}^{(k+1)})V_{k+1} &= \left[J_f(V_{k+1}\mathbf{z}^{(k+1)})V_k,\ J_f(V_{k+1}\mathbf{z}^{(k+1)})\mathbf{v}^{(k+1)}\right] \\ &\approx \left[J_f(V_k\mathbf{z}^{(k)})V_k + H^{(k)}V_k,\ J_f(V_k\mathbf{z}^{(k)})\mathbf{v}^{(k+1)} + H^{(k)}\mathbf{v}^{(k+1)}\right]. \end{aligned}$$

These updating formulas can be applied repeatedly, for $k = 0, 1, \ldots$, with a suitable choice of $H^{(0)}$. Then $H^{(k)}$ is a low-rank matrix when k is fairly small; see [20]. The matrix $H^{(k)}$ typically is not explicitly formed; instead the

vectors that make up $H^{(k)}$ are stored and used when evaluating matrix-vector products with $H^{(k)}$. When the number of updating steps increases, the quality of the approximation of J_f by $H^{(k)}$ at desired points $\mathbf{x}$ may deteriorate, but not by very much. Dennis and Schnabel [20, p. 176] refer to this behavior as "bounded deterioration"; see [20, Lemma 8.2.1] for bounds on the deterioration. When the approximation $H^{(0)}$ of the Jacobian at the initial point $\mathbf{x}_{\text{init}}$ is sufficiently close to the Jacobian at the desired solution, $J_f(\mathbf{x}^*)$, the bounded deterioration of the quality of the approximations of $H^{(k)}$ is mild enough to secure convergence to the solution $\mathbf{x}^*$ as k increases when repeatedly using the updating formula; see [20, Theorem 8.2.2] for details. Nevertheless, since it is difficult to assess whether the conditions of this theorem are satisfied, we recompute the Jacobian from scratch every $\tilde{k}$ iterations. Algorithm 3 summarizes the computations.

3.2 Restarting strategy

The main computational cost of the outlined solution method, if we ignore the computation of the Jacobian of f, is the solution of a linear system of equations of fairly small size at every iteration. However, if many iterations are carried out, the systems may become large and require a non-negligible computational effort to solve. To avoid this issue, we employ the restart strategy proposed in [11] for the Maximization-Minimization algorithm. Specifically, the solution method described in [11] restarts the generalized Krylov subspace when a fixed user-specified number of iteration have been carried out. We denote this number by $k_{\text{rest}} > 1$ as the number of iterations after which we restart the GKS. Since we set $V_0 \in \mathbb{R}^{n\times 1}$, we have that, if no restarting is used, $\dim(V_k) = k+1$. To avoid that the dimension of V_k and, therefore, the size of $J^{(k)}$, become too large, we set, if $k \equiv 0 \bmod k_{\text{rest}}$,

$$V_k = \frac{\mathbf{x}^{(k)}}{\left\|\mathbf{x}^{(k)}\right\|}.$$

This ensures that the number of columns of V_k never exceeds k_{rest} and that the computational effort per iteration does not increase too much with k. For large-scale problems, this also has the added benefit of reducing the memory requirement of the method. This may be important if only a small amount of fast memory is available or if the size of the problem to be solved is very large; see [11] for a discussion.

Since when using restarts the spaces $\mathcal{V}_k$ are not nested, the proof of Theorem 2 does not hold. We only can state that, thanks to the Armijo-Goldstein rule, it holds

$$\left\|\mathbf{r}^{(k+1)}\right\| \leq \left\|\mathbf{r}^{(k)}\right\|, \quad \forall k > 0.$$

Algorithm 4 summarizes the computations.

Algorithm 3: Gauss-Newton in GKS with secant update

input : Nonlinear function f, data $\mathbf{y}$, initial guess for the approximate solution $\mathbf{x}^{(0)}$, initial guess for the damping parameter α_0, tolerance τ, $\tilde{k}$ number of iterations for which the Jacobian is updated, maximum number of iterations K.

output: Approximate solution $\widetilde{\mathbf{x}}^*$.

$V_0 = \mathbf{x}^{(0)} / \left\|\mathbf{x}^{(0)}\right\|$;
$\mathbf{z}^{(0)} = \left\|\mathbf{x}^{(0)}\right\|$;
Compute the Jacobian J_f of f in $\mathbf{x}^{(0)}$;
for $k = 0, 1, 2, \ldots, K$ **do**
  if $k - 1 \equiv 0 \bmod \tilde{k}$ & $k \neq 1$ **then**
    $J^{(k)} = [J^{(k-1)} - H^{(k-1)} V_{k-1}, -J^{(k-1)}\mathbf{v}^{(k)} - H^{(k-1)}\mathbf{v}^{(k)}]$;
  else
    $J^{(k)} = -J_f V_k$;
  $\mathbf{r}^{(k)} = \mathbf{y} - f\left(V_k \mathbf{z}^{(k)}\right)$;
  $\mathbf{q}^{(k)} = \arg \min_{\mathbf{q} \in \mathbb{R}^{d_k}} \left\|\mathbf{r}^{(k)} + J^{(k)}\mathbf{q}\right\|^2$;
  $\alpha_0^{(k)} = \alpha_0$;
  $j = 0$;
  while $\left\|r\left(V_k \mathbf{z}^{(k)}\right)\right\|^2 - \left\|r\left(V_k\left(\mathbf{z}^{(k)} + \alpha_j^{(k)}\mathbf{q}^{(k)}\right)\right)\right\|^2 \leq \frac{1}{2}\alpha_j^{(k)} \left\|J^{(k)}\mathbf{q}^{(k)}\right\|^2$ **do**
    $\alpha_{j+1}^{(k)} = \frac{\alpha_j^{(k)}}{2}$;
    $j = j + 1$;
  $\alpha^{(k)} = \alpha_j^{(k)}$;
  $\mathbf{z}^{(k+1)} = \mathbf{z}^{(k)} + \alpha^{(k)}\mathbf{q}^{(k)}$;
  if $\left\|\mathbf{z}^{(k)} - \mathbf{z}^{(k+1)}\right\| \leq \tau \left\|\mathbf{z}^{(k)}\right\|$ **then**
    exit;
  if $k \equiv 0 \bmod \tilde{k}$ & $k \neq 0$ **then**
    $\Delta\mathbf{x}^{(k)} = \mathbf{x}^{(k+1)} - \mathbf{x}^{(k)}$ and $\Delta\mathbf{r}^{(k)} = \mathbf{r}^{(k+1)} - \mathbf{r}^{(k)}$;
    $H^{(k)} = \dfrac{\Delta\mathbf{r}^{(k)} - J_f\left(V_k\mathbf{z}^{(k)}\right)}{\left\|\Delta\mathbf{x}^{(k)}\right\|^2}\left(\Delta\mathbf{x}^{(k)}\right)^T$;
    $J_f = J_f + H^{(k)}$;
  else
    Compute the Jacobian J_f of f at $V_k\mathbf{z}^{(k+1)}$;
  $\mathbf{g}^{(k+1)} = J_f^T \mathbf{r}^{(k)}$;
  $\tilde{\mathbf{g}}^{(k+1)} = \mathbf{g}^{(k+1)} - V_k V_k^T \mathbf{g}^{(k+1)}$;
  $V_{k+1} = \left[V_k,\ \tilde{\mathbf{g}}^{(k+1)} / \left\|\tilde{\mathbf{g}}^{(k+1)}\right\|\right]$;
  $\mathbf{z}^{(k+1)} = \left[\mathbf{z}^{(k+1)}\ 0\right]^T$;
$\widetilde{\mathbf{x}}^* = V_k \mathbf{z}^{(k+1)}$;

4 Numerical examples

In this section, we apply the algorithms described in the previous sections to three nonlinear least-squares problems: a real-world problem from applied geophysics, a partial differential equation that appears in a number of applications such as in fuel ignition models of thermal combustion theory, and

Algorithm 4: A restarted Gauss-Newton method using generalized Krylov subspaces

input : Nonlinear function f, data vector $\mathbf{y}$, initial guess for the approximate solution $\mathbf{x}^{(0)}$, initial guess for the damping parameter α_0, number of iterations after which a restarting of the GKS is performed k_{rest}, tolerance τ, maximum number of iterations K.

output: Approximation $\widetilde{\mathbf{x}}^*$ of the solution $\mathbf{x}^*$.

$V_0 = \mathbf{x}^{(0)} / \left\|\mathbf{x}^{(0)}\right\|$;
$\mathbf{z}^{(0)} = \left\|\mathbf{x}^{(0)}\right\|$;
Compute the Jacobian J_f of f at $\mathbf{x}^{(0)}$;
for $k = 0, 1, 2, \ldots, K$ **do**
  if $k \equiv 0 \, mod \, k_{\text{rest}}$ **then**
    $\mathbf{x} = V_k \mathbf{z}^{(k)}$;
    $V_k = \mathbf{x} / \|\mathbf{x}\|$;
  $J^{(k)} = -J_f V_k$;
  $\mathbf{r}^{(k)} = \mathbf{y} - f\left(V_k \mathbf{z}^{(k)}\right)$;
  $\mathbf{q}^{(k)} = \arg \min_{\mathbf{q} \in \mathbb{R}^{d_k}} \left\|\mathbf{r}^{(k)} + J^{(k)} \mathbf{q}\right\|^2$;
  $\alpha_0^{(k)} = \alpha_0$;
  $j = 0$;
  while $\left\|r\left(V_k \mathbf{z}^{(k)}\right)\right\|^2 - \left\|r\left(V_k\left(\mathbf{z}^{(k)} + \alpha_j^{(k)} \mathbf{q}^{(k)}\right)\right)\right\|^2 \leq \frac{1}{2} \alpha_j^{(k)} \left\|J^{(k)} \mathbf{q}^{(k)}\right\|^2$ **do**
    $\alpha_{j+1}^{(k)} = \frac{\alpha_j^{(k)}}{2}$;
    $j = j + 1$;
  $\alpha^{(k)} = \alpha_j^{(k)}$;
  $\mathbf{z}^{(k+1)} = \mathbf{z}^{(k)} + \alpha^{(k)} \mathbf{q}^{(k)}$;
  if $\left\|\mathbf{z}^{(k)} - \mathbf{z}^{(k+1)}\right\| \leq \tau \left\|\mathbf{z}^{(k)}\right\|$ **then**
    exit;
  Compute the Jacobian J_f of f at $V_k \mathbf{z}^{(k+1)}$;
  $\mathbf{g}^{(k+1)} = J_f^T \mathbf{r}^{(k)}$;
  $\tilde{\mathbf{g}}^{(k+1)} = \mathbf{g}^{(k+1)} - V_k V_k^T \mathbf{g}^{(k+1)}$;
  $V_{k+1} = \left[V_k, \ \tilde{\mathbf{g}}^{(k+1)} / \left\|\tilde{\mathbf{g}}^{(k+1)}\right\|\right]$;
  $\mathbf{z}^{(k+1)} = \left[\mathbf{z}^{(k+1)} \ 0\right]^T$;
$\widetilde{\mathbf{x}}^* = V_k \mathbf{z}^{(k+1)}$;

a simple problem for which the Jacobian is a very sparse matrix. For each problem we compare the algorithms in terms of accuracy of the computed solution, number of iterations required to satisfy the stopping criterion, and CPU time. We measure the accuracy by means of the Relative Reconstruction Error (RRE)

$$\text{RRE}(\mathbf{x}) = \frac{\|\mathbf{x} - \mathbf{x}_{\text{true}}\|}{\|\mathbf{x}_{\text{true}}\|},$$

where $\mathbf{x}_{\text{true}}$ denotes the desired solution of the problem. For each algorithm we set the maximum number of iterations to $K = 100$ and the tolerance for the stopping criterion to $\tau = 10^{-5}$. In Algorithm 3 we set $\tilde{k} = 10$, i.e., we compute the Jacobian from scratch every $\tilde{k} = 10$ iterations. The updating formulas

perform best if the updated Jacobian is close to the Jacobian at $\mathbf{x}_{\text{true}}$. We therefore do not use the updating formulas in Algorithm 3 for $k = 1, 2, \ldots, \tilde{k}$. Finally, we set in Algorithm 4 the number of iterations after which the space $\mathcal{V}_k$ is restarted to $k_{\text{rest}} = 20$. All computations were carried out in MATLAB 2021b running on a laptop computer with an AMD Ryzen 7 5800HS CPU and 16GB of RAM.

A geophysics problem. We consider a nonlinear model from geophysics with the aim of reconstructing the distribution of the electrical conductivity and the magnetic permeability of the subsoil from measured data recorded with a ground conductivity meter (GCM). This device is made up of two coils, a transmitter and a receiver. The transmitter sends electromagnetic waves into the subsoil, while the receiver measures the induced electromagnetic field; see, e.g., [8, 14, 17–19] and references therein for more details on the mathematical model and [15, 16] for numerical results obtained.

We briefly recall how the model is defined. It is based on Maxwell's equations. The subsoil is assumed to have a layered structure with n layers of thickness d_ℓ, $\ell = 1, 2, \ldots, n$. Layer ℓ has electric conductivity σ_ℓ and magnetic permeability μ_ℓ. The last layer is assumed to be infinite. We define the so-called propagation constant as

$$u_\ell(\lambda) = \sqrt{\lambda^2 + \mathrm{i}\sigma_\ell \mu_\ell \omega}, \qquad \mathrm{i} = \sqrt{-1},$$

where λ is the integration variable ranging from zero to infinity. It represents the ratio of the depth below the surface measured in meters and the inter-coil distance ρ (the distance between the transmitter and the receiver). The parameter ω stands for the angular frequency, that is, 2π times the frequency in Hertz.

The model, which describes the interaction between the GCM and the subsoil for the vertical and horizontal orientation of the device coils, is given by

$$\begin{cases} M^V(\boldsymbol{\sigma}, \boldsymbol{\mu}; h, \omega, \rho) = -\rho^3 \displaystyle\int_0^\infty e^{-2\lambda h} \lambda^2 R_{\omega,0}(\lambda) J_0(\rho\lambda)\, d\lambda, \\ M^H(\boldsymbol{\sigma}, \boldsymbol{\mu}; h, \omega, \rho) = -\rho^2 \displaystyle\int_0^\infty e^{-2\lambda h} \lambda R_{\omega,0}(\lambda) J_1(\rho\lambda)\, d\lambda, \end{cases}$$

where $\boldsymbol{\sigma} = [\sigma_1, \ldots, \sigma_n]^T$, $\boldsymbol{\mu} = [\mu_1, \ldots, \mu_n]^T$, h represents the height of the device while measuring above the ground, J_s denotes the Bessel function of the first kind of order s, and $R_{\omega,0}(\lambda)$ is the reflection factor defined as

$$R_{\omega,0}(\lambda) = \frac{N_0(\lambda) - Y_1(\lambda)}{N_0(\lambda) + Y_1(\lambda)}.$$

Here $N_0(\lambda) = \lambda/(\mathrm{i}\mu_0\omega)$, $\mu_0 = 4\pi \cdot 10^{-7}\mathrm{H/m}$ is the magnetic permeability of free space and $Y_1(\lambda)$ is computed by the recursive formula

$$Y_\ell(\lambda) = N_\ell(\lambda) \frac{Y_{\ell+1}(\lambda) + N_\ell(\lambda)\tanh(d_\ell u_\ell(\lambda))}{N_\ell(\lambda) + Y_{\ell+1}(\lambda)\tanh(d_\ell u_\ell(\lambda))}, \tag{4.1}$$

for $\ell = n-1, n-2, \ldots, 1$, where $Y_\ell(\lambda)$ is the surface admittance at the top of each layer. The coefficient $N_\ell(\lambda) = u_\ell(\lambda)/(\mathrm{i}\mu_\ell\omega)$ represents the characteristic admittance at the ℓ-th layer and yields the initialization of the recursion formula (4.1), that is when $\ell = n$, we set $Y_n(\lambda) = N_n(\lambda)$.

For simplicity, we focus on the reconstruction of the distribution of the electrical conductivity and assume the magnetic permeability to be known. To this end, we construct three test cases and consider three different profiles for the electrical conductivity: a Gaussian function, a continuous, but not everywhere differentiable function (which we refer to as "triangular"), and a step function. It is known that in this application the model requires the solution to be differentiable. Therefore, the last two cases are very challenging to solve. When we construct our test cases, we assume that there is no noise except for round-off errors. Thus, we commit an inverse-crime. However, we remark that the Jacobian of f is so ill-conditioned that perturbations introduced by round-off errors produce significant fluctuations in the computed solutions.

We set $\mathbf{x}^{(0)}$ to be a constant function. Since the problem is non-convex and underdetermined, the choice of the function $\mathbf{x}^{(0)}$ is important. For the Gaussian and triangular profiles, we set $\mathbf{x}_0 = [0.5,\, 0.5, \ldots, 0.5]^T$, while for the step function we set $\mathbf{x}_0 = [1.5,\, 1.5, \ldots, 1.5]^T$. Moreover, we calibrate the instrument so that measurements are taken for 10 different heights, i.e., $m = 10$, and the subsoil is assumed to be composed of 100 layers, i.e., $n = 100$.

Table 1 reports results obtained with Algorithms 1, 2, 3, and 4. Since the problems considered are extremely ill-conditioned, the Gauss-Newton algorithm produces poor reconstructions. Therefore, similarly to the Levenberg-Marquardt scheme (see, e.g., [26] and references therein) we regularize the inversion of the Jacobian by applying Tikhonov regularization; see, e.g., [21] for a discussion and analysis of Tikhonov regularization. In particular, we solve

$$\mathbf{q}^{(k+1)} = \arg\min_{\mathbf{q}\in\mathbb{R}^{d_k}} \left\{ \left\| \mathbf{r}^{(k)} + J^{(k)}\mathbf{q} \right\|^2 + 10^{-4} \left\|\mathbf{q}\right\|^2 \right\},$$

and refer to the algorithm so defined as "Algorithm 1 Regularized". Here 10^{-4} is the regularization parameter and $\|\mathbf{q}\|^2$ is the regularization operator.

We can observe that, even when regularized, the Gauss-Newton algorithm fails to provide an accurate approximation of the desired solution and the algorithm is significantly slower than Algorithms 2, 3, and 4 both in terms of the number of iterations required and CPU time. The latter three algorithms provide satisfactory reconstructions for all three problems and are inexpensive computationally. Since very few iterations are performed Algorithms 2 and 4 produce basically indistinguishable results. Moreover, the use of the Broyden update reduces the computational cost with small to no impact on the accuracy of the computed solution. We point out that Algorithms 2, 3, and 4 do not require fine-tuning of any parameters to perform well, and they do not require Tikhonov regularization. The number of restart iterations k_{rest} can be fixed to 20 in all cases and, in our experience, the method is quite robust with respect to the choice of this parameter.

Table 1 Geophysics problem: RRE, number of iterations performed, and CPU times obtained with the considered algorithms for the three different profiles of the electrical conductivity.

Profile	Method	RRE	Iter.	CPU time (sec.)
Gaussian	Algorithm 1	0.92282	100	64.04
	Algorithm 1 Reg.	0.68111	100	30.33
	Algorithm 2	0.020354	22	7.13
	Algorithm 3	0.074752	15	3.76
	Algorithm 4	0.020360	20	6.54
Triangular	Algorithm 1	0.53842	100	60.33
	Algorithm 1 Reg.	0.32460	100	30.85
	Algorithm 2	0.058914	17	5.45
	Algorithm 3	0.060965	14	3.83
	Algorithm 4	0.058914	17	5.43
Step	Algorithm 1	$> 5 \cdot 10^4$	100	52.41
	Algorithm 1 Reg.	0.83263	100	30.43
	Algorithm 2	0.25470	17	5.38
	Algorithm 3	0.25226	19	3.58
	Algorithm 4	0.25470	17	5.42

Figure 1 displays the computed reconstructions determined by the regularized Gauss-Newton method and Algorithms 2 and 3. We do not show the graphs determined by Algorithm 1 as these computed solutions are not meaningful approximation of the desired solutions. Moreover, we do not display the approximations computed by Algorithm 4 as they are nearly identical to the ones obtained by Algorithms 2.

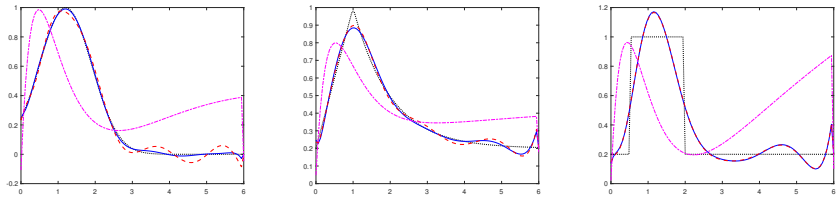
Note that we do not propose here to combine both the secant update with the restarting of the solution subspace. The reason is twofold, firstly we do not want to complicate any further the algorithm by inserting multiple parameters that need to be tuned. Secondly, in our computed example few iterations are performed and the possible computational advantage of this combination would be negligible. Moreover, the obtained algorithm would be highly heuristic and it may be possible to expect erratic behavior. Therefore, at this time we do not consider the combinations of these two acceleration techniques.

Partial differential equation. This example describes a nonlinear partial differential equation that can be solved with the Gauss-Newton method. Specifically, we consider the Bratu problem

$$\begin{cases} -\Delta x(s,t) + \alpha x_s(s,t) + \lambda \mathrm{e}^{x(s,t)} = y, & (s,t) \in \Omega, \\ x(s,t) = 0, & (s,t) \in \partial\Omega, \end{cases} \tag{4.2}$$

where $\Delta x(s,t)$ stands for the Laplacian of x, x_s denotes the partial derivative of x along the s-direction, and α is a constant. We are interested in investigating how the solution changes when the parameter λ is varied; see, e.g., [12, 24, 29] for discussions and illustrations of these kinds of path-following problems.

We let $\Omega = [-3, 3]^2$ and discretize the problem with standard 2nd order finite differences on an $n \times n$ equispaced grid. The discretized problem can be



(a) (b) (c)

Figure 1 Geophysics problem: Reconstruction of the Gaussian (a), triangular (b), and step function (c) test profiles. The black dotted curves show the exact solutions, the magenta dashed-dotted curves are approximations determined by the regularized Gauss-Newton method, the blue solid curves show approximate solutions computed by Algorithm 2, and the red dashed curves display approximate solutions obtained with Algorithm 3. We do not report the approximations computed by Algorithm 1 as they are not meaningful and the ones obtained by Algorithm 4 since they are indistinguishable from the one returned by Algorithm 2.

written as

$$\arg\min_{\mathbf{x}\in\mathbb{R}^{n^2}} \|\mathbf{y} - (L\mathbf{x} + \alpha D\mathbf{x} + \lambda \mathrm{e}^{\mathbf{x}})\|^2,$$

where the entries of $\mathbf{x}$ represent the discretized solution arranged in lexicographical order, $L = L_1 \otimes I + I \otimes L_1$, $D = D_1 \otimes I$, where $\otimes$ denotes the Kronecker product, and the exponential is meant element-wise. We let

$$L_1 = \begin{bmatrix} 2 & -1 & & & \\ -1 & 2 & -1 & & \\ & \ddots & \ddots & \ddots & \\ & & -1 & 2 & -1 \\ & & & -1 & 2 \end{bmatrix} \in \mathbb{R}^{n\times n} \quad \text{and} \quad D_1 = \begin{bmatrix} -1 & 1 & & \\ & \ddots & \ddots & \\ & & -1 & 1 \\ & & & -1 \end{bmatrix} \in \mathbb{R}^{n\times n}.$$

For simplicity, once we compute the sampling of the function x and denote it by $\mathbf{x}_{\text{true}}$, we construct $\mathbf{y}$ by

$$\mathbf{y} = L\mathbf{x}_{\text{true}} + \alpha D\mathbf{x}_{\text{true}} + \lambda \mathrm{e}^{\mathbf{x}_{\text{true}}},$$

i.e., we disregard approximation errors in the PDE as well as the scaling.

In our test, we let $x(s,t) = \mathrm{e}^{-10(s^2+t^2)}$ and sample x on a 100×100 equispaced grid on the square $[-3,3]^2$. Therefore, in (1.1), $f : \mathbb{R}^{10^4} \to \mathbb{R}^{10^4}$. The Jacobian of f at the point $\mathbf{x}$ is given by

$$J_f = L + \alpha \cdot D + \lambda \cdot \mathrm{diag}(\mathrm{e}^{\mathbf{x}}).$$

Since we impose zero Dirichlet boundary conditions, L and D are Block Toeplitz with Toeplitz Block (BTTB) matrices. Therefore, the matrix J_f is a Generalized Locally Toeplitz (GLT) matrix; see [1, 2, 22, 23]. The GLT theory furnishes a tool for studying the behavior of the singular values of J_f. Providing a complete and precise analysis of the behavior of the singular values of J_f is outside the scope of this paper. Here we just note that the singular

values of J_f can be approximated quite accurately, if n^2 is large enough, by a uniform sampling over $[0,\pi]^2$ of the modulus of the GLT symbol of J_f. Let $x : [0,1]^2 \to \mathbb{R}$ be the function that $\mathbf{x}$ is a sampling of. Then, the GLT symbol of J_f is given by

$$S(\theta,\phi,s,t) = (2-2\cos(\theta))+(2-2\cos(\phi))+\alpha(-1+\cos(\theta)-\mathrm{i}\sin(\theta)))+\lambda \mathrm{e}^{x(s,t)},$$

where $\mathrm{i} = \sqrt{-1}$, $(\theta,\phi) \in [-\pi,\pi]^2$, and $(s,t) \in [0,1]^2$. It is straightforward to see that, if α is large and λ is small, then the singular values of J_f decay rapidly to 0 with increasing index and, therefore, J_f is ill-conditioned. On the other hand, if λ is large and α is small, then the singular values decay slowly with increasing index and, therefore, J_f is well-conditioned. This is confirmed by Figure 2, which displays the approximated singular values of J_f for two choices of α and λ, namely $(\alpha,\lambda) = (1,10)$ and $(\alpha,\lambda) = (10,1)$, with $\mathbf{x} = \mathbf{x}_{\text{true}}$. We observe that in the first case the matrix J_f is very well-conditioned, while in the latter case the matrix is poorly conditioned. We can estimate the condition number of J_f in these two cases with the MATLAB function `condest`. In the first case, i.e., the well-conditioned one, the computed condition number is $\kappa_2 \approx 3.3$, while in the latter case, i.e., the ill-conditioned one, we obtain $\kappa_2 \approx 1.7 \cdot 10^{30}$. This illustrates that the conditioning for large values of α is far worse than what the GLT theory predicts.

We illustrate the performances of Algorithms 1, 2, and 4 for several choices of α and λ. In particular, we let $(\alpha,\lambda) \in \{1,\ 2,\ldots,\ 10\}^2$. Since we observed that the standard Gauss-Newton method for some choices of α and λ stops after a single iteration, we forced this method to carry out at least 5 iterations.

Table 2 reports the means, standard deviations, maximum values, and minimum values for the RRE, number of iterations, and CPU times required for Algorithms 1, 2, and 4. Since the computation of J_f is very cheap, we do not consider Algorithm 3 in this example. We note that, on average, Algorithms 1 and 4 yield less accurate computed solutions than Algorithm 2. However, for some parameter pairs (α,λ) the accuracy obtained with Algorithms 1 and 4 is much higher than with Algorithm 2. Nevertheless, the approximate solutions computed by the latter algorithms are always fairly accurate, even in the worst case. On the other hand, the Gauss-Newton method determines very poor reconstructions for several choices of α and λ. Figures 3(a), 3(d), and 3(g) show $\log_{10}(\text{RRE})$ for each choice of the parameters. We can observe that, for a fairly large number of choices of α and λ, the Gauss-Newton method performs quite poorly in terms of accuracy, while Algorithms 2 and 4 always provides accurate reconstructions. This is due to the fact that the projection into the generalized Krylov subspace regularizes the problem without reducing the accuracy of the computed solutions.

The CPU times reported in Table 2 and in Figures 3(c), 3(f), and 3(i) illustrate that Algorithms 2 and 4 are faster in terms of CPU time than Algorithm 1, despite that the first two methods usually require more iterations to converge than Algorithm 1. We can also observe that the speed-up obtained by using the restarting strategy is significant as it reduces the average computational cost by a factor of almost 10.

Table 2 Partial differential equation test case: means, standard deviations, maximum values, and minimum values for the RRE, number of iterations, and CPU times obtained for Algorithms 1, 2, and 4. The results are obtained for $(\alpha, \lambda) \in \{1,\ 2, \dots,\ 10\}^2$ in (4.2).

Method	Parameter	Mean	Std. Dev.	Min.	Max.
Algorithm 1	RRE	0.5580	0.5146	$7.92 \cdot 10^{-16}$	1.655
	Iter.	8.44	17.13	5	100
	CPU time (sec.)	0.6989	0.6246	0.1125	3.09
Algorithm 2	RRE	0.0097	0.0121	$8.21 \cdot 10^{-6}$	0.0654
	Iter.	48.5	26.08	16	100
	CPU time (sec.)	0.2734	0.2948	0.0269	1.0251
Algorithm 4	RRE	0.0142	0.0197	$9 \cdot 10^{-5}$	0.1261
	Iter.	20.34	2.85	16	40
	CPU time (sec.)	0.0356	0.0057	0.0268	0.0763

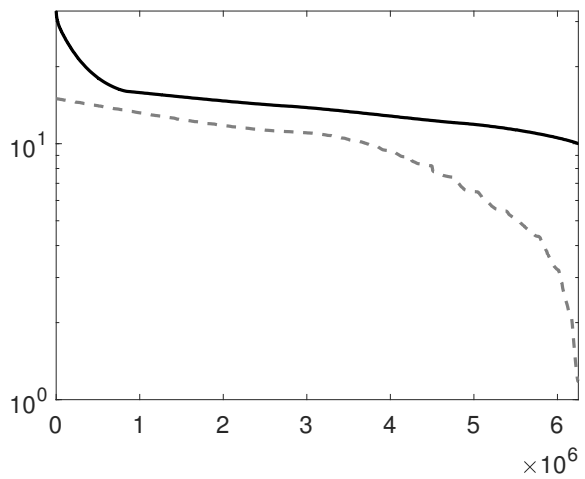


Figure 2 Partial differential equation test case: approximation of the singular values of J_f via the GLT theory with two different couple of parameters (α, λ). The solid black line is obtained for $\alpha = 1$ and $\lambda = 10$, while the dashed gray line is for $\alpha = 10$ and $\lambda = 1$.

We now discuss how the CPU times changes when the dimensions of the problem increase. We fix $\alpha = 5$ and $\lambda = 10$ and let $n \in 100, 150, \dots, 1000$. Therefore, when $n = 1000$, we have $f : \mathbb{R}^{10^6} \to \mathbb{R}^{10^6}$. We run the considered algorithms and plot the CPU times in Figure 4. We can observe that, obviously, the CPU time required to solve the problem increases with n. However, the rate of increase is substantially lower when the projection in the generalized Krylov subspace is employed. In particular, using Algorithm 4, we can see that we are able to solve a nonlinear problem with 10^6 unknowns in less than 10 seconds on a laptop.

An extremely sparse problem. In this example, we would like to show a very particular situation when Algorithms 2 and 3 are not faster than Algorithm 1.

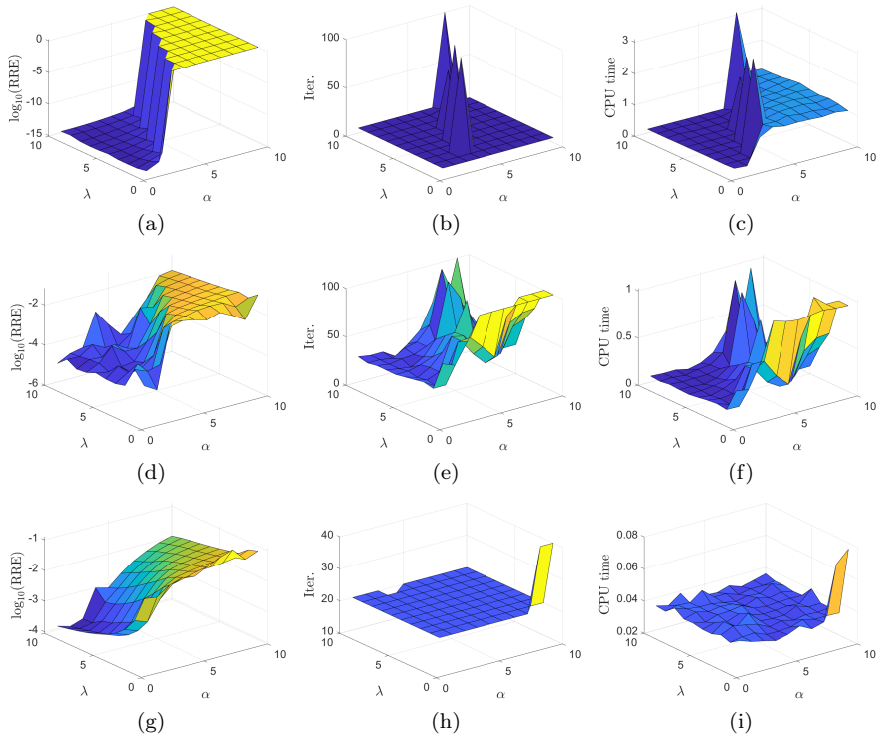


Figure 3 Partial differential equation test case: RRE, number of iterations, and CPU times obtained with Algorithms 1, 2, and 4 for each choice of $(\alpha, \lambda) \in \{1,\ 2, \ldots,\ 10\}^2$ in (4.2). The first row reports the results obtained with Algorithm 1, the second row shows results for Algorithm 2, and the third row contains results for Algorithm 4.

However, we hasten to point out that this situation arises very seldom. Introduce the nonlinear differentiable function $f : \mathbb{R}^n \to \mathbb{R}^{n-1}$ given by

$$f(\mathbf{x}) = \sin\left(\begin{bmatrix} x_1 + x_2 \\ x_2 + x_3 \\ \vdots \\ x_{n-1} + x_n \end{bmatrix}\right),$$

where the sin operation is meant element-wise. We construct the problem by choosing as the exact solution $\mathbf{x} \in \mathbb{R}^n$ of the problem (1.1) a sampling of the function $\mathbf{x} = \frac{1}{2}\sin(x)$, with $x \in (-\pi, \pi)$, on an equispaced grid with $n = 10^3$. The Jacobian $J_f(\mathbf{x})$ of f is bidiagonal and given by

$$J_f(\mathbf{x}) = \begin{bmatrix} \cos(x_1 + x_2) & \cos(x_1 + x_2) & & & \\ & \cos(x_2 + x_3) & \cos(x_2 + x_3) & & \\ & & \ddots & \ddots & \\ & & & \cos(x_{n-1} + x_n) & \cos(x_{n-1} + x_n) \end{bmatrix} \in \mathbb{R}^{(n-1)\times n}.$$

Least-square problems with such a matrix can be solved very inexpensively.

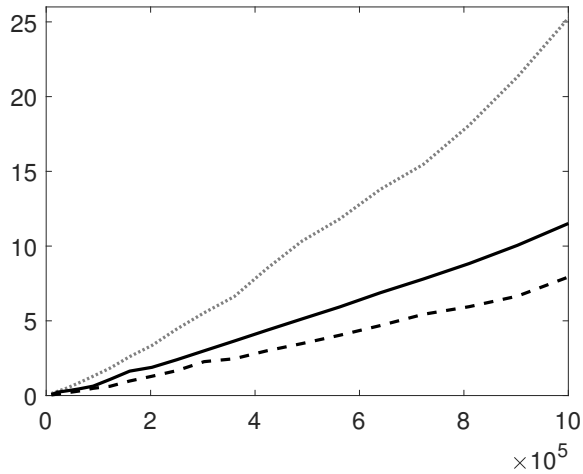


Figure 4 Partial differential equation test case: CPU times obtained with Algorithms 1, 2, and 4 with $(\alpha, \lambda) = (5, 10)$ in (4.2) for increasing dimensions of the problem. The dotted gray line reports the results obtained with Algorithm 1, the solid black line shows results for Algorithm 2, and the dashed black contains results for Algorithm 4.

Table 3 An extremely sparse problem: RRE, number of iterations performed, and CPU times for Algorithms 1 and 2.

Method	RRE	Iter.	CPU time (sec.)
Algorithm 1	0.085038	5	0.00608
Algorithm 2	0.00012168	17	0.0185

Table 3 reports results obtained for Algorithms 1 and 2. We note that the latter algorithm determines a more accurate approximate solution than the former. However, since the solution of least-squares problems with the sparse matrix J_f is very inexpensive to compute and Algorithm 1 carries out fewer iterations than Algorithm 2, the overall cost of the former algorithm is slightly smaller than of the latter algorithm. Nevertheless, Algorithm 2 reduces the cost per iteration significantly, when compared with Algorithm 1, and provides more accurate reconstructions. Therefore, one may still want to use Algorithm 2 even if it is slightly more expensive.

Note that, since Algorithm 2 carried out fewer than k_{rest} iterations, we do not report the results for Algorithm 4 as they are identical to the ones obtained with Algorithm 2.

5 Conclusion and extensions

This paper presents new implementations of the Gauss-Newton algorithm based on the use of generalized Krylov subspaces. The approach described

easily can be extended to other nonlinear optimization methods such as the Levenberg-Marquardt method. We have shown that Algorithm 2 determines approximate solutions that converge to a stationary point of the minimized functional. Several numerical examples show that Algorithms 2, 3, and 4 outperform the standard Gauss-Newton method. Extensions to regularized problems as well as other iterative algorithms are presently being developed.

Declarations

Author contributions All authors contributed equally.
Funding A. B., P. D. de A., and F. P. are members of the GNCS group of INdAM and are partially supported by the INdAM-GNCS 2022 Project "Metodi e modelli di regolarizzazione per problemi mal posti di grandi dimensioni" (CUP_E55F22000270001). A. B. and F. P. are partially supported by Fondazione di Sardegna, Progetto biennale bando 2021, "Computational Methods and Networks in Civil Engineering (COMANCHE)" and by INdAM-GNCS 2023 Project "Tecniche numeriche per lo studio dei problemi inversi e l'analisi delle reti complesse" (CUP_E53C22001930001). P. D. de A. is partially supported by the INdAM-GNCS 2023 Project "Metodi numerici per modelli descritti mediante operatori differenziali e integrali non locali" (CUP_E53C22001930001) and gratefully acknowledges Fondo Sociale Europeo REACT EU - Programma Operativo Nazionale Ricerca e Innovazione 2014-2020 and Ministero dell'Università e della Ricerca for the financial support.
Data availability Data sharing is not applicable to this article as no datasets were generated or analyzed
Competing interests The authors declare no competing interests. during the current study.

Acknowledgments

The authors would like to thank Fabio Durastante for insightful discussions and the anonymous referees for their comments that helped us to greatly improve this work.

References

1. Barbarino, G., Garoni, C., Serra-Capizzano, S.: Block generalized locally Toeplitz sequences: Theory and applications in the multidimensional case. Electronic Transactions on Numerical Analysis **53**, 113–216 (2020)
2. Barbarino, G., Garoni, C., Serra-Capizzano, S.: Block generalized locally Toeplitz sequences: Theory and applications in the unidimensional case. Electronic Transactions on Numerical Analysis **53**, 28–112 (2020)
3. Björck, Å.: Numerical Methods for Least Squares Problems. SIAM. Philadelphia (1996)
4. Brown, P.N., Saad, Y.: Hybrid Krylov methods for nonlinear systems of equations. SIAM Journal on Scientific and Statistical Computing **11**(3), 450–481 (1990)

5. Brown, P.N., Saad, Y.: Convergence theory of nonlinear Newton–Krylov algorithms. SIAM Journal on Optimization **4**(2), 297–330 (1994)
6. Broyden, C.G.: A class of methods for solving nonlinear simultaneous equations. Mathematics of Computation **19**(92), 577–593 (1965)
7. Buccini, A.: Fast Alternating Direction Multipliers Method by Generalized Krylov Subspaces. Journal of Scientific Computing **90**(1), 60 (2022)
8. Buccini, A., Díaz de Alba, P.: A variational non-linear constrained model for the inversion of FDEM data. Inverse Problems **38**, 014,001 (2022)
9. Buccini, A., De la Cruz Cabrera, O., Koukouvinos, C., Mitrouli, M., Reichel, L.: Variable selection in saturated and supersaturated designs via minimization. Communications in Statistics - Simulation and Computation, in press (2021)
10. Buccini, A., Pragliola, M., Reichel, L., Sgallari, F.: A comparison of parameter choice rules for ℓ^p-ℓ^q minimization. Annnali dell'Università di Ferrara **68**(2), 441–463 (2022)
11. Buccini, A., Reichel, L.: Limited memory restarted $\ell^p - \ell^q$ minimization methods using generalized Krylov subspaces. Advances in Computational Mathematics **49**(2), 26 (2023)
12. Calvetti, D., Reichel, L.: Iterative methods for large continuation problems. Journal of Computational and Applied Mathematics **123**, 217–240 (2000)
13. Darvishi, M.T., Shin, B.C.: High-order Newton-Krylov methods to solve systems of nonlinear equations. J. KSIAM **15**, 19–30 (2011)
14. Deidda, G.P., Díaz de Alba, P., Rodriguez, G.: Identifying the magnetic permeability in multi-frequency EM data inversion. Electronic Transactions on Numerical Analysis **47**, 1–17 (2017)
15. Deidda, G.P., Díaz de Alba, P., Fenu, C., Lovicu, G., Rodriguez, G.: FDEMtools: A MATLAB package for FDEM data inversion. Numerical Algorithms **84(4)**, 1313–1327 (2020)
16. Deidda, G.P., Díaz de Alba, P., Pes, F., Rodriguez, G.: Forward Electromagnetic Induction Modelling in a Multilayered Half-Space: An Open-Source Software Tool. Remote Sensing **15**(7), 1772 (2023)
17. Deidda, G.P., Díaz de Alba, P., Rodriguez, G., Vignoli, G.: Smooth and sparse inversion of EMI data from multi-configuration measurements. In: 2018 IEEE 4th International Forum on Research and Technology for Society and Industry (RTSI) (RTSI 2018), pp. 213–218. Palermo, Italy (2018)
18. Deidda, G.P., Díaz de Alba, P., Rodriguez, G., Vignoli, G.: Inversion of multiconfiguration complex emi data with minimum gradient support regularization: A case study. Mathematical Geosciences **52(7)**, 945–970 (2020)
19. Deidda, G.P., Fenu, C., Rodriguez, G.: Regularized solution of a nonlinear problem in electromagnetic sounding. Inverse Problems **30**, 125,014 (27 pages) (2014)
20. Dennis Jr., J.E., Schnabel, R.B.: Numerical Methods for Unconstrained Optimization and Nonlinear Equations. SIAM, Philadelphia (1996)
21. Engl, H.W., Hanke, M., Neubauer, A.: Regularization of Inverse Problems. Kluwer, Dordrecht (1996)
22. Garoni, C., Serra-Capizzano, S.: Generalized Locally Toeplitz Sequences: Theory and Applications, vol. 1. Springer, Berlin (2017)
23. Garoni, C., Serra-Capizzano, S.: Generalized Locally Toeplitz Sequences: Theory and Applications, vol. 2. Springer, Berlin (2018)
24. Glowinski, R., Keller, H.B., Reinhart, L.: Continuation-conjugate gradient methods for the least squares solution of nonlinear boundary value problems. SIAM Journal on Scientific and Statistical Computing **6**(4), 793–832 (1985)
25. Grippo, L., Sciandrone, M.: Globally convergent block-coordinate techniques for unconstrained optimization. Optimization methods and software **10**(4), 587–637 (1999)
26. Hanke, M.: A regularizing Levenberg-Marquardt scheme, with applications to inverse groundwater filtration problems. Inverse problems **13**(1), 79 (1997)
27. Huang, G., Lanza, A., Morigi, S., Reichel, L., Sgallari, F.: Majorization-minimization generalized Krylov subspace methods for ℓ_p-ℓ_q optimization applied to image restoration. BIT Numerical Mathematics **57**, 351–378 (2017)
28. Kan, Z., Song, N., Peng, H., Chen, B.: Extension of complex step finite difference method to Jacobian-free Newton–Krylov. Journal of Computational and Applied Mathematics **399**, Art. 113,732 (2022)

29. Keller, H.B.: Lectures on Numerical Methods in Bifurcation Problems. Springer, Berlin (1987)
30. Lampe, J., Reichel, L., Voss, H.: Large-scale Tikhonov regularization via reduction by orthogonal projection. Linear Algebra and its Applications **436**, 2845–2865 (2012)
31. Lanza, A., Morigi, S., Reichel, L., Sgallari, F.: A generalized Krylov subspace method for ℓ_p-ℓ_q minimization. SIAM Journal on Scientific Computing **37**, S30–S50 (2015)
32. Pes, F., Rodriguez, G.: The minimal-norm Gauss-Newton method and some of its regularized variants. Electronic Transactions on Numerical Analysis **53**, 459–480 (2020)
33. Pes, F., Rodriguez, G.: A doubly relaxed minimal-norm Gauss–Newton method for underdetermined nonlinear least-squares problems. Applied Numerical Mathematics **171**, 233–248 (2022)
34. Ruhe, A.: Accelerated Gauss–Newton algorithms for nonlinear least squares problems. BIT Numerical Mathematics **19**(3), 356–367 (1979)