1993 11

Faculteit der Economische Wetenschappen en Econometrie

Serie Research Memoranda

On a Riemannian Version of the Levenberg-Marquardt Algorithm

Ralf Peeters

Research-Memorandum 1993-11
March 1993

vrije Universiteit *amsterdam*

On a Riemannian Version of the Levenberg–Marquardt Algorithm *

Ralf Peeters †
Free University, Amsterdam

Abstract

This paper deals with the problem of minimizing a function that exhibits the special structure of a (possibly nonlinear) sum of squares over a Riemannian differentiable manifold. Well–known minimization algorithms to handle similar problems in the (standard) Euclidean case are the methods of Gauss–Newton and Levenberg–Marquardt, which latter method may be regarded as an extension of Gauss–Newton. Since any differentiable manifold can be covered by an atlas of charts of local coordinates, an obvious approach towards this problem consists of applying forementioned methods to all relevant charts individually.
Recently, it has been shown ([14, 31]) that the Gauss–Newton method proceeds virtually independent of the choice of local coordinates, which property makes it especially suited for application to the problem studied here. The method of Levenberg–Marquardt in its present form (cf. [7, 27]) is not coordinate free, however. This is unfortunate, because when restricting to just one chart of local coordinates the Levenberg–Marquardt method often is superior to Gauss–Newton, as it can handle situations where Gauss–Newton breaks down.
In this paper we resolve this situation by constructing a "Riemannian" version of the Levenberg–Marquardt algorithm, i.e., a version acting on a Riemannian manifold in a coordinate independent way. Our motivation for this research stems from the field of system identification, and therefore we have applied it to some test problems of that type. These are described in a companion paper [29].

*This research was carried out as part of NWO research project 611-304-019.

†Address: Free University, Department of Economics and Econometrics, De Boelelaan 1105, 1081 HV Amsterdam, The Netherlands. E-mail: ralf@sara.nl.

1 Introduction

In the literature one can find a wide variety of algorithms for the (local) minimization of a function defined on Euclidean n–space $\mathbf{R}^n$. Depending on the nature of the function to be minimized, some algorithms are better suited for finding a local mimimum quickly than others, but there does not exist such a thing as a "best" algorithm, useful to all applications. Various issues play a role in the choice of an optimization method. These include the size of the problem (the number of variables), the nature of the function (quadratic, convex, etc.), the ease and cost of obtaining function values, first and second order derivatives, etc. In practical implementations also the algorithmic complexity plays a role, but we shall not go into that here.
In the present paper we are interested in the situation where the function to be minimized is a (possibly nonlinear) *sum of squares*, defined over a *Riemannian manifold*. If the domain is Euclidean, which is the standard case, there exist algorithms exploiting the sum–of–squares structure of the function. The most well–known such algorithm is the *Gauss–Newton method*, of which several variants exist. Its "standard" version (involving no step–size controlling parameter) exhibits superlinear convergence under certain conditions, (cf., e.g., [7]). However, this method may fail to converge if those conditions are violated. Then, introduction of a step–size controlling parameter (with various possible strategies for operating it) provides a way out in most cases, but usually at the cost of much slower convergence. Additional problems arise in situations where the Gauss–Newton approximation to the Hessian of the function is (almost) singular; these cannot be handled satisfactorily. (Cf. [7, Ch. 10].)
A second way out of these problems is provided by the so–called *Levenberg–Marquardt algorithm*, that can be viewed as a method that generates search directions by interpolating between those of the methods of Gauss–Newton and steepest descent. In [27] an especially robust implementation of this algorithm was developed, making it of the *trust–region* type. This implementation is discussed in detail in Sect. 2, while related issues and modifications are considered in App. A. It is the purpose of this paper to give a strongly numerically oriented treatment, involving many details, so that an actual computer implementation can readily be developed.
Our motivation for studying the Levenberg–Marquardt algorithm stems from the field of *system identification*. There we are dealing with the problem of estimating parameters in order to identify a model from available measurement data. Most available estimation procedures are based on the concept of trying to minimize an associated criterion function that expresses the misfit of a candidate model. Commonly used criterion functions often fall into the class of *prediction error criteria* (cf., e.g., [24], [32]). The most popular one consists of the sum of squares of the prediction errors that emerge when applying the available measurement data to the candidate model with the objective of one–step–ahead prediction of future outputs. One can get to this criterion in various ways, e.g. via the principle of maximum likelihood.
In many of these situations, the parameter space actually corresponds to only a *part* of the set of candidate models that one has in mind. Indeed, it is a known fact that various model sets that are commonly being used, e.g. the sets of linear multivariable models of a fixed finite order n, do not form Euclidean spaces, but *differentiable manifolds* instead. In the multivariable case it has been proved that one cannot restrict to just *one* continuous canonical form as it can never capture all candidate models (cf. [15]). One therefore has to consider several structures instead, and their related (pseudo–)canonical forms, which provide a set of overlapping parameter charts for the differentiable manifold under consideration. Smooth differentiable manifolds can always be endowed with a Riemannian metric (cf., e.g., [5]), and for this particular application several choices are discussed in [13, 30]. Thus, this application has led us to the study of minimization algorithms for functions that are defined on a Riemannian manifold, covered by a set of overlapping coordinate charts.
This area has turned out to be quite undeveloped. What seems to be common practice is to apply existing algorithms (that were developed for the *Euclidean* case) within a number of local coordinate charts (each describing only a *part* of the manifold). Such an approach is conceptually unsatisfactory in the sense that it generally introduces dependence of the resulting iteration path on the coordinate charts being used — even when applying the *same* minimization method to every chart (cf. [28]). Additionally, this approach may also cause problems that are not inherent to the structure of the function being minimized, but rather due to an inadequate choice of local coordinates.

Some work has been done, however. In, for instance, [11, 22, 23, 25], one can find what we will call "Riemannian" minimization algorithms, i.e. algorithms acting on a Riemannian manifold in a coordinate independent way. In particular there exist Riemannian versions of the steepest descent method, (quasi–)Newton methods and conjugate gradient methods. Initially, a basic obstacle for developing such methods lies in the fact that for a function defined on a differentiable manifold its Hessian is only defined in a coordinate independent way at critical points. In the case of Riemannian manifolds, however, it turns out that one can use geodesics (being the counterpart of straight lines in Euclidean space) as a basis for redefining the Hessian in a coordinate free way everywhere. (As a matter of fact, one then resorts to the use of so–called *normal coordinates.*) We remark that recently, the use of Riemannian geometry is being investigated within more areas in the field of optimization, such as linear programming, cf., e.g., [18, 17].
Other related work of interest, more specific to the application we have in mind, is [12], where a Riemannian version of a *recursive* Gauss–Newton system identification procedure is studied. Also, with respect to nonlinear least squares problems it has recently been shown that one can interpret the Gauss–Newton algorithm as a Riemannian steepest descent minimization method with a certain self–induced (local) Riemannian metric, see [14, 31]. These results provide another incentive for the research of the present paper.
The main subject of this paper is the construction of a Riemannian version of the Levenberg–Marquardt method, which will be treated in Sect. 3. It can be regarded as an extension of the Gauss–Newton method (just as in the Euclidean case) and it gives a conceptually elegant basis for the use of "scaling strategies" that can yield better results than the ones introduced in [27] (cf. App. C). Additionally, it also supplies us with a geometrical interpretation for the shape of the trust–regions.
In a companion paper, [29], this method is applied to a few simulated system identification problems, such as briefly sketched above. Efficient implementation of the Riemannian algorithms in these cases, involving large amounts of data, asks for an alternative organization of certain calculations; these are discussed in App. B.

2 The standard Levenberg–Marquardt algorithm

2.1 Problem statement

We are concerned with the following problem. Suppose we are given a criterion function $\Phi(x)$, which is a sum of squares of possibly nonlinear functions $f^i(x)$, $i = 1, \dots, m$. We are interested in finding a (locally) minimizing argument for $\Phi(x)$, say x_*. Let us introduce the following notation and terminology. Let

$$\Phi(x) = \frac{1}{2}\|f(x)\|^2 = \frac{1}{2}\sum_{i=1}^{m}[f^i(x)]^2 \tag{2.1}$$

where $f : \mathbf{R}^n \to \mathbf{R}^m$ denotes the *residual mapping*, with corresponding coordinate functions (the *residuals*) $f^i : \mathbf{R}^n \to \mathbf{R}$, ($i = 1, \dots, m$), and where $\|\cdot\|$ denotes the Euclidean norm (in $\mathbf{R}^m$). [1] For reasons of convenience when comparing to Newton's method we assume f to be at least *twice* continuously differentiable, though the Gauss–Newton and Levenberg–Marquardt algorithms can be constructed equally well if f is only *once* continuously differentiable. The associated Jacobian mapping is denoted by $J : \mathbf{R}^n \to \mathbf{R}^{m\times n}$ and defined in each point $x \in \mathbf{R}^n$ as

$$J(x) = \begin{pmatrix} \frac{\partial f^1}{\partial x^1}(x) & \cdots & \frac{\partial f^1}{\partial x^n}(x) \\ \vdots & & \vdots \\ \frac{\partial f^m}{\partial x^1}(x) & \cdots & \frac{\partial f^m}{\partial x^n}(x) \end{pmatrix} \tag{2.2}$$

We adopt the notational convention that coordinates (and coordinate functions) are indexed by a superscript, whereas quantities changing every iteration are indexed by a subscript k, denoting the iteration number.

2.2 Outline of the Levenberg–Marquardt algorithm

As indicated in the introduction, to solve this nonlinear least squares problem numerically, one can apply the method of Gauss-Newton (GN), cf., e.g., [7, 31], or make use of an extension of this method, known as the Levenberg–Marquardt (LM) algorithm. Below we describe a well-known robust implementation of this method, as first described in [27]. In Figure 1 a flowdiagram of this algorithm is given. We shall discuss each of its steps in more detail.

ALGORITHM (Levenberg–Marquardt)

STEP I
Initialize all variables. Set iteration counter $k = 0$.

STEP II
Increase k to $k + 1$. Calculate a GN step (i.e., a LM step corresponding to $\lambda_k = 0$).

STEP III
Test if the GN step lies inside the trust–region.
If not so, then apply "subalgorithm A" to determine an acceptable LM parameter $\lambda_k > 0$ for which the resulting step lies approximately on the trust–region boundary.

STEP IV
Calculate the performance ρ_k, defined as the ratio of actual to predicted reduction of the criterion function value. This yields a measure of the validity of the linearization of the residual mapping in the proposed direction.

[1] In our applications of the Levenberg–Marquardt algorithm in [29] we shall mostly be dealing with situations where m is much larger than n. In the present description, however, there will be no such assumption: the algorithm applies equally well to the case where $m < n$. But in the latter case one must be aware that the Gauss–Newton approximation to the Hessian of Φ at x will always be singular.

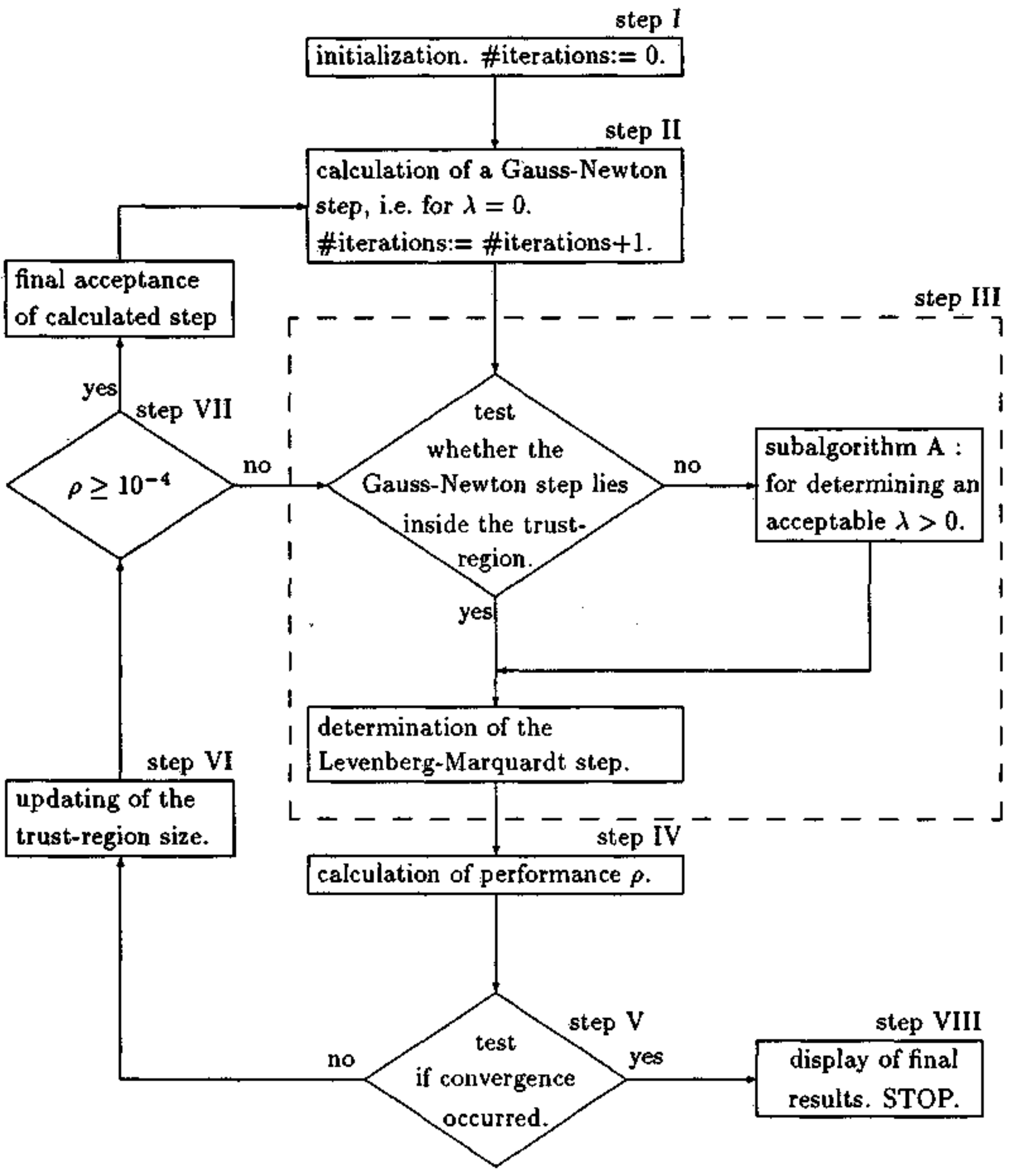
step I
initialization. #iterations:= 0.
step II
calculation of a Gauss-Newton step, i.e. for λ = 0. #iterations:= #iterations+1.
final acceptance of calculated step
step III
yes
step VII
ρ ≥ 10⁻⁴
no
test whether the Gauss-Newton step lies inside the trust-region.
no
subalgorithm A : for determining an acceptable λ > 0.
yes
determination of the Levenberg-Marquardt step.
step VI
updating of the trust-region size.
step IV
calculation of performance ρ.
step V
no
test if convergence occurred.
yes
step VIII
display of final results. STOP.

Figure 1. Flowdiagram of the Levenberg–Marquardt algorithm, according to Moré's trust–region implementation.

STEP V

Test if convergence has occurred. For this one can use various numerical criteria, e.g. Moré's or Marquardt's.
If so, proceed with step VIII, else continue with step VI.

STEP VI

Update the trust-region size Δ_k, using ρ_k.

STEP VII

Decide on the acceptance of the proposed LM step, according to the following rule.
If $\rho_k \geq 10^{-4}$ then accept the proposed step and proceed with step II (i.e., a new iteration).
If $\rho_k < 10^{-4}$, the trust-region size will just have been decreased in step VI. Therefore, skip step II and proceed with step III to find a better value of $\lambda_k > 0$.

STEP VIII

Calculate and display the final results. Stop the algorithm.

2.3 Detailed description of the steps in the LM algorithm

The original description of the above implementation of the LM algorithm is given in [27]. However, we have found that in that paper there are several unclarified issues and inaccuracies. In order to address and correct these we give a detailed account of all steps in the algorithm. This then also provides us with the necessary background for quickly developing a Riemannian version of Moré's trust-region implementation (see Sect. 3). Moreover, it enables us to come up with new insights concerning the various scaling strategies involved.

Ad step I — Initialization.

The initialization involves the choice of a *starting point* $x_0 \in \mathbf{R}^n$ and an initial *trust-region* of ellipsoidal shape, consisting of all points $x_0 + p$ for which $\|D_0 p\| \leq \Delta_0$ where D_0 is a nonsingular $n \times n$ matrix and Δ_0 a positive scalar. All of these must be user supplied, and good choices will depend on the specific problem at hand.
Moré advises to choose $D_0 = \mathrm{diag}(d_0^1, \ldots, d_0^n)$ with $d_0^i = \|\frac{\partial f}{\partial x^i}(x_0)\|$, with the aim of improving numerical scaling. Subsequently he proposes three different ways for updating D_k (see step VII). An obvious alternative is provided by choosing $D_k = I$ for all k. This implies that one accepts the scaling corresponding to the choice of basis that is implicit in the coordinates being used. Other choices are conceivable; as a matter of fact we shall make a well-motivated third proposal in our extension of the algorithm to Riemannian manifolds (cf. Sect. 3).
For fixed D_0 the value of Δ_0 determines the *size* of the initial trust-region. One can choose $\Delta_0 = 1$, since an inadequate initial size of the trust-region will generally be adapted fairly quickly. Notice that the trust-region size influences the acceptance of a GN step and also the determination of a LM step with $\lambda_k > 0$. Therefore, the choice of Δ_0 has influence on the iteration path.
Yet another quantity to be initialized is the *flexibility factor* $\sigma \in (0,1)$. Moré proposes to choose $\sigma = 0.1$, which we also have found to be satisfactory. The interpretation of σ is as follows. When calculating an LM step, which is in principle required to lead us to the boundary of the trust-region, we express by our choice of σ that we are going to accept also steps for which $\|D_k p_k\|$ is not exactly equal to Δ_k, but lies in the interval $[(1-\sigma)\Delta_k, (1+\sigma)\Delta_k]$.
Further, we must choose some *tolerance levels*, X_{tol} and F_{tol}, that are of importance for detecting convergence (see step V). Following [27] we set $X_{tol} = F_{tol} = 10^{-8}$.
Finally, we set iteration counter $k = 0$ and calculate residual vector $f_k = f(x_k)$ as well as its corresponding Jacobian matrix $J_k = J(x_k)$.

Ad step II — Gauss–Newton step.

At point x_k the residual mapping can be linearized, yielding

$$f(x_k + p) = f_k + J_k p + O(\|p\|^2) \tag{2.3}$$

Thus, minimization of $\Phi(x_k + p)$, being equivalent to minimization of $\|f(x_k + p)\|$, can be approximated locally around x_k by the *linear* least squares problem of finding a minimizer of $\|f_k + J_k p\|$ (or equivalently of $\|f_k + J_k p\|^2$). This approach is known as quasi-linearization. Let us denote

$$p_k(0) = \arg\min_{p \in \mathbb{R}^n} \|f_k + J_k p\|^2 \tag{2.4}$$

As is well known, in case of J_k having full column rank n, vector $p_k(0)$ is unique and identical to the step calculated in the (undamped) GN method (cf., e.g., [7]). In this case it is given by $p_k(0) = -(J_k^T J_k)^{-1} J_k^T f_k$. However, in situations where J_k does not have full column rank n (as will always be the case for $m < n$) the GN method breaks down. Then matrix $J_k^T J_k$ is no longer invertible.
Nevertheless, the approximating linear least squares problem above still admits a solution, albeit no longer a unique one. In Moré's approach a particular solution is calculated, applying QR-decomposition with column pivoting to J_k. Such a QR-decomposition can be accomplished numerically stable, using Householder transformations. We proceed along the following lines.

- Apply QR-decomposition with column pivoting to J_k, yielding matrices Q_k ($m \times m$, orthogonal), π_k ($n \times n$, permutation and therefore also orthogonal), T_k ($r_k \times r_k$, upper triangular, nonsingular), S_k ($r_k \times (n - r_k)$) satisfying:

$$J_k = Q_k^T \begin{pmatrix} T_k & S_k \\ O & O \end{pmatrix} \pi_k^T \tag{2.5}$$

  where r_k denotes the rank of J_k, and the zero matrix blocks are of appropriate size.

- Construct a generalized left-inverse J_k^- of J_k, of size $n \times m$, as follows:

$$J_k^- = \pi_k \begin{pmatrix} T_k^{-1} & O \\ O & O \end{pmatrix} Q_k \tag{2.6}$$

  If $r_k = n$ then J_k^- will be a true left-inverse of J_k. In any case will hold $J_k J_k^- J_k = J_k$ and $J_k J_k^-$ is symmetric. However, for square, rank-deficient J_k one must be aware that J_k^- is in general *not* equal to the Moore-Penrose pseudo-inverse J_k^+ (cf. App. A).

- Calculate the GN step $\tilde{p}_k(0)$ via

$$\tilde{p}_k(0) = -J_k^- f_k \tag{2.7}$$

  In case $r_k = n$ this is the standard GN step. Otherwise it still is a step minimizing the approximating linear least squares criterion.

Lemma 2.1 *Vector $\tilde{p}_k(0)$ obtained by the procedure above yields a minimizing argument of $\|J_k p + f_k\|^2$.*

Proof We have that

$$\|J_k p + f_k\|^2 = \left\| Q_k^T \begin{pmatrix} T_k & S_k \\ O & O \end{pmatrix} \pi_k^T p + f_k \right\|^2 = \left\| \begin{pmatrix} T_k & S_k \\ O & O \end{pmatrix} \pi_k^T p + Q_k f_k \right\|^2 =$$
$$= \left\| \begin{pmatrix} T_k & S_k \\ O & O \end{pmatrix} \pi_k^T p + \begin{pmatrix} u_k \\ v_k \end{pmatrix} \right\|^2 = \| \begin{pmatrix} T_k & S_k \end{pmatrix} \pi_k^T p + u_k \|^2 + \|v_k\|^2$$

where the second equality follows from the orthogonality of Q_k and the third equality is motivated by the definition $\begin{pmatrix} u_k \\ v_k \end{pmatrix} = Q_k f_k$, with u_k an r_k-vector and v_k an $(m - r_k)$-vector. Now, it is clear that the second term of the remaining expression remains uninfluenced by the choice of p. Furthermore, the choice $p = \tilde{p}_k(0)$, as calculated above, reduces the first term of the remaining expression to zero. Therefore, $\tilde{p}_k(0)$ minimizes $\|J_k p + f_k\|^2$. □

In case $r_k < n$ this minimizing vector $\tilde{p}_k(0)$ is nonunique. In App. A we show that the set of all minimizing vectors is given by the affine space $\{\tilde{p}_k(0) + s_k \,|\, s_k \in \ker(J_k)\}$. There we also show that among this set there is a unique element for which $\|D_k p\|^2$ is minimal, which we shall denote by $p_k(0)$. In view of subalgorithm A, discussed in the comments to step III and more extensively in App. A, it is of importance to take $p_k(0)$ as the GN step.

Ad step III — Levenberg–Marquardt step with $\lambda_k > 0$.

We consider trust–regions of ellipsoidal shape, that are described at iteration k by the set $\{x_k + p \,|\, \|D_k p\| \leq \Delta_k\}$. It is our policy never to accept steps that lead us outside of a trust–region.

Because of the superlinear convergence property of the standard (undamped) GN method under certain conditions, see for instance [7], it is in general advisable to exploit GN steps when they are acceptable, i.e., when they lead to a sufficient decrease in the criterion value. Thus, we start step III of the algorithm by testing whether the GN step $p_k(0)$ lies inside the trust–region. If this is so, we set $\lambda_k = 0$. If not, we proceed as follows.

In this latter situation, the idea due to [21] is to minimize simultaneously the approximate criterion $\|J_k p + f_k\|^2$ and the size of the increments $\|p\|_*^2$, where $\|\cdot\|_*$ denotes some norm on $\mathbf{R}^n$ (not necessarily the Euclidean). In the present implementation, the purpose of matrix D_k is to shape the trust–region, which can be viewed as adopting the norm $\|p\|_* = \|D_k p\|$. Apart from choosing the norm $\|\cdot\|_*$ we have to balance both objectives, which is achieved by introducing a Levenberg–Marquardt parameter $\lambda \geq 0$ and restricting to the combined (i.e. LM) criterion

$$\|J_k p + f_k\|^2 + \lambda \|D_k p\|^2 \tag{2.8}$$

Clearly, in case $\lambda = 0$ we are dealing with our original (approximate) criterion, whereas any larger value of λ restricts the size of the optimal step.

A minimizing vector p for the above criterion will be denoted by $p_k(\lambda)$, in agreement with our earlier notation $p_k(0)$. Due to the nonsingularity assumption on D_k this LM criterion will have a unique minimizing solution for $\lambda > 0$. The remaining question is of course how to choose an appropriate value λ_k for λ. In Moré's trust–region approach the idea is to choose λ_k such that the resulting solution $p_k(\lambda_k)$ lies on the boundary of the trust–region. Thus, the step p_k is determined via *constrained optimization* of the approximate criterion $\|J_k p + f_k\|^2$ over the trust–region, leaving for λ_k the interpretation of a Lagrange multiplier. For the sake of keeping the computational effort to find λ_k small, Moré introduces a "flexibility parameter" σ with respect to the trust–region size. We then shall accept λ_k if $\|D_k p_k(\lambda_k)\|$ lies in the interval $[(1-\sigma)\Delta_k, (1+\sigma)\Delta_k]$.

An appropriate value for λ_k is now found using an iterative scheme, originally due to [16] and speeded up in [27]. We refer to this procedure as "subalgorithm A" and it is discussed extensively in App. A. But here we first address the problem of calculating $p_k(\lambda)$ for a *given* positive value of λ, thereby exploiting the results of the computations for the GN step.

Calculation of $p_k(\lambda)$ for given $\lambda > 0$.

First of all, notice that the problem of minimizing the LM criterion (2.8) is equivalent to the problem of minimizing the linear least squares criterion

$$\left\| \begin{pmatrix} J_k \\ \sqrt{\lambda} D_k \end{pmatrix} p + \begin{pmatrix} f_k \\ 0 \end{pmatrix} \right\|^2 \tag{2.9}$$

Therefore, in view of the comments to step II, we have that: (a) the problem can be solved using QR–decomposition with column pivoting; (b) its solution is unique, due to the nonsingularity of D_k and the fact that $\sqrt{\lambda}$ is strictly positive so that full column rank occurs; (c) $p_k(\lambda) = -(J_k^T J_k + \lambda D_k^T D_k)^{-1} J_k^T f_k$.

It turns out that it is possible to make use of the information earlier obtained in step II while calculating the solution to the present problem. Indeed, it was shown in the proof of Lemma 2.1 that minimizing $\|J_k p + f_k\|^2$ is equivalent to minimizing $\| \begin{pmatrix} T_k & S_k \end{pmatrix} \pi_k^T p + u_k\|^2 + \|v_k\|^2$. Therefore we can just as well consider

$$\left\| \begin{pmatrix} T_k & S_k \\ \multicolumn{2}{c}{\sqrt{\lambda}\, D_k \pi_k} \end{pmatrix} \pi_k^T p + \begin{pmatrix} u_k \\ 0 \end{pmatrix} \right\|^2 + \|v_k\|^2 \tag{2.10}$$

Premultiplication of the last block–row in the first term by any $n \times n$ orthogonal matrix does not change the criterion value. Moré chooses matrix π_k^T for this purpose, motivated by the fact that his matrices D_k are diagonal, in which case $\pi_k^T D_k \pi_k$ is diagonal also. Further, we can omit the second term because it is independent of p. So we arrive at the following expression for the criterion to be minimized:

$$\left\| \begin{pmatrix} T_k & S_k \\ \multicolumn{2}{c}{D_k(\lambda)} \end{pmatrix} \pi_k^T p + \begin{pmatrix} u_k \\ 0 \end{pmatrix} \right\|^2 \tag{2.11}$$

where $D_k(\lambda) = \sqrt{\lambda}\, \pi_k^T D_k \pi_k$.
It is important to notice that the sizes of the matrices and vectors in this final expression are independent of m. Matrix $\begin{pmatrix} T_k & S_k \\ \multicolumn{2}{c}{D_k(\lambda)} \end{pmatrix}$ is of size $(n + r_k) \times n$ and $\begin{pmatrix} u_k \\ 0 \end{pmatrix}$ is an $(n+r_k)$–vector, with $r_k \leq n$. This implies a large reduction of required computer memory space for data storage in case m is much larger than n, as compared to the more naieve approach where matrix $\begin{pmatrix} J_k \\ \sqrt{\lambda}\, D_k \end{pmatrix}$ is dealt with explicitly.
The actual computations now proceed along the following lines.

- Calculate matrix $D_k(\lambda) = \sqrt{\lambda}\, \pi_k^T D_k \pi_k$.
- Using Givens transformations, which are orthogonal, construct an orthogonal $(n + r_k) \times (n + r_k)$ matrix $W_k(\lambda)$ such that

$$\begin{pmatrix} T_k & S_k \\ \multicolumn{2}{c}{D_k(\lambda)} \end{pmatrix} = W_k(\lambda)^T \begin{pmatrix} R_k(\lambda) \\ O \end{pmatrix} \tag{2.12}$$

  where $R_k(\lambda)$ is an $n \times n$ nonsingular, upper triangular matrix.
- Define $\begin{pmatrix} w_k(\lambda) \\ z_k(\lambda) \end{pmatrix} = W_k(\lambda) \begin{pmatrix} u_k \\ 0 \end{pmatrix}$.
- Calculate the LM step as

$$p_k(\lambda) = -\pi_k R_k(\lambda)^{-1} w_k(\lambda) \tag{2.13}$$

Lemma 2.2 *Vector $p_k(\lambda)$, calculated via the scheme above, minimizes the LM criterion (2.8).*

Proof We can write

$$\begin{aligned} &\left\| \begin{pmatrix} T_k & S_k \\ \multicolumn{2}{c}{D_k(\lambda)} \end{pmatrix} \pi_k^T p + \begin{pmatrix} u_k \\ 0 \end{pmatrix} \right\|^2 = \left\| W_k(\lambda)^T \begin{pmatrix} R_k(\lambda) \\ O \end{pmatrix} \pi_k^T p + \begin{pmatrix} u_k \\ 0 \end{pmatrix} \right\|^2 = \\ &= \left\| \begin{pmatrix} R_k(\lambda) \\ O \end{pmatrix} \pi_k^T p + W_k(\lambda) \begin{pmatrix} u_k \\ 0 \end{pmatrix} \right\|^2 = \left\| \begin{pmatrix} R_k(\lambda) \\ O \end{pmatrix} \pi_k^T p + \begin{pmatrix} w_k(\lambda) \\ z_k(\lambda) \end{pmatrix} \right\|^2 = \\ &= \|R_k(\lambda)\pi_k^T p + w_k(\lambda)\|^2 + \|z_k(\lambda)\|^2 \end{aligned} \tag{2.14}$$

Obviously, the choice $p = p_k(\lambda)$ is the unique vector reducing the first term of the latter expression to zero, whereas the second term is independent of p. □

Subalgorithm A: determination of an acceptable $\lambda_k > 0$.

In order to determine a value for λ for which the LM step $p_k(\lambda)$ lies sufficiently close to the boundary of the trust-region $\|D_k p\| \le \Delta_k$, we introduce a function $\phi_k : \mathbf{R}_+ \to \mathbf{R}$, defined by

$$\phi_k(\alpha) = \|D_k p_k(\alpha)\| - \Delta_k \tag{2.15}$$

It is clear that, for $\alpha > 0$, $p_k(\alpha)$ lies in the acceptable interval $[(1-\sigma)\Delta_k, (1+\sigma)\Delta_k]$ if and only if $|\phi_k(\alpha)| \le \sigma\Delta_k$.
In case $\phi_k(0) \le 0$, the GN step lies inside the trust-region so that subalgorithm A is not needed. We can therefore restrict ourselves to the case $\phi_k(0) > 0$. In App. A we prove that in general ϕ_k is a strictly monotonically decreasing convex function on $\mathbf{R}_+$, tending to $-\Delta_k < 0$ as α tends to infinity. This implies that there exists a unique value α^* for which $\phi_k(\alpha^*) = 0$, i.e. $p_k(\alpha^*)$ lies on the boundary of the trust-region. Moreover, we can apply the Newton–Raphson method to approximate this value of α with second order rate of convergence. Due to the convexity of ϕ_k the Newton–Raphson iterate will always yield a lower bound to the optimal value. As pointed out in [16] it is possible to develop an alternative iterative scheme that is much more efficient than Newton–Raphson. This scheme exploits the expected (approximate) structure of $\phi_k(\cdot)$ and has quadratic convergence properties too. However, it has to be safeguarded by the use of upper and lower bounds on the iterates if it is to converge. Moré developed a modification to Hebden's original method so that it generally will find an acceptable value of α more quickly. Actually, Moré claims that on the average less than two iterations are needed in practice, something that agrees with our own experience. A proof of convergence of subalgorithm A can be found in App. A.

SUBALGORITHM A (Hebden–Moré)

STEP A-I Initialization and first iteration.
Set subiteration counter $\ell = 0$ and let $\alpha_0 = 0$.
If $r_k = n$, calculate $\phi_k(\alpha_0) = \|D_k p_k(0)\| - \Delta_k$ and $\phi_k'(0)$ (see step A-IV).
Set lower bound $lb_0 = -\frac{\phi_k(0)}{\phi_k'(0)}$. If $r_k < n$, set lower bound $lb_0 = 0$.
Calculate upper bound $ub_0 = \frac{\|(J_k D_k^{-1})^T f_k\|}{\Delta_k}$. (See App. A for a validation.)
Set $\alpha_{\ell+1} = \max(10^{-3} ub_\ell, \sqrt{lb_\ell\, ub_\ell}\,)$ in order to obtain an estimate in the open interval (lb_ℓ, ub_ℓ).
Set $\ell = 1$. Calculate $\phi_k(\alpha_\ell) = \|D_k p_k(\alpha_\ell)\| - \Delta_k$ and $\phi_k'(\alpha_\ell)$ (see step A-IV).

STEP A-II Test of convergence and adaptation of the bounds.
Test for convergence: if $|\phi_k(\alpha_\ell)| \le \sigma\Delta_k$ then stop this subalgorithm A.
Else, calculate a new lower bound using the Newton–Raphson scheme, i.e. set $lb_\ell = \max(lb_{\ell-1}, \alpha_\ell - \frac{\phi_k(\alpha_\ell)}{\phi_k'(\alpha_\ell)})$.
In case $\phi_k(\alpha_\ell) < 0$, adapt the upper bound via $ub_\ell = \min(ub_{\ell-1}, \alpha_\ell)$. If not, leave the upper bound unchanged, that is, set $ub_\ell = ub_{\ell-1}$.

STEP A-III Calculation of the next approximate.
To calculate a new iterate $\alpha_{\ell+1}$ the following formula is used.

$$\alpha_{\ell+1} = \alpha_\ell - \frac{\phi_k(\alpha_\ell) + \Delta_k}{\Delta_k}\left(\frac{\phi_k(\alpha_\ell)}{\phi_k'(\alpha_\ell)}\right) \tag{2.16}$$

If this procedure leads to a value of $\alpha_{\ell+1}$ that is not inside the open interval (lb_ℓ, ub_ℓ), set $\alpha_{\ell+1} = \max(10^{-3} ub_\ell, \sqrt{lb_\ell\, ub_\ell}\,)$.
Set $\ell = \ell + 1$.

STEP A-IV Determination of $\phi_k(\alpha_\ell)$ and $\phi_k'(\alpha_\ell)$.
The calculation of $\phi_k(\alpha_\ell)$ follows from its definition: $\phi_k(\alpha_\ell) = \|D_k p_k(\alpha_\ell)\| - \Delta_k$, where $p_k(\alpha_\ell)$ should be calculated as discussed earlier.

Calculation of $\phi'_k(\alpha_\ell)$ makes use of the following formula, established in App. A.

$$\phi'_k(\alpha_\ell) = -\|q_k(\alpha_\ell)\| \left\| R_k(\alpha_\ell)^{-T} \left(\frac{\pi_k^T D_k^T q_k(\alpha_\ell)}{\|q_k(\alpha_\ell)\|} \right) \right\|^2 \tag{2.17}$$

Here, $q_k(\alpha_\ell) = D_k p_k(\alpha_\ell)$.
After calculation of $\phi_k(\alpha_\ell)$ and $\phi'_k(\alpha_\ell)$, proceed with step A-II.

In this subalgorithm the choice of $\alpha_{\ell+1}$ inside the open interval (lb_ℓ, ub_ℓ) via $\alpha_{\ell+1} = \max(10^{-3} ub_\ell, \sqrt{lb_\ell \, ub_\ell})$ is based on the heuristic that one wants an iterate that is biased towards lb_ℓ, whence the second argument. The first argument is present to protect against exceedingly small values of lb_ℓ, in particular against the initial value $lb_0 = 0$.
On leaving subalgorithm A, we set $\lambda_k = \alpha_\ell$. The LM step, vector p_k, is set to $p_k = p_k(\lambda_k)$.

Ad step IV — Performance ρ_k.

Computation of the performance ρ_k is organized as follows. By definition we have that

$$\rho_k = \frac{\Phi(x_k) - \Phi(x_k + p_k)}{\Phi(x_k) - \frac{1}{2}\|f_k + J_k p_k\|^2} \tag{2.18}$$

which can be rewritten as

$$\rho_k = \frac{1 - \left(\frac{\|f_{k,+}\|}{\|f_k\|}\right)^2}{\left(\frac{\|J_k p_k\|}{\|f_k\|}\right)^2 + \left(\sqrt{2\lambda_k}\frac{\|D_k p_k\|}{\|f_k\|}\right)^2} \tag{2.19}$$

where $f_{k,+}$ denotes $f(x_k + p_k)$.
As pointed out in [27], the advantage of this last formula is that it prevents us from generating unnecessary overflows when calculating ρ_k. Since we are only interested in values of ρ_k that are nonnegative (see step VI), we can set $\rho_k = 0$ in case $\|f_{k,+}\| > \|f_k\|$.
The interpretation of ρ_k is obvious. If $\rho_k \approx 1$ this means that the linearization applied to $f(\cdot)$ is almost exactly valid for the step p_k at x_k. For larger values of ρ_k it shows that the decrease in $\Phi(\cdot)$ is even more than expected. In those cases the trust-region can be increased, showing that we expect the linearization to be profitable in a larger area than was currently the case. However, for smaller values of ρ_k (and especially negative ones) we see that the linearization is not justified for the step p_k. This implies that we should decrease the trust-region size.

Ad step V — Convergence detection.

There are several ways to test on numerical convergence. Marquardt's test [26] consists of checking whether for all $i = 1, \ldots, n$ we have that

$$|p_k^i| \le 10^{-4}(|x_k^i| + 10^{-3}) \tag{2.20}$$

If this is indeed the case, then this means that the absolute values of the increments with respect to all coordinates are relatively small, compared to the absolute values of the corresponding coordinates themselves, showing that further iterations are not likely to lead to substantial changes in the approximations. Of course there is freedom of choice with respect to the factor 10^{-4}. The term 10^{-3} is added to protect against situations where the optimum has coordinates $x_*^i = 0$.
According to [3] this stopping criterion has worked well in practice, but tends to be somewhat on the conservative side, allowing for more iterations than strictly necessary.

Another test for convergence is provided by Moré, and it is related to the trust-region approach of this specific algorithm. It consists of checking whether one of two inequalities is satisfied. These are

$$\Delta_k \le X_{tol}\|D_k x_k\| \tag{2.21}$$

and

$$\left(\frac{\|J_k p_k\|}{\|f_k\|}\right)^2 + \left(\sqrt{2\lambda_k}\frac{\|D_k p_k\|}{\|f_k\|}\right)^2 \le F_{tol} \tag{2.22}$$

The first of these two reflects the desirable property that Δ_k be relatively small (so that the trust-region be small), compared to the norm of x_k. This norm of x_k is taken to be $\|x_k\|_* = \|D_k x_k\|$, so that the components of x_k are weighted in a similar way as the components of p_k, when calculating a step. Of course, this expresses the fact that one wants the components of possible future steps to be small, compared to the absolute values of the corresponding components of x_k.
The second reflects that the denominator of ρ_k be small, that is, the relative expected reduction in the criterion value should be small. Of course, at a stationary point this quantity would be zero. (One does not address the numerator of ρ_k because of its unreliability in case the applied linearization is not valid.)
The choice of the tolerances X_{tol} and F_{tol} is up to the user. They should be such that, in combination with the choice of D_0 and Δ_0, the algorithm does not immediately stop after one iteration, something that can happen when the initial trust-region is too small.
Our choice of stopping criterion is Moré's. It has worked satisfactory in all of our experiments. For a comparison between the performance of the criteria of Moré and Marquardt, see App. C.

Ad step VI — Trust-region size updating.
The updating of the trust-region size, in particular of Δ_k, is based on the value of ρ_k. As explained under step IV, a value of ρ_k approximately equal to 1, or larger, asks for an increase of Δ_k, whereas a value close to 0, or less, demands a decrease. Therefore, assuming $\lambda_k > 0$, Moré distinguishes between three situations.

1. $\rho_k \le \frac{1}{4}$.
   The trust-region should be decreased. This is done by a factor that lies in between $\frac{1}{10}$ and $\frac{1}{2}$. We denote this factor by μ_k, so that the updating formula becomes $\Delta_{k+1} = \mu_k \Delta_k$. The exact value of μ_k is calculated in accordance with [8]. The idea is to fit a quadratic to the function $\delta_k(\cdot)$, defined at $\theta \in \mathbf{R}$ by

$$\delta_k(\theta) = \Phi(x_k + \theta p_k) \tag{2.23}$$

   We require $\delta_k(0)$, $\delta_k'(0)$ and $\delta_k(1)$ to coincide with the corresponding values of the quadratic fit. Then μ_k is chosen as the unique minimizer of $\delta_k(\cdot)$. This is easily seen to lead to the formula

$$\mu_k = \frac{\frac{1}{2}\gamma_k}{\gamma_k + \frac{1}{2}\left[1 - \left(\frac{\|f_{k,+}\|}{\|f_k\|}\right)^2\right]} \tag{2.24}$$

   where $\gamma_k = \frac{p_k^T J_k^T f_k}{\|f_k\|^2} \in [-1, 0]$ is given by

$$\gamma_k = \left(\frac{\|J_k p_k\|}{\|f_k\|}\right)^2 + \left(\sqrt{\lambda_k}\frac{\|D_k p_k\|}{\|f_k\|}\right)^2 \tag{2.25}$$

   Notice that γ_k is different from the denominator of ρ_k, since a factor 2 no longer appears under the square-root sign.
   In case these formulas do not lead to a value in the required interval we replace μ_k by the closest endpoint. Obviously it is only necessary to perform calculations for $\|f_k\|^2 < \|f_{k,+}\|^2 < 10\|f_k\|^2$.
2. $\frac{1}{4} < \rho_k < \frac{3}{4}$.
   In this case we still achieve substantial reduction in the actual criterion, albeit not as much as predicted by the linearization of the residual function. Therefore we think an increase of the trust-region size to be unjustified, whereas a decrease is still not necessary, so that Δ_k is left unchanged: $\Delta_{k+1} = \Delta_k$.

3. $\rho_k \geq \frac{3}{4}$.
   The trust–region can be increased, since the linearization is apparently valid. This is done by roughly a factor 2. We set $\Delta_{k+1} = 2\|D_k p_k\| \approx 2\Delta_k$.

We shall shed some light upon the situation where $\lambda_k = 0$. This means that the GN step is accepted. But the GN method is known to have superlinear convergence properties under certain circumstances, particularly when the linearization of the residual function is valid for the corresponding steps. Therefore, we regard an accepted GN step as something special, so that, if $\rho_k > \frac{1}{4}$, we set $\Delta_{k+1} = 2\|D_k p_k\|$. This strategy incorporates the possibility of automatically decreasing Δ_k as long as GN steps are accepted, so especially when GN is converging. Notice that this is necessary for the algorithm to terminate if the problem has a zero residual, in view of the first criterion of Moré for convergence.
In case $\rho_k \leq \frac{1}{4}$ we modify Δ_k in the same way as described above for $\lambda_k > 0$.
We conclude by remarking that of course the choice of the boundary values $\frac{1}{4}$ and $\frac{3}{4}$ is in principle up to the user, but they appear to work well. A similar statement applies to the interval $[\frac{1}{10}, \frac{1}{2}]$ for μ_k, if $\rho_k \leq \frac{1}{4}$.

Ad step VII — Acceptance of a calculated step.
We have the policy of not adapting the *shape* of the trust–region, unless we have come to another point x_{k+1}. This point x_{k+1} is found, after calculation of a step p_k, as

$$x_{k+1} = x_k + p_k \tag{2.26}$$

but we accept it only if $\rho_k \geq 10^{-4}$, because then there is still some substantial improvement in the criterion value. We use a strictly positive bound 10^{-4} in order to avoid problems caused by numerical round–off errors in case $\rho_k \approx 0$.
In case $\rho_k < 10^{-4}$, the trust–region size has just been decreased in the previous step VI, so that new calculations may now turn out to be more successful.
Adequate shaping of the trust–region is a subject in itself. As already mentioned in the comments to step I, there are several possible strategies and the best strategy is most likely to depend on the specific optimization problem at hand. Actually, Moré has compared three different strategies ([27]) of which the following rule, the so–called *adaptive* rule, turned out to be most successful. Here one puts

$$D_{k+1} = \mathrm{diag}(d^1_{k+1}, \ldots, d^n_{k+1}), \qquad \text{with } d^i_{k+1} = \max(d^i_k, \left\|\frac{\partial f}{\partial x^i}(x_{k+1})\right\|) \tag{2.27}$$

However, this strategy is only justified *heuristically*, and particularly the fact that what Moré calls the *continuous* strategy is inferior, makes the use of the present rule doubtful. This continuous strategy is defined by putting

$$D_{k+1} = \mathrm{diag}(d^1_{k+1}, \ldots, d^n_{k+1}), \qquad \text{with } d^i_{k+1} = \left\|\frac{\partial f}{\partial x^i}(x_{k+1})\right\| \tag{2.28}$$

A third possible rule is called the *initial* strategy, where D_0, as specified in the comments to step I, is left unchanged throughout the further algorithm.
These three scaling strategies share the property that they make the algorithm *scaling invariant*, that is, if we transform to new coordinates by scaling in the old coordinate directions, also the iteration paths will be scaled versions of each other.
However, we are the opinion that it is more safe, in some sense, to rely upon a fourth possible scaling, implicit in the coordinates being used. This consists of applying a *nonscaling* strategy, and choosing $D_k = I$ for all $k \geq 0$. Of course, this requires from the user of the algorithm a certain intuition about what are good and bad coordinates, but on the other hand we found that application of this very strategy to the four test–problems treated in [27] (on which the choice in favor of the "adaptive" strategy is founded) led to superior convergence behaviour! This can be explained as follows, using a counter–argument against Moré's heuristic. The

automatic scaling strategies (either "adaptive" or "continuous") tend to put a heavy penalty on those directions for which the function decreases rapidly. Thus, they not merely prevent unreliably large steps in such directions, but they have a tendency to block the most profitable ones. As a result, only small steps are taken. But there is no apparent need for such a strategy, since the updating scheme for Δ_k is especially suited to handle such situations. In App. C we present the results of our own experiments, carried out with the four test-problems studied in [27], supporting our point of view in the discussion above.
Another aspect of Moré's scaling strategies is that restriction is made to *diagonal* matrices D_k, which might be another cause of the "failed convergence" results in some of his experiments. In Sect. 3 we discuss an alternative approach towards the choice of D_k that forms the basis of our Riemannian version of the LM algorithm. There we translate the problem of determining what is a suitable scaling to the problem of choosing an adequate Riemannian metric on the manifold at hand. This will make the algorithm completely coordinate independent.

Ad step VIII — Final results.

Among the final results one could present, apart from the point x_k, also the length of the gradient and the value of Δ_k, as well as the number of iterations k itself. Another thing in which one might be interested is whether convergence took place via GN steps or if LM steps were essential to reach the optimum.

3 A Riemannian version of the Levenberg–Marquardt algorithm

We shall now expose how the standard version of the LM algorithm, as described in the previous section, can be modified to act on a Riemannian manifold in a parametrization independent way. We start by introducing some notation appropriate for the new problem setting, mainly based on [5].

Let M denote a (smooth) differentiable manifold of dimension n, with for each point $p \in M$ the tangent space to M at p denoted by $T_p(M)$. Further, let $\mathcal{R} : T_p(M) \times T_p(M) \to \mathbf{R}$ denote a positive 2–form on M which defines a Riemannian metric. It is a key result in differential geometry that each (smooth) differentiable manifold can be endowed with a Riemannian metric (cf.,e.g., [5]).

Suppose (U, ϕ) is a coordinate neighbourhood containing p, corresponding to local coordinates denoted by $x^1, \ldots, x^n$. (Thus, U is an open subset of M and ϕ is a homeomorphism of U to an open subset of $\mathbf{R}^n$. The local coordinates relate to the image space of ϕ.) The 2–form $\mathcal{R}$ induces a norm on $T_p(M)$ that can be represented in these local coordinates $x = (x^1, \ldots, x^n)^T$ by a positive definite matrix $R(x)$ such that for each tangent vector $\dot{p} \in T_p(M)$

$$\|\dot{p}\|^2_{\mathcal{R}} = \dot{x}^T R(x) \dot{x} \tag{3.1}$$

where $\dot{x} = (\dot{x}^1, \ldots, \dot{x}^n)^T$ denotes the vector of coefficients corresponding to $\dot{p}$ with respect to the naturally induced basis $\{\frac{\partial}{\partial x^1}\big|_p, \ldots, \frac{\partial}{\partial x^n}\big|_p\}$ for $T_p(M)$, and $\|\cdot\|_{\mathcal{R}}$ denotes the norm on $T_p(M)$ induced by the Riemannian metric.

If (V, ψ) denotes an alternative neighbourhood containing p, with local coordinates denoted by $y^1, \ldots, y^n$, we have similarly (with a slight abuse of notation) that

$$\|\dot{p}\|^2_{\mathcal{R}} = \dot{y}^T R(y) \dot{y} \tag{3.2}$$

The matrix representations of $\mathcal{R}$, in local coordinates x and y respectively, are related by

$$R(y) = \left(\frac{\mathrm{d}x}{\mathrm{d}y}\right)^T R(x) \left(\frac{\mathrm{d}x}{\mathrm{d}y}\right), \qquad R(x) = \left(\frac{\mathrm{d}y}{\mathrm{d}x}\right)^T R(y) \left(\frac{\mathrm{d}y}{\mathrm{d}x}\right) \tag{3.3}$$

all evaluated at the same point $p \in M$. Now let Φ be redefined as the function $\Phi : M \to \mathbf{R}$ given by $\Phi(p) = \frac{1}{2}\|f(p)\|^2$, where $f : M \to \mathbf{R}^m$ is redefined to be an at least twice continuously differentiable mapping from M to $\mathbf{R}^m$. As before in Sect. 2, we call Φ the criterion function and f the residual mapping. It is again our objective to minimize Φ, but this time over M. (With the same abuse of notation as already introduced above we shall write $\Phi(x)$ and $f(x)$ when using local coordinates x, and $\Phi(y)$ and $f(y)$ when using local coordinates y.)

In local coordinates x, the Jacobian $J(x)$ of the residual mapping f at x is given by

$$J(x) = \frac{\partial f}{\partial x}(x) = \begin{pmatrix} \frac{\partial f^1}{\partial x^1}(x) & \cdots & \frac{\partial f^1}{\partial x^n}(x) \\ \vdots & & \vdots \\ \frac{\partial f^m}{\partial x^1}(x) & \cdots & \frac{\partial f^m}{\partial x^n}(x) \end{pmatrix} \tag{3.4}$$

Thus, we have that $J(x)$ and $J(y)$ are related by

$$J(y) = J(x) \left(\frac{\mathrm{d}x}{\mathrm{d}y}\right), \qquad J(x) = J(y) \left(\frac{\mathrm{d}y}{\mathrm{d}x}\right) \tag{3.5}$$

In local coordinates x, the gradient of $\Phi(x)$ is given by the *row* vector (by convention):

$$\nabla\Phi(x) = \frac{\partial \Phi}{\partial x}(x) = f(x)^T J(x) \tag{3.6}$$

Therefore $\nabla\Phi(x)$ and $\nabla\Phi(y)$ are related by

$$\nabla\Phi(y)^T = \left(\frac{\mathrm{d}x}{\mathrm{d}y}\right)^T \nabla\Phi(x)^T, \qquad \nabla\Phi(x)^T = \left(\frac{\mathrm{d}y}{\mathrm{d}x}\right)^T \nabla\Phi(y)^T \tag{3.7}$$

This shows that in general the gradient of a function Φ on M is *not* independent of the coordinates being used. To obtain a parametrization free object one has to make use of the Riemannian metric that is available. One then can define the so-called *Riemannian gradient* (cf. e.g., [1]), which *is* coordinate free, as

$$\nabla_{\mathcal{R}}\Phi(x) = \nabla\Phi(x)R(x)^{-1} = f(x)^T J(x)R(x)^{-1} \tag{3.8}$$

Indeed, one has

$$\nabla_{\mathcal{R}}\Phi(y)^T = \left(\frac{\mathrm{d}y}{\mathrm{d}x}\right) \nabla_{\mathcal{R}}\Phi(x)^T \tag{3.9}$$

showing that the Riemannian gradient can be identified with a tangent vector to M, and therefore is independent of the choice of coordinates. As a matter of fact, the Riemannian gradient occurs as the *maximizing normalized tangent direction*, where the normalization is in terms of the local Riemannian metric.

If one has the intention of using a gradient-based algorithm for minimizing Φ over M then, from a conceptual point of view, it is desirable that the search directions that are generated are independent of the local coordinates being used. It will then be possible to address the problem of minimizing Φ independent of the problem of choosing adequate local coordinates. As we have seen above, the method of steepest descent is in general *not* independent of the choice of local coordinates (cf. [28]), as opposed to its Riemannian version, and a similar statement holds true for Newton's method and related quasi-Newton strategies. It is worth noticing, however, that in the present case of nonlinear least squares the GN method can be applied and that this method *does* generate parametrization free search directions. See [14, 31]. As a matter of fact, the search direction generated in local coordinates x by the GN method is given by

$$s(x) = -[J(x)^T J(x)]^{-1} J(x)^T f(x) \tag{3.10}$$

When switching to local coordinates y we see that $s(x)$ and $s(y)$ are related according to

$$s(y) = \left(\frac{\mathrm{d}y}{\mathrm{d}x}\right) s(x) \tag{3.11}$$

showing that we can identify this search direction with a tangent vector to M. (It is actually possible to regard the GN method as a Riemannian steepest descent method, with the Riemannian metric induced (locally) via the imbedding of the image space of residual vectors $f(p)$ ($p \in M$) in Euclidean m-space. This induced Riemannian metric depends on the residual mapping only and is completely independent of $\mathcal{R}$. Cf. [14, 31].)

Since the LM method is based on the GN method and in certain situations even exploits the same search direction, we can hope to be able to modify the standard LM method to a parametrization free one. Using the interpretation of the LM method (without scaling) where it is considered to be an interpolation between the GN method and the method of steepest descent, it is natural to attempt to design a Riemannian version by requiring it to be an *interpolation between the GN method and the Riemannian steepest descent method.* This is achieved by choosing matrix D_k in iteration k of the algorithm such that

$$D_k^T D_k = R(x_k) \tag{3.12}$$

We will presently discuss the interpretation and further consequences of this choice.

First of all, notice that there is still some freedom left in the above choice for D_k. Premultiplication of D_k by an arbitrary orthogonal matrix will not disturb the property required above. (Actually, premultiplication of D_k by elements of the group of orthogonal matrices precisely generates *all* valid choices.) However, this freedom does *not* imply that the method would be ill-defined. What really *is* of importance for the algorithm is the value of $D_k^T D_k$ and that *is* well-defined. Indeed, if we take a look at the LM criterion we see that the term $\|D_k p\|^2$ occurs, which can be written as $p^T D_k^T D_k p$. Therefore, on the contrary, the freedom left in the choice of D_k does not make the algorithm ill-defined but can be exploited for optimizing numerical properties.

Second, notice that the fact that $\mathcal{R}$ defines a Riemannian metric on M implies that $R(x)$ is always positive definite. This guarantees the existence of D_k satisfying the requirement. We can for instance take a Choleski factor of $R(x)$.

Third, we not only have the interpretation of the LM method as an interpolation between the GN method and the method of steepest descent, but we can also regard the alternative point of view, where we consider it a *trust-region algorithm*. We now have that the trust-region is not merely shaped on the basis of a heuristic with respect to expected numerical properties, but instead *related to the Riemannian metric* defined by $\mathcal{R}$ on M. Indeed, the "trust-region" $\|D_k\dot{x}\| \leq \Delta_k$ now indicates that we shall accept only search directions related to tangent vectors $\dot{x}$ which have Riemannian norm at most Δ_k. This has the following consequences.

1. Suppose at point x_k we are dealing with a trust-region that is shaped by $R(x_k)$, with its size determined by Δ_k. If we want to change to coordinates y, then we need not adapt the value of Δ_k, because it is related to a "geometric" object: just changing $\dot{x}$ to $\dot{y}$ and $R(x)$ to $R(y)$ suffices.

2. In the GN method the induced Riemannian metric is independent of the Riemannian metric given by $\mathcal{R}$ on M. In the LM method the latter *does* play a role. It provides us with a means of expressing local numerical properties by adequately choosing a 2-form $\mathcal{R}$. This way one can imitate the situations in [27] occurring for the "initial" and "continuous" scaling strategies by choosing an appropriate Riemannian metric on $M = \mathbf{R}^n$. Of course, the "unscaled" algorithm can also be obtained this way, which merely comes down to accepting the standpoint that all coordinates are already properly scaled and that a change in each of them is of equal importance.
   Furthermore, Moré's arguments against the "continuous" scaling strategy now seem to be not really to the point: the gradient of a function already balances the sensitivity of it with respect to changes in each coordinate individually, so that a "discontinuous" scaling strategy, where the sensitivity of the function with respect to the coordinates at *old* iterates keeps on playing a role at the current point, is not properly based. As a matter of fact, one can consider his discouraging results using the "continuous" strategy as an argument against the use of the other scaling strategies as well. In this light it is perhaps of interest that application of the standard LM method *without* further scaling to the same four test-problems as reported in [27] has led to considerably faster convergence in all difficult situations. (See App. C).

3. One can regard each step in the LM algorithm as following from a *constrained* optimization procedure: the approximating (quasi-linearized) criterion is optimized over the trust-region. This same interpretation holds true for its Riemannian version if we use *normal coordinates*. Then the trust-region, which is currently defined on the tangent space to the manifold at x_k, defines a neighbourhood *on the manifold*.

So far we have only touched upon one aspect that is necessary for constructing a parametrization free algorithm, namely that the generated search direction should be related to the geometric object of a tangent vector (and therefore is parametrization free). However, in practice one will be working with coordinates from a certain coordinate chart and it is usual to take a step from the current iterate in the proposed direction with respect to these coordinates. This introduces a second source of parameter dependence. But there is a standard way to overcome it.
For this purpose one has to introduce the notion of *geodesics*, which play a similar role on a Riemannian manifold as straight lines do in Euclidean space. In particular, the shortest path that is entirely in the manifold M connecting two points p and q constitutes a geodesic. (For a definition and further discussion, see [5].) The fact that M is a *Riemannian* manifold plays a crucial role in the existence of geodesics: one needs to have a notion of length, which is provided by the 2-form $\mathcal{R}$. It is a standard result in differential geometry that in case of a Riemannian manifold M, one can construct around each $p \in M$ an open coordinate neighbourhood with coordinates that are related to the geodesics through p. More precisely, we have that: (a) we can associate with each geodesic through p a unique unit tangent vector in $T_p(M)$ and conversely with each unit tangent vector in $T_p(M)$ a unique geodesic through p; (b) we can construct an open neighbourhood of p such that for each q in it there exists a unique geodesic connecting p and q. Thus, one can assign coordinates to each q in the neighbourhood of p by specifying a tangent vector from $T_p(M)$: then its *direction* (i.e., after normalization to unit length) is associated with the geodesic connecting p and q and

its (Riemannian) *length* corresponds to the distance between p and q. Such coordinates are called *normal coordinates.*
Then the standard way of removing this second source of parameter dependence consists of prescribing that the step to be taken in the proposed search direction should be taken with respect to the normal coordinates around x_k. That is, one must take a step along the geodesic related to the search direction, of appropriate length as measured by the Riemannian metric. One can find this approach for instance in [11, 22, 23, 25]. (As a matter of fact, the approach using geodesics and normal coordinates admits also the construction of coordinate free versions of (quasi–)Newton and conjugate gradient methods. One then defines the Hessian of a function on a Riemannian manifold at a point p as the Hessian that occurs for normal coordinates.)
To conclude this section we remark that an exact implementation of a coordinate free version of any minimization algorithm on a Riemannian manifold is in general hard to obtain. This is due to the necessity of following geodesics, which can be characterized as solutions to a system of coupled nonlinear second–order differential equations, specified by the so–called Christoffel symbols. Therefore, in practice the computationally most efficient way of minimizing a nonlinear least squares criterion with the Riemannian version of the LM method as presented above, is probably to take steps, as usual, with respect to the current coordinates being used and to switch to a better conditioned coordinate chart when necessary. This asks for a coordinate chart selection strategy, which might very well be problem dependent. We shall confine ourselves to this approach in the application and experiments discussed in the companion paper [29].

4 Summary and Conclusions

In this paper we have studied the problem of minimizing a criterion function that is a possibly nonlinear sum of squares, defined on a Riemannian manifold M. In Sect. 2 we have discussed in detail an existing robust implementation (due to [27]) of the Levenberg–Marquardt algorithm (an extension of the well-known Gauss–Newton method) which is especially suited for solving such problems in the Euclidean case. We have pointed out some weak points and inaccuracies, in particular the heuristic scaling strategies and a sometimes incorrect calculation of a Gauss–Newton step in degenerate cases. These are analyzed and resolved in App. A and C.
In App. B we have proposed alternative ways of organizing the calculations involved in the LM algorithm such that only a limited amount of computer memory space for data storage is required. Theoretical results relating the behaviour (viz. the iteration paths) of these modified methods to that of the standard version are presented. These alternative calculations are particularly useful for large data fitting problems, where the number of datapoints m exceeds the number of parameters n by several orders of magnitude. Such applications are studied in the companion paper [29].
In Sect. 3 the LM algorithm has been modified to a so-called Riemannian version, acting on a Riemannian manifold in a coordinate independent way. This substantially enlarges its area of applicability and constitutes the main result of this paper. Moreover, the approach allows for a natural choice with respect to the shape of the trust-regions, thus by-passing the problem of finding adequate scaling strategies.

Appendix A : Subalgorithm A

In this appendix we consider "subalgorithm A" for the determination of an acceptable value for the LM parameter λ_k. The algorithm is based on an iterative scheme originally due to [16] but modified in [27]. The scheme involves a protection mechanism to prevent approximants to lie outside an interval containing the optimal value; the sequence of successive intervals is decreasing in length and we shall prove second order convergence of the approximants to the optimal value.
First of all we have to prove *right-continuity at zero* of function $\phi_k : \mathbf{R}_+ \to \mathbf{R}$, defined by $\phi_k(\alpha) = \|D_k p_k(\alpha)\| - \Delta_k$, with, for $\alpha > 0$, $p_k(\alpha)$ defined as the unique minimizing vector for criterion $c_\alpha(p)$, and, for $\alpha = 0$, $p_k(0)$ defined as the unique vector minimizing $\|D_k p\|^2$ among the set of vectors minimizing criterion $c_0(p)$. Here, for $\alpha \geq 0$, $c_\alpha(p)$ is defined as

$$c_\alpha(p) = \|J_k p + f_k\|^2 + \alpha\|D_k p\|^2 \tag{A.1}$$

It will be assumed that D_k is invertible, $\Delta_k > 0$ and that J_k is not identically zero (in which case the situation would be trivial).
Next, we shall prove that ϕ_k is a *strictly monotonically decreasing* function of α that converges to $-\Delta_k$ for α tending to infinity. Moreover, we will show that ϕ_k is *convex* on $\mathbf{R}_+$.
Then, in case $\phi_k(0) > 0$, this will imply that there exists a *unique* value $\alpha^* > 0$ for which $\phi_k(\alpha^*) = 0$ and we will show that subalgorithm A converges *quadratically* to it.

The proof of right-continuity of ϕ_k at zero is added here explicitly because of the fact that in [27] a GN step is calculated that leads to a value of $\phi_k(0)$ for which right-continuity does *not* necessarily occur. As a result, in case of a rank-deficient matrix J_k there generally will exist vectors f_k for which, in Moré's original scheme, subalgorithm A will be activated but will never terminate, because of the fact that ϕ_k does not have a zero. By the analysis presented here we show how to avoid such a situation.

To start with, consider the situation for $\alpha = 0$. Now $c_\alpha(p) = c_0(p) = \|J_k p + f_k\|^2$.
Apply QR-decomposition with column pivoting to J_k, as explained in Sect. 2 in the comments to step II of the algorithm, so that we obtain

$$J_k = Q_k^T \begin{pmatrix} T_k & S_k \\ O & O \end{pmatrix} \pi_k^T \tag{A.2}$$

where Q_k is an $m \times m$ orthogonal matrix, π_k is $n \times n$ permutation matrix (hence also orthogonal), S_k is an $r_k \times (n - r_k)$ matrix, T_k is an $r_k \times r_k$ nonsingular, upper triangular matrix and the zero matrix blocks are of appropriate size, with r_k denoting the rank of J_k. This can be achieved with Householder transformations in a numerically stable way.
Define $\begin{pmatrix} u_k \\ v_k \end{pmatrix} = Q_k f_k$, with u_k an r_k-vector and v_k an $(n - r_k)$-vector. Then

$$c_0(p) = \|J_k p + f_k\|^2 = \left\| Q_k^T \begin{pmatrix} T_k & S_k \\ O & O \end{pmatrix} \pi_k^T p + f_k \right\|^2 = \left\| \begin{pmatrix} T_k & S_k \\ O & O \end{pmatrix} \pi_k^T p + Q_k f_k \right\|^2 =$$

$$= \left\| \begin{pmatrix} T_k & S_k \\ O & O \end{pmatrix} \pi_k^T p + \begin{pmatrix} u_k \\ v_k \end{pmatrix} \right\|^2 = \| \begin{pmatrix} T_k & S_k \end{pmatrix} \pi_k^T p + u_k\|^2 + \|v_k\|^2 \tag{A.3}$$

Obviously, the last term $\|v_k\|^2$ remains uninfluenced by the choice of p. Choosing $p = \tilde{p}_k(0) = -\pi_k \begin{pmatrix} T_k^{-1} \\ O \end{pmatrix} u_k$, as calculated in [27], we find that $\begin{pmatrix} T_k & S_k \end{pmatrix} \pi_k^T \tilde{p}_k(0) + u_k = 0$. This shows that $\tilde{p}_k(0)$ is a minimizing vector for $c_0(p)$.
To find the set of all minimizing vectors p, we write $p = \tilde{p}_k(0) + s_k$ and consider the equation

$$\begin{pmatrix} T_k & S_k \end{pmatrix} \pi_k^T(\tilde{p}_k(0) + s_k) + u_k = 0 \tag{A.4}$$

This is equivalent to

$$\begin{pmatrix} T_k & S_k \end{pmatrix} \pi_k^T s_k = 0 \iff \begin{pmatrix} T_k & S_k \\ O & O \end{pmatrix} \pi_k^T s_k = 0 \iff$$

$$Q_k^T \begin{pmatrix} T_k & S_k \\ O & O \end{pmatrix} \pi_k^T s_k = 0 \iff J_k s_k = 0 \iff s_k \in \ker(J_k) \tag{A.5}$$

It shows that the set of all vectors p minimizing $c_0(p)$ is equal to the affine space $\{\tilde{p}_k(0) + s_k \,|\, s_k \in \ker(J_k)\}$.
As stated before, we define $p_k(0)$ to be the unique element in this set for which $||D_k p||^2$ is minimal. It can be calculated as follows. Define $h = \pi_k^T p$ and partition h as $\begin{pmatrix} h_1 \\ h_2 \end{pmatrix}$ with h_1 an r_k-vector and h_2 an $(n - r_k)$-vector. Then $\begin{pmatrix} T_k & S_k \end{pmatrix} \pi_k^T p + u_k = 0 \iff T_k h_1 + S_k h_2 + u_k = 0$. We can choose h_2 freely; it then determines h_1 according to

$$h_1 = -T_k^{-1}(S_k h_2 + u_k) \tag{A.6}$$

As a result we find that

$$||D_k p||^2 = \left\| D_k \pi_k \begin{pmatrix} -T_k^{-1} S_k \\ I \end{pmatrix} h_2 + D_k \tilde{p}_k(0) \right\|^2 \tag{A.7}$$

This is a *linear* least squares problem in which matrix $D_k \pi_k \begin{pmatrix} -T_k^{-1} S_k \\ I \end{pmatrix}$ has full column rank. Accordingly, it can be solved in a standard way, yielding a unique solution for h_2. From this minimizing vector h_2 we then can calculate vector $p_k(0) = \pi_k \begin{pmatrix} -T_k^{-1} & -T_k^{-1} S_k \\ O & I \end{pmatrix} \begin{pmatrix} u_k \\ h_2 \end{pmatrix}$.
Taking $p_k(0)$ to be an alternative particular solution to the problem of minimizing $c_0(p)$, we can rewrite the set of all minimizing vectors p as $\{p_k(0) + s_k \,|\, s_k \in \ker(J_k)\}$.

We now introduce some more notation. By q we denote vector $D_k p$ and accordingly we let $q_k(\alpha) = D_k p_k(\alpha)$ for all $\alpha \geq 0$. Also, we define $\tilde{J}_k = J_k D_k^{-1}$. Thus the problem of minimizing $c_\alpha(p)$ is equivalent to the problem of minimizing $\tilde{c}_\alpha(q)$ where

$$\tilde{c}_\alpha(q) = ||\tilde{J}_k q + f_k||^2 + \alpha ||q||^2 \tag{A.8}$$

Moreover, for $\alpha = 0$ the extra requirement that $||D_k p||^2$ be minimal among the set of minimizing solutions translates to the requirement that $||q||^2$ be minimal.
The set of minimizing solutions to $\tilde{c}_0(q)$ is given by

$$\{q_k(0) + t_k \,|\, t_k \in \ker(\tilde{J}_k)\} \tag{A.9}$$

Now, $q_k(0)$ can be viewed as the orthogonal projection of zero onto this affine space.

Lemma A.1 *Let $q \in \mathbf{R}^n$, $J \in \mathbf{R}^{m \times n}$. Then the following two statements are equivalent.*

1. $0 = \arg\min_{t \in \ker(J)} ||q + t||^2$.
2. $\forall t \in \ker(J)$: $q^T t = 0$.

Proof (1. $\Longrightarrow$ 2.) By contradiction.
Suppose $0 = \arg\min_{t \in \ker(J)} ||q + t||^2$, then for all $t \in \ker(J)$: $||q||^2 \leq ||q + t||^2$.
Suppose there exists $\hat{t} \in \ker(J)$ for which $q^T \hat{t} \neq 0$. Then obviously $\hat{t} \neq 0$. Consider $\bar{t}$ defined as $\bar{t} = -((q^T \hat{t})/||\hat{t}||^2)\hat{t}$. Since $\ker(J)$ is a linear space we have that $\bar{t} \in \ker(J)$.
Then $||q + \bar{t}||^2 = ||q||^2 - \frac{(q^T \hat{t})^2}{||\hat{t}||^2} < ||q||^2$. This is in contradiction with the minimality of $t = 0$ over $\ker(J)$.
(2. $\Longrightarrow$ 1.) Assume $\forall t \in \ker(J)$: $q^T t = 0$. Consider for $t \in \ker(J)$: $||q + t||^2 = ||q||^2 + 2q^T t + ||t||^2 = ||q||^2 + ||t||^2 \geq ||q||^2$, with equality if and only if $||t||^2 = 0$, i.e. if $t = 0$. This shows minimality at $t = 0$. □

Application of this lemma to the present situation, so with $q_k(0)$ assuming the role of q and $\tilde{J}_k$ the role of J, yields the fact that $q_k(0)$ is the unique point among the set of vectors q that minimize $\tilde{c}_0(q)$

for which $q^T t = 0, \forall t \in \ker(\tilde{J}_k)$. In other words, $q_k(0)$ is the single point of intersection of the affine space of vectors q minimizing $\tilde{c}_0(q)$ with the linear space $\ker(\tilde{J}_k)^\perp$, the orthogonal complement of $\ker(\tilde{J}_k)$, or equivalently, we have that the intersection of the linear space $\ker(\tilde{J}_k)^\perp$ with the set of points q in $\mathbf{R}^n$ for which $\tilde{c}_0(q) = \|u_k\|^2$ consists of the single point $q_k(0)$.

We next address the situation for $\alpha > 0$. We can rewrite $\tilde{c}_\alpha(q)$ as

$$\tilde{c}_\alpha(q) = \left\| \begin{pmatrix} \tilde{J}_k \\ \sqrt{\alpha} I \end{pmatrix} q + \begin{pmatrix} f_k \\ 0 \end{pmatrix} \right\|^2 \tag{A.10}$$

For $\alpha > 0$ matrix $\begin{pmatrix} \tilde{J}_k \\ \sqrt{\alpha} I \end{pmatrix}$ is nonsingular, so that $q_k(\alpha)$ yields a unique solution that can be expressed as

$$q_k(\alpha) = -(\tilde{J}_k^T \tilde{J}_k + \alpha I)^{-1} \tilde{J}_k^T f_k \tag{A.11}$$

These solutions have the special property that for all $\alpha > 0$ they are elements of the linear space $\ker(\tilde{J}_k)^\perp$. To prove this, notice the following lemma.

Lemma A.2 *For all matrices M and N and values of α for which the following expressions are well-defined we have that*

$$N(MN + \alpha I)^{-1} = (NM + \alpha I)^{-1} N \tag{A.12}$$

Proof Consider matrix $NMN + \alpha N$ and assume that both inverses in this lemma exist. Apply postmultiplication by $(MN + \alpha I)^{-1}$ and premultiplication by $(NM + \alpha I)^{-1}$. Notice that one can write $NMN + \alpha N$ both as $N(MN + \alpha I)$ and $(NM + \alpha I)N$. The lemma then readily follows. □

For $t \in \ker(\tilde{J}_k)$ consider the product $q_k(\alpha)^T t$. This can be written as

$$q_k(\alpha)^T t = -f_k^T \tilde{J}_k (\tilde{J}_k^T \tilde{J}_k + \alpha I)^{-1} t \tag{A.13}$$

Now, apply Lemma A.2 with $N = \tilde{J}_k$ and $M = \tilde{J}_k^T$. Then

$$q_k(\alpha)^T t = -f_k^T (\tilde{J}_k \tilde{J}_k^T + \alpha I)^{-1} \tilde{J}_k t = 0 \tag{A.14}$$

because, by assumption, $t \in \ker(\tilde{J}_k)$ so that $\tilde{J}_k t = 0$. This shows that indeed $q_k(\alpha) \in \ker(\tilde{J}_k)^\perp$.

Another property of the vectors $q_k(\alpha)$, $\alpha > 0$, is that they are bounded in length by $\|q_k(0)\|$. To prove this, notice that by definition $q_k(0)$ minimizes $\tilde{c}_0(q) = \|\tilde{J}_k q + f_k\|^2$ and $q_k(\alpha)$ minimizes $\tilde{c}_\alpha(q) = \|\tilde{J}_k q + f_k\|^2 + \alpha\|q\|^2$, so that

$$\|\tilde{J}_k q_k(0) + f_k\|^2 \leq \|\tilde{J}_k q_k(\alpha) + f_k\|^2 \tag{A.15}$$

whence

$$\|\tilde{J}_k q_k(0) + f_k\|^2 + \alpha\|q_k(\alpha)\|^2 \leq \|\tilde{J}_k q_k(\alpha) + f_k\|^2 + \alpha\|q_k(\alpha)\|^2 \leq \|\tilde{J}_k q_k(0) + f_k\|^2 + \alpha\|q_k(0)\|^2 \tag{A.16}$$

Therefore

$$\alpha\|q_k(\alpha)\|^2 \leq \alpha\|q_k(0)\|^2 \tag{A.17}$$

or equivalently

$$\|q_k(\alpha)\|^2 \leq \|q_k(0)\|^2 \tag{A.18}$$

proving the statement.
In a similar fashion we can prove the fact that $\lim_{\alpha\downarrow 0} \tilde{c}_0(q_k(\alpha)) = \tilde{c}_0(q_k(0))$. For this we must notice that

$$\begin{aligned} \tilde{c}_0(q_k(0)) = \|\tilde{J}_k q_k(0) + f_k\|^2 \leq \|\tilde{J}_k q_k(\alpha) + f_k\|^2 \leq \\ \leq \|\tilde{J}_k q_k(\alpha) + f_k\|^2 + \alpha\|q_k(\alpha)\|^2 \leq \|\tilde{J}_k q_k(0) + f_k\|^2 + \alpha\|q_k(0)\|^2 \end{aligned} \tag{A.19}$$

For $\alpha \downarrow 0$ we see that the last expression converges to the first one, $\tilde{c}_0(q_k(0))$, because $\alpha\|q_k(0)\|^2$ converges to 0. Therefore, also the second expression $\tilde{c}_0(q_k(\alpha)) = \|\tilde{J}_k q_k(\alpha) + f_k\|^2$ converges to $\tilde{c}_0(q_k(0))$. Summarizing, we have established the following facts.

1. $\tilde{c}_0 : \mathbf{R}^n \rightarrow \mathbf{R}$ is a continuous function, by its very definition.
2. $\ker(\tilde{J}_k)^\perp$ is a linear subspace of $\mathbf{R}^n$, also by definition.
3. For $\alpha > 0$ the corresponding points $q_k(\alpha)$ are elements of $\ker(\tilde{J}_k)^\perp$.
4. For $\alpha > 0$ the points $q_k(\alpha)$ are bounded in length, since we have proven $\|q_k(\alpha)\|^2 \leq \|q_k(0)\|^2$.
5. The limit $\lim_{\alpha \downarrow 0} \tilde{c}_0(q_k(\alpha))$ exists and is equal to $\tilde{c}_0(q_k(0))$.
6. The intersection of $\ker(\tilde{J}_k)^\perp$ with the set of points q in $\mathbf{R}^n$ for which $\tilde{c}_0(q) = \tilde{c}_0(q_k(0))$ consists of the single point $q_k(0)$.

Now consider the following lemma.

Lemma A.3 *Let $f : \mathbf{R}^n \rightarrow \mathbf{R}$ be continuous and $L \subseteq \mathbf{R}^n$ a linear subspace.*
Let $\{q_i\}_{i \in \mathbf{N}}$ be a sequence of points in L which are bounded in length, and for which $\lim_{i \rightarrow \infty} f(q_i)$ exists and is equal to, say, $d \in \mathbf{R}$.
Suppose that the intersection of L with the set $\{q \in \mathbf{R}^n \mid f(q) = d\}$ consists of a single point q^*.
Then $\lim_{i \rightarrow \infty} q_i$ exists and is equal to q^*.

Proof By assumption we have that $\{q_i\}_{i \in \mathbf{N}}$ is bounded, so there exists $M \in \mathbf{R}^+$ such that for all $i \in \mathbf{N}$: $\|q_i - q^*\| \leq M$.
Define $K = L \cap \{q \in \mathbf{R}^n \mid \|q - q^*\| \leq M\}$.
Notice that any linear subspace of $\mathbf{R}^n$ is closed so that L is closed. Notice also that the set $\{q \in \mathbf{R}^n \mid \|q - q^*\| \leq M\}$ is closed and bounded in $\mathbf{R}^n$ and therefore compact. Thus, K is also compact.
Define $g : \mathbf{R}^n \rightarrow \mathbf{R}$ by $g(q) = |f(q) - d|^2$. Because f is continuous, g is continuous too. Moreover, q^* is the unique point in K for which $g(q^*) = 0$. For all other points q in K we have $g(q) > 0$.
Let $\delta > 0$ be given. Define $N_\delta = K \cap \{q \in \mathbf{R}^n \mid \|q - q^*\| < \delta\}$.
Then N_δ is open in K, so that its complement with respect to K, denoted by $K - N_\delta$, is closed and therefore compact. So, if $K - N_\delta$ is nonempty, there exists $g^\delta_{\min} = \min_{K - N_\delta} g(q) > 0$, which is actually assumed for some element of $K - N_\delta$. The fact that $g^\delta_{\min} > 0$ follows from the observation that q^* is the unique point in K with $g(q^*) = 0$ and that for all other points q in K we have $g(q) > 0$, but $q^* \notin K - N_\delta$ since $q^* \in N_\delta$ by construction.
By assumption we have that $\lim_{i \rightarrow \infty} f(q_i) = d$ exists, so that $\lim_{i \rightarrow \infty} g(q_i) = 0$. Thus, $\exists I_\delta \in \mathbf{N}$ such that $\forall i \geq I_\delta$: $g(q_i) < g^\delta_{\min}$. As a consequence, $\forall i \geq I_\delta$: $q_i \in N_\delta$.
By letting δ decrease to zero we obtain that $\lim_{i \rightarrow \infty} q_i = q^*$. □

We can apply this lemma to the situation at hand. Indeed, the six items above that summarize the results found so far make clear that the assumptions for Lemma A.3 are fulfilled if we identify L with $\ker(\tilde{J}_k)^\perp$, f with $\tilde{c}_0$, d with $\tilde{c}_0(q_k(0))$, q^* with $q_k(0)$ and if we take $q_i = q_k(\alpha_i)$ where $\{\alpha_i\}_{i \in \mathbf{N}}$ is a sequence of positive numbers with $\lim_{i \rightarrow \infty} \alpha_i = 0$.
Thus, we find that $\lim_{\alpha \downarrow 0} q_k(\alpha) = q_k(0)$. Because of nonsingularity of D_k this implies that also $\lim_{\alpha \downarrow 0} p_k(\alpha) = p_k(0)$. This means that we have proven the following proposition.

Proposition A.4 *For $\alpha > 0$, let $p_k(\alpha)$ be defined as the unique vector in $\mathbf{R}^n$ minimizing $c_\alpha(p) = \|J_k p + f_k\|^2 + \alpha \|D_k p\|^2$, where D_k is nonsingular.*
For $\alpha = 0$ define $p_k(0)$ as the unique vector in the set of vectors minimizing $c_0(p) = \|J_k p + f_k\|^2$ that minimizes $\|D_k p\|^2$.
Then $\lim_{\alpha \downarrow 0} p_k(\alpha) = p_k(0)$.

As a corollary we find that all vectors $p \in p_k(0) + \ker(J_k)$ unequal to $p_k(0)$ will have $\|D_k p\| > \|D_k p_k(0)\| = \lim_{\alpha \downarrow 0} \|D_k p_k(\alpha)\|$, so that in particular the use of $\tilde{p}_k(0)$ for a GN step in case of a rank-deficient J_k (as in [27]) will in general lead to the loss of right-continuity at zero of $\phi_k(\alpha)$. Right-continuity at zero of ϕ_k is obtained only when using $p_k(0)$, as constructed above. Clearly, when following Moré's description one might encounter a situation where $\|D_k \tilde{p}_k(0)\|$ is too large (i.e., leads us outside the trust-region), but where $\|D_k p_k(0)\|$ is small enough (i.e., still inside the trust–region). In such a case there will not exist a value α^* for which $\phi_k(\alpha^*) = 0$, because $\|D_k p_k(\alpha)\| = \|q_k(\alpha)\| \leq \|q_k(0)\|$ for all $\alpha > 0$. In such a situation subalgorithm A as described in [27] breaks down. The use of $p_k(0)$ instead of $\tilde{p}_k(0)$ prevents this. (Of course it requires an extra amount of computation, but it is good to remember that this is only necessary in a situation where $r_k < n$.)

Next we shall prove some further properties of $\phi_k(\cdot)$, in case $p_k(0)$ is used as described above.

Proposition A.5 *Suppose $\phi_k : \mathbf{R}_+ \to \mathbf{R}$ is defined by $\phi_k(\alpha) = \|q_k(\alpha)\| - \Delta_k$ (where $\Delta_k > 0$), with $q_k(\alpha)$ as described before. Suppose that $q_k(0) \neq 0$.*
Then ϕ_k is strictly monotonically decreasing and convex, with $\lim_{\alpha\to\infty} \phi_k(\alpha) = -\Delta_k$.

Proof

a. The assumption $q_k(0) \neq 0$ is equivalent to $q_k(\alpha) \neq 0$ for each value of $\alpha > 0$, and also to $\tilde{J}_k^T f_k \neq 0$. This follows from the fact that $q_k(\alpha) = 0$ for *some* $\alpha > 0$ implies that $-(\tilde{J}_k^T \tilde{J}_k + \alpha I)^{-1} \tilde{J}_k^T f_k = 0$ so that $\tilde{J}_k^T f_k = 0$. Therefore, $q_k(\alpha) = 0$ for *any* $\alpha > 0$. But we already have shown that $\lim_{\alpha\downarrow 0} q_k(\alpha) = q_k(0)$. Thus, also $q_k(0)$ would be zero, which contradicts our assumption. The fact that $\|q_k(\alpha)\| \leq \|q_k(0)\|$ provides us with the converse implication.
Thus, for use in the rest of this proof we can assume that $q_k(\alpha) \neq 0$ and $\|q_k(\alpha)\| \neq 0$.

b. We shall prove that $\lim_{\alpha\to\infty} \phi_k(\alpha) = -\Delta_k$.
Denote the eigenvalues of matrix $\tilde{J}_k^T \tilde{J}_k$ by μ_i, $(i = 1, \ldots, n)$. Since $\tilde{J}_k^T \tilde{J}_k$ is positive semi-definite by construction we have that all eigenvalues are real and nonnegative. Denote the smallest eigenvalue by $\mu_{\min}$ and the largest by $\mu_{\max}$. Notice that for $\alpha > 0$ the eigenvalues of $(\tilde{J}_k^T \tilde{J}_k + \alpha I)^{-1}$ are given by $\frac{1}{\mu_i + \alpha}$, $(i = 1, \ldots, n)$.
According to Rayleigh's principle we have for all vectors $u \in \mathbf{R}^n$ that

$$\frac{1}{\mu_{\max} + \alpha}\|u\| \leq \|(\tilde{J}_k^T \tilde{J}_k + \alpha I)^{-1} u\| \leq \frac{1}{\mu_{\min} + \alpha}\|u\| \tag{A.20}$$

Taking $u = -\tilde{J}_k^T f_k$ we find

$$\frac{1}{\mu_{\max} + \alpha}\|\tilde{J}_k^T f_k\| \leq \|q_k(\alpha)\| \leq \frac{1}{\mu_{\min} + \alpha}\|\tilde{J}_k^T f_k\| \tag{A.21}$$

For $\alpha \to \infty$ both bounds tend to zero, so that $\lim_{\alpha\to\infty} \|q_k(\alpha)\| = 0$. Thus, $\lim_{\alpha\to\infty} \phi_k(\alpha) = -\Delta_k$.

c. For $\alpha > 0$, notice that ϕ_k is infinitely often continuously differentiable. We will now calculate the derivative of ϕ_k at α, denoted by $\phi_k'(\alpha)$.

$$\phi_k'(\alpha) = \tfrac{d}{d\alpha}\left\{ \left[q_k(\alpha)^T q_k(\alpha)\right]^{\frac{1}{2}} \right\} = \tfrac{d}{d\alpha}\left\{ \left[f_k^T \tilde{J}_k (\tilde{J}_k^T \tilde{J}_k + \alpha I)^{-2} \tilde{J}_k^T f_k\right]^{\frac{1}{2}} \right\} =$$
$$= -\frac{f_k^T \tilde{J}_k (\tilde{J}_k^T \tilde{J}_k + \alpha I)^{-3} \tilde{J}_k^T f_k}{\|q_k(\alpha)\|} < 0 \tag{A.22}$$

Here we make use of the fact that, for all $n \in \mathbf{Z}$ and for all matrices A of size $n \times n$, the following relation holds at points where the derivative exists:

$$\frac{d}{d\alpha}\left\{(A + \alpha I)^n\right\} = n(A + \alpha I)^{n-1} \tag{A.23}$$

We also use the fact that $(\tilde{J}_k^T \tilde{J}_k + \alpha I)^{-3}$ is positive definite by construction. Thus, the numerator of the fraction above would be zero if and only if $\tilde{J}_k^T f_k = 0$. But in part a. we have shown that $\tilde{J}_k^T f_k = 0$ cannot occur. There we also found that the denominator is unequal to zero.

d. We will now show that the second derivative function of ϕ_k, denoted by ϕ_k'', is strictly positive. This then establishes convexity of ϕ_k.
Indeed, straightforward differentiation of the expression already obtained for $\phi_k'(\alpha)$, while noticing that the denominator in that expression equals $\phi_k(\alpha) + \Delta_k$, yields

$$\phi_k''(\alpha) = \frac{d}{d\alpha}\{\phi_k'(\alpha)\} = \frac{1}{\|q_k(\alpha)\|}\left[3 f_k^T \tilde{J}_k (\tilde{J}_k^T \tilde{J}_k + \alpha I)^{-4} \tilde{J}_k^T f_k - (\phi_k'(\alpha))^2\right] \tag{A.24}$$

Multiplication by the positive factor $\|q_k(\alpha)\|^3$ yields the expression

$$\begin{aligned}&3[f_k^T \tilde{J}_k (\tilde{J}_k^T \tilde{J}_k + \alpha I)^{-4} \tilde{J}_k^T f_k][f_k^T \tilde{J}_k (\tilde{J}_k^T \tilde{J}_k + \alpha I)^{-2} \tilde{J}_k^T f_k] + \\ &-[f_k^T \tilde{J}_k (\tilde{J}_k^T \tilde{J}_k + \alpha I)^{-3} \tilde{J}_k^T f_k][f_k^T \tilde{J}_k (\tilde{J}_k^T \tilde{J}_k + \alpha I)^{-3} \tilde{J}_k^T f_k]\end{aligned} \tag{A.25}$$

Using $q_k(\alpha) = -(\tilde{J}_k^T \tilde{J}_k + \alpha I)^{-1} \tilde{J}_k^T f_k$ we can write this as:

$$\begin{aligned}&3[q_k(\alpha)^T (\tilde{J}_k^T \tilde{J}_k + \alpha I)^{-2} q_k(\alpha)][q_k(\alpha)^T q_k(\alpha)] + \\ &-[q_k(\alpha)^T (\tilde{J}_k^T \tilde{J}_k + \alpha I)^{-1} q_k(\alpha)][q_k(\alpha)^T (\tilde{J}_k^T \tilde{J}_k + \alpha I)^{-1} q_k(\alpha)]\end{aligned} \tag{A.26}$$

Notice that the Cauchy-Schwarz inequality gives us for any pair of vectors (a, b) that $a^T b \leq \sqrt{a^T a\, b^T b}$, so that $a^T a\, b^T b - a^T b\, a^T b \geq 0$, whence $3\, a^T a\, b^T b - a^T b\, a^T b > 0$ in case both a and b are nonzero. Using $a = (\tilde{J}_k^T \tilde{J}_k + \alpha I)^{-1} q_k(\alpha)$ and $b = q_k(\alpha)$ we obtain $\phi_k''(\alpha) > 0$. Observe that according to part a. both a and b are indeed nonzero. This completes the proof of the proposition. □

Remark 1. From the discussion in part a. of the proof we see that $q_k(0) = 0$ if and only if $\tilde{J}_k^T f_k = 0$, i.e. if we are in a stationary point of Φ. Notice that subalgorithm A is only invoked for $\phi_k(0) > 0$, which implies $\|q_k(0)\| > \Delta_k > 0$. Therefore, we will only meet situations where ϕ_k is indeed strictly monotonically decreasing and strictly convex on $\mathbf{R}_+$.

Remark 2. The numerator of the expression obtained for $\phi_k'(\alpha)$ can be rewritten as

$$q_k(\alpha)^T (\tilde{J}_k^T \tilde{J}_k + \alpha I)^{-1} q_k(\alpha) \tag{A.27}$$

Here we can write $(\tilde{J}_k^T \tilde{J}_k + \alpha I)^{-1}$ as $D_k (J_k^T J_k + \alpha D_k^T D_k)^{-1} D_k^T$. Now $(J_k^T J_k + \alpha D_k^T D_k)$ can be seen to be equivalent to $\pi_k R_k(\alpha)^T R_k(\alpha) \pi_k^T$ in the light of the decompositions obtained in step II and step III of the algorithm (see Sect. 2). This shows that we can rewrite the expression for the numerator of $\phi_k'(\alpha)$ as

$$\begin{aligned}&q_k(\alpha)^T D_k \pi_k (R_k(\alpha)^T R_k(\alpha))^{-1} \pi_k^T D_k^T q_k(\alpha) = \\ &= q_k(\alpha)^T D_k \pi_k R_k(\alpha)^{-1} R_k(\alpha)^{-T} \pi_k^T D_k^T q_k(\alpha) = \\ &= \|R_k(\alpha)^{-T} \pi_k^T D_k^T q_k(\alpha)\|^2\end{aligned} \tag{A.28}$$

This shows the correctness of the formula presented in Sect. 2 for the calculation of $\phi_k'(\alpha)$.

As a corollary of Prop. A.5 we have that in case $\phi_k(0) > 0$ there exists a unique $\alpha^* > 0$ for which $\phi_k(\alpha^*) = 0$. Convexity of ϕ_k implies that the Newton-Raphson method applied to ϕ_k will always yield iterates giving a lower bound for α^*. That is, for all $\alpha \geq 0$ we have that

$$\alpha - \left(\frac{\phi_k(\alpha)}{\phi_k'(\alpha)}\right) < \alpha^* \tag{A.29}$$

This explains the updating rule for the lower bound for α^*.
The updating rule for the upper bound of α^* is clear: if $\phi_k(\alpha) < 0$ then we must have $\alpha > \alpha^*$. Thus,

the only thing to be proven for this bound is the correctness of the rule by which it is initialized, namely

$$ub_0 = \frac{\|\tilde{J}_k^T f_k\|}{\Delta_k} \tag{A.30}$$

Considering step b. in the proof above, we see that for all $\alpha > 0$

$$\|q_k(\alpha)\| \leq \frac{1}{\mu_{\min} + \alpha}\|\tilde{J}_k^T f_k\| \leq \frac{1}{\alpha}\|\tilde{J}_k^T f_k\| \tag{A.31}$$

Recalling that $\phi_k(\alpha) = \|q_k(\alpha)\| - \Delta_k$, we get for $\alpha = ub_0$ that $\phi_k(\alpha) \leq 0$, so that ub_0 provides an upper bound on α^*.
In fact, this upper bound will be strict, provided $q_k(0) \neq 0$, that is $\tilde{J}_k^T f_k \neq 0$ (see part a.). Indeed, the inequality mentioned above will become an equality if and only if $\tilde{J}_k^T f_k$ is a vector in the eigenspace of $(\tilde{J}_k^T \tilde{J}_k + \alpha I)^{-1}$ corresponding to eigenvalue $\frac{1}{\mu_{\min}+\alpha}$, with $\mu_{\min} = 0$. Therefore we have that

$$(\tilde{J}_k^T \tilde{J}_k + \alpha I)^{-1} \tilde{J}_k^T f_k = \frac{1}{\alpha}\tilde{J}_k^T f_k \tag{A.32}$$

so that upon premultiplication by $(\tilde{J}_k^T \tilde{J}_k + \alpha I)$ it follows that

$$\tilde{J}_k^T f_k = \frac{1}{\alpha}(\tilde{J}_k^T \tilde{J}_k + \alpha I)\tilde{J}_k^T f_k = \frac{1}{\alpha}\tilde{J}_k^T \tilde{J}_k \tilde{J}_k^T f_k + \tilde{J}_k^T f_k \tag{A.33}$$

Hence $\tilde{J}_k^T \tilde{J}_k \tilde{J}_k^T f_k = 0$, from which we deduce by premultiplication by $f_k^T \tilde{J}_k$ that $\|\tilde{J}_k \tilde{J}_k^T f_k\|^2 = 0$, whence $\tilde{J}_k \tilde{J}_k^T f_k = 0$. Now premultiplication by f_k^T does the job. It yields $\|\tilde{J}_k^T f_k\|^2 = 0$, whence $\tilde{J}_k^T f_k = 0$. This shows that equality will occur if and only if $q_k(0) = 0$, which is excluded by assumption.

We shall now consider convergence of subalgorithm A and point out some relationships to the Newton–Raphson procedure for finding a zero of a convex, monotonically decreasing function.
The updating scheme for α_ℓ in subalgorithm A is given by

$$\alpha_{\ell+1} = \alpha_\ell - \frac{\phi_k(\alpha_\ell) + \Delta_k}{\Delta_k}\left(\frac{\phi_k(\alpha_\ell)}{\phi_k'(\alpha_\ell)}\right) \tag{A.34}$$

This scheme can be motivated as follows. The idea is to approximate $\phi_k(\alpha)$ locally around α_ℓ by some function $\tilde{\phi}_k(\alpha)$ of the form

$$\tilde{\phi}_k(\alpha) = \frac{c_k}{d_k + \alpha} - \Delta_k, \tag{A.35}$$

where c_k and d_k are constants that are chosen such that

$$\tilde{\phi}_k(\alpha_\ell) = \phi_k(\alpha_\ell), \qquad \tilde{\phi}_k'(\alpha_\ell) = \phi_k'(\alpha_\ell) \tag{A.36}$$

Notice that, just like ϕ_k, function $\tilde{\phi}_k$ is monotonically decreasing and convex for $\alpha > -d_k$, provided $c_k > 0$. Moreover, $\lim_{\alpha\to\infty} \tilde{\phi}_k(\alpha) = -\Delta_k = \lim_{\alpha\to\infty} \phi_k(\alpha)$. In the Newton–Raphson procedure the idea is to approximate $\phi_k(\alpha)$ locally around α_ℓ by the line tangent to ϕ_k at α_ℓ. In view of the properties of $\tilde{\phi}_k$ mentioned above, we expect $\tilde{\phi}_k$ to yield a better approximation to ϕ_k than is provided by the tangent line.
Some easy calculations show that c_k and d_k are given by

$$c_k = -\frac{(\phi_k(\alpha_\ell) + \Delta_k)^2}{\phi_k'(\alpha_\ell)}, \qquad d_k = -\alpha_\ell - \frac{\phi_k(\alpha_\ell) + \Delta_k}{\phi_k'(\alpha_\ell)} \tag{A.37}$$

We see that indeed $c_k > 0$.
The zero of $\tilde{\phi}_k$ is given by $\alpha_{\ell+1} = \frac{c_k}{\Delta_k} - d_k$. This precisely leads to the updating formula for α_ℓ given above.

Local quadratic convergence to α^* of this scheme is easily proven by Taylor series expansions of ϕ_k and ϕ_k' around $\alpha_\ell = \alpha^* + h$, with $h \ll 1$. We have

$$\phi_k(\alpha_\ell) = \phi_k(\alpha^*) + h\phi_k'(\alpha^*) + \frac{1}{2}h^2\phi_k''(\alpha^*) + O(h^3) = h\phi_k'(\alpha^*) + \frac{1}{2}h^2\phi_k''(\alpha^*) + O(h^3) \qquad (A.38)$$

because $\phi_k(\alpha^*) = 0$, and

$$\phi_k'(\alpha_\ell) = \phi_k'(\alpha^*) + h\phi_k'(\alpha^*) + O(h^2) \qquad (A.39)$$

Thus,

$$\frac{\phi_k(\alpha_\ell)+\Delta_k}{\Delta_k} = 1 + h\frac{\phi_k'(\alpha^*)}{\Delta_k} + \frac{1}{2}h^2\frac{\phi_k''(\alpha^*)}{\Delta_k} + O(h^3) \qquad (A.40)$$

$$\frac{\phi_k(\alpha_\ell)}{\phi_k'(\alpha_\ell)} = h - \frac{1}{2}h^2\frac{\phi_k''(\alpha^*)}{\phi_k'(\alpha^*)} + O(h^3) \qquad (A.41)$$

This shows that

$$\begin{aligned}\alpha_{\ell+1} &= \alpha_\ell - h + \tfrac{1}{2}h^2\frac{\phi_k''(\alpha^*)}{\phi_k'(\alpha^*)} - h^2\frac{\phi_k'(\alpha^*)}{\Delta_k} + O(h^3) = \\ &= \alpha^* + \tfrac{1}{2}h^2\left[\frac{\phi_k''(\alpha^*)}{\phi_k'(\alpha^*)} - 2\frac{\phi_k'(\alpha^*)}{\Delta_k}\right] + O(h^3)\end{aligned} \qquad (A.42)$$

Therefore, local quadratic convergence occurs.
We conclude this appendix by presenting a rationale for the approximation of $\phi_k(\alpha)$ by a function $\phi_k'(\alpha)$ of the particular form above, based on an expression for $\phi_k(\alpha)$ in terms of the singular values of $\tilde{J}_k$. The derivation is slightly different from that in [27], but based on the same ideas. We shall also prove that Hebden's algorithm has indeed faster local convergence properties than the standard Newton–Raphson method.
Consider $q_k(\alpha) = -(\tilde{J}_k^T\tilde{J}_k + \alpha I)^{-1}\tilde{J}_k^T f_k$. In view of Lemma A.2 we can write this alternatively as $-\tilde{J}_k^T(\tilde{J}_k\tilde{J}_k^T + \alpha I)^{-1}f_k$. Therefore

$$\|q_k(\alpha)\|^2 = f_k^T(\tilde{J}_k\tilde{J}_k^T + \alpha I)^{-1}\tilde{J}_k\tilde{J}_k^T(\tilde{J}_k\tilde{J}_k^T + \alpha I)^{-1}f_k \qquad (A.43)$$

Notice that $\tilde{J}_k\tilde{J}_k^T$ is a real, square, symmetric matrix that is positive semi-definite and therefore it can be decomposed (via a singular value decomposition, cf., e.g., [33, p.142,293]) as

$$\tilde{J}_k\tilde{J}_k^T = P_k\Lambda_k P_k^T \qquad (A.44)$$

where P_k is orthogonal and Λ_k is diagonal, containing the eigenvalues of $\tilde{J}_k\tilde{J}_k^T$, which are all real nonnegative, in decreasing order. Let $\Sigma_k = \text{diag}(\sigma_k^1, \ldots, \sigma_k^m)$ with $\sigma_k^i \geq 0$ such that $\Lambda_k = \Sigma_k^2$. Then σ_k^i denotes the ith singular value of $\tilde{J}_k$. (For $i > \min(n,m)$ we have $\sigma_k^i = 0$.) Also, define vector g_k, with components g_k^i, as $g_k = P_k^T f_k$. Then

$$\begin{aligned}\|q_k(\alpha)\|^2 &= f_k^T(P_k\Lambda_k P_k^T + \alpha I)^{-1}P_k\Lambda_k P_k^T(P_k\Lambda_k P_k^T + \alpha I)^{-1}f_k = \\ &= f_k^T P_k(\Lambda_k + \alpha I)^{-1}P_k^T P_k\Lambda_k P_k^T P_k(\Lambda_k + \alpha I)^{-1}P_k^T f_k = \\ &= g_k^T(\Lambda_k + \alpha I)^{-1}\Lambda_k(\Lambda_k + \alpha I)^{-1}g_k = \\ &= \textstyle\sum_{i=1}^m \frac{(\sigma_k^i)^2(g_k^i)^2}{((\sigma_k^i)^2+\alpha)^2}\end{aligned} \qquad (A.45)$$

This shows that

$$\phi_k(\alpha) = \left[\sum_{i=1}^m \frac{(\sigma_k^i)^2(g_k^i)^2}{((\sigma_k^i)^2+\alpha)^2}\right]^{1/2} - \Delta_k \qquad (A.46)$$

providing the rationale for the choice of $\tilde{\phi}_k(\alpha)$.

When comparing to the Newton–Raphson procedure, given by the alternative updating formula

$$\tilde{\alpha}_{\ell+1} = \tilde{\alpha}_\ell - \left(\frac{\phi_k(\tilde{\alpha}_\ell)}{\phi_k'(\tilde{\alpha}_\ell)}\right) \qquad (A.47)$$

we see that in subalgorithm A a factor $\frac{\phi_k(\alpha_\ell)+\Delta_k}{\Delta_k} = 1 + O(h)$ is added, which does not affect the quadratic convergence property. Actually, one should notice that in the Newton–Raphson scheme, as a result of the linear approximation to a convex function, the resulting approximant $\tilde{\alpha}_{\ell+1}$ always provides a lower bound for α^*. However, we see that $\alpha_{\ell+1} > \tilde{\alpha}_{\ell+1}$ when starting from the same initial approximant $\alpha_\ell = \tilde{\alpha}_\ell$. This shows that the approximant $\alpha_{\ell+1}$ for α^* will be larger than the lower bound resulting from the Newton–Raphson procedure and therefore closer to α^* in case $\alpha_{\ell+1} \leq \alpha^*$, whereas in the opposite case ($\alpha_{\ell+1} > \alpha^*$) the upper bound on α^* is likely to be adapted to a value that is close to α^*.
In fact, for small h, we can prove that $\alpha_{\ell+1}$ resulting from Hebden's algorithm will lie in between $\tilde{\alpha}_{\ell+1}$ (resulting from the Newton–Raphson scheme) and α^*. This then proves that subalgorithm A is more efficient than Newton–Raphson.
To show that this statement holds true, we have to reconsider the Taylor series expansions of $\phi_k(\alpha)$ and $\phi_k'(\alpha)$ around $\alpha_\ell = \alpha^* + h$, and compare the second–order terms for both schemes. One easily finds that in the Newton–Raphson case

$$\tilde{\alpha}_{\ell+1} = \alpha^* + \frac{1}{2}h^2 \frac{\phi_k''(\alpha^*)}{\phi_k'(\alpha^*)} + O(h^3) \tag{A.48}$$

whereas for Hebden's algorithm we already had that

$$\alpha_{\ell+1} = \alpha^* + \frac{1}{2}h^2 \left[\frac{\phi_k''(\alpha^*)}{\phi_k'(\alpha^*)} - 2\frac{\phi_k'(\alpha^*)}{\Delta_k}\right] + O(h^3) \tag{A.49}$$

Since $\phi_k'(\alpha^*) < 0$ and $\phi_k''(\alpha^*) > 0$ it is clear that the second–order term in the Newton–Raphson case is negative, and because $\Delta_k > 0$ it is also clear that the second–order term in Hebden's scheme is larger than the second–order Newton–Raphson term. To get more precise results, it is easiest to start by considering the formula for $\phi_k(\alpha)$ in terms of the singular values of $\tilde{J}_k$.
From the fact that α^* constitutes a zero for $\phi_k(\alpha)$, we have that

$$\Delta_k = \left[\sum_{i=1}^{m} \frac{(\sigma_k^i)^2(g_k^i)^2}{((\sigma_k^i)^2+\alpha)^2}\right]^{1/2} \tag{A.50}$$

Therefore, we can express the second order term in the Hebden scheme as

$$\begin{aligned}\tfrac{1}{2}h^2 \left[\tfrac{\phi_k''(\alpha^*)}{\phi_k'(\alpha^*)} - 2\tfrac{\phi_k'(\alpha^*)}{\Delta_k}\right] = -\tfrac{3}{2}h^2 \Bigg[&\left(\textstyle\sum_{i=1}^m \tfrac{(\sigma_k^i)^2(g_k^i)^2}{((\sigma_k^i)^2+\alpha^*)^4}\right)\left(\textstyle\sum_{i=1}^m \tfrac{(\sigma_k^i)^2(g_k^i)^2}{((\sigma_k^i)^2+\alpha^*)^3}\right)^{-1} + \\ &- \left(\textstyle\sum_{i=1}^m \tfrac{(\sigma_k^i)^2(g_k^i)^2}{((\sigma_k^i)^2+\alpha^*)^3}\right)\left(\textstyle\sum_{i=1}^m \tfrac{(\sigma_k^i)^2(g_k^i)^2}{((\sigma_k^i)^2+\alpha^*)^2}\right)^{-1}\Bigg]\end{aligned} \tag{A.51}$$

This follows from the fact that

$$\begin{aligned}\tfrac{\phi_k''(\alpha^*)}{\phi_k'(\alpha^*)} = &-3\left(\textstyle\sum_{i=1}^m \tfrac{(\sigma_k^i)^2(g_k^i)^2}{((\sigma_k^i)^2+\alpha^*)^4}\right)\left(\textstyle\sum_{i=1}^m \tfrac{(\sigma_k^i)^2(g_k^i)^2}{((\sigma_k^i)^2+\alpha^*)^3}\right)^{-1} + \\ &+ \left(\textstyle\sum_{i=1}^m \tfrac{(\sigma_k^i)^2(g_k^i)^2}{((\sigma_k^i)^2+\alpha^*)^3}\right)\left(\textstyle\sum_{i=1}^m \tfrac{(\sigma_k^i)^2(g_k^i)^2}{((\sigma_k^i)^2+\alpha^*)^2}\right)^{-1}\end{aligned} \tag{A.52}$$

and

$$\frac{\phi_k'(\alpha^*)}{\Delta_k} = -\left(\sum_{i=1}^{m} \frac{(\sigma_k^i)^2(g_k^i)^2}{((\sigma_k^i)^2+\alpha^*)^3}\right)\left(\sum_{i=1}^{m} \frac{(\sigma_k^i)^2(g_k^i)^2}{((\sigma_k^i)^2+\alpha^*)^2}\right)^{-1} \tag{A.53}$$

because

$$\phi_k'(\alpha^*) = -\left(\sum_{i=1}^{m} \frac{(\sigma_k^i)^2(g_k^i)^2}{((\sigma_k^i)^2+\alpha^*)^3}\right)\left(\sum_{i=1}^{m} \frac{(\sigma_k^i)^2(g_k^i)^2}{((\sigma_k^i)^2+\alpha^*)^2}\right)^{-1/2} \tag{A.54}$$

and

$$\begin{aligned}\phi_k''(\alpha^*) = 3&\left(\textstyle\sum_{i=1}^m \tfrac{(\sigma_k^i)^2(g_k^i)^2}{((\sigma_k^i)^2+\alpha^*)^4}\right)\left(\textstyle\sum_{i=1}^m \tfrac{(\sigma_k^i)^2(g_k^i)^2}{((\sigma_k^i)^2+\alpha^*)^2}\right)^{-1/2} + \\ &- \left(\textstyle\sum_{i=1}^m \tfrac{(\sigma_k^i)^2(g_k^i)^2}{((\sigma_k^i)^2+\alpha^*)^3}\right)^2\left(\textstyle\sum_{i=1}^m \tfrac{(\sigma_k^i)^2(g_k^i)^2}{((\sigma_k^i)^2+\alpha^*)^2}\right)^{-3/2}\end{aligned} \tag{A.55}$$

Therefore, by taking vectors a and b as

$$a = \begin{pmatrix} \frac{\sigma_k^1 g_k^1}{((\sigma_k^1)^2+\alpha^*)^2} \\ \vdots \\ \frac{\sigma_k^m g_k^m}{((\sigma_k^m)^2+\alpha^*)^2} \end{pmatrix} \qquad b = \begin{pmatrix} \frac{\sigma_k^1 g_k^1}{((\sigma_k^1)^2+\alpha^*)} \\ \vdots \\ \frac{\sigma_k^m g_k^m}{((\sigma_k^m)^2+\alpha^*)} \end{pmatrix} \tag{A.56}$$

application of the Cauchy-Schwarz inequality $a^T a\, b^T b - (a^T b)^2 \geq 0$ immediately gives that

$$\left[\frac{\phi_k''(\alpha^*)}{\phi_k'(\alpha^*)} - 2\frac{\phi_k'(\alpha^*)}{\Delta_k}\right] \leq 0 \tag{A.57}$$

(Notice that this constitutes an alternative proof of the fact that $\phi_k''(\alpha) > 0$ for all $\alpha > 0$.) Equality occurs if and only if vector a is a multiple of b, that is, if and only if for all indices i for which $\sigma_k^i g_k^i$ is nonzero, the singular value σ_k^i is the same. This automatically includes the scalar case ($m = 1$), which should be, because ϕ_k then indeed has the structure assumed for $\tilde{\phi}_k$, so that a perfect fit occurs.
We now have shown that, for h small enough, the approximate resulting from the Hebden scheme will be in between the Newton–Raphson approximate and the optimal value α^*. It follows that, in our application, subalgorithm A is more efficient than Newton–Raphson.

Appendix B : Calculations with reduced memory storage capacity

In [29] we discuss an application of the Riemannian LM algorithm where in general the number m of residuals is very large compared to the number n of parameters. This can cause serious problems on the level of required computer memory space for data storage. Indeed, for the experiments in [29] where $n = 16$ and $m = 4000$ we need to store vector f_k, of size 4000, and matrix J_k, of size 4000 by 16. (Notice that in that example $p = 2$, i.e., there are only 2 outputs associated with every datapoint; it is the amount of available observations that makes the size of the matrices and vectors so large, and not the complexity of the model to be estimated.) Moreover, we need to apply QR–decomposition to matrix J_k, leading to an orthogonal matrix Q_k of size 4000 by 4000. Fortunately, it turns out to be unnecessary to calculate this extremely large matrix explicitly, because what is needed really is vector $Q_k f_k$ (which by the way is another vector of size 4000). What should be immediately clear from this example is that a naieve approach concerning the organization of the calculations will not do. A more sophisticated approach is desirable an in some sense even necessary. This appendix is devoted to the problem of obtaining the desired quantities in the LM algorithm with a reduced amount of required memory space for storage of variables. We shall discuss two approaches. The first one involves a recursive strategy with respect to calculation of the QR–decomposition of J_k and of vector $Q_k f_k$, where the recursion is applied with respect to the m rows of J_k. The second concerns the problem of obtaining all desired information from the knowledge of GN matrix $\Gamma_k = J_k^T J_k$ and gradient $g_k = J_k^T f_k$ instead of from J_k and f_k. Its motivation, apart from the fact that the sizes of Γ_k and g_k are independent of m, stems also from the observation (see [29]) that in our application it is possible to obtain those with reduced computer memory space requirements.

B.1 Recursive calculation of the QR–decomposition of J_k

Close inspection of what is really needed in the calculation scheme to obtain the GN or LM step, shows that it is essential to obtain matrices T_k, S_k and π_k and also vector u_k that consists of the first r_k components of vector $Q_k f_k$. Matrices T_k and S_k represent the nonzero part of $Q_k J_k \pi_k$. It is *not* essential that matrices Q_k and π_k are identical to those obtained in the original algorithm, but we *must* have that the properties of π_k being a permutation, Q_k being orthogonal and T_k being nonsingular, upper triangular, still hold. At least, this is the case if we want to stay close to the further calculations performed in Moré's implementation of the algorithm.
The present subsection shall be devoted to obtaining a scheme for calculating a QR–decomposition (with column pivoting) of J_k, that uses a recursion on the rows of J_k. The advantage of such a scheme lies in the fact that the essential quantities mentioned above all have sizes that are independent of m, so that also the required amount of computer memory space will be independent of m. We propose the following scheme.
Suppose J is an $m \times n$ matrix, which can be QR–decomposed as

$$J = Q^T \begin{pmatrix} T & S \\ O & O \end{pmatrix} \pi^T \tag{B.1}$$

with Q an $m \times m$ orthogonal matrix, T an $r \times r$ upper triangular, nonsingular matrix and π an $n \times n$ permutation matrix, where r denotes the rank of J. The zero blocks are supposed to be of appropriate size.
Let $\tilde{J}$ be defined as the matrix obtained by adding an extra row x^T to J, that is,

$$\tilde{J} = \begin{pmatrix} J \\ x^T \end{pmatrix} \tag{B.2}$$

Then obviously the following relation holds.

$$\tilde{J} = \begin{pmatrix} Q & 0 \\ 0 & 1 \end{pmatrix}^T \begin{pmatrix} T & S \\ O & O \\ \multicolumn{2}{c}{x^T \pi} \end{pmatrix} \pi^T \tag{B.3}$$

(Notice that the premultiplying matrix is still orthogonal.) This makes clear that we have to find an orthogonal matrix $\bar{Q}$ of size $(m+1) \times (m+1)$ and a postmultiplying permutation $\bar{\pi}$ such that

$$\bar{Q} \begin{pmatrix} T & S \\ O & O \\ \multicolumn{2}{c}{x^T\pi} \end{pmatrix} \bar{\pi} = \begin{pmatrix} \tilde{T} & \tilde{S} \\ O & O \end{pmatrix} \tag{B.4}$$

where $\tilde{T}$ is still upper triangular and nonsingular, of size $\tilde{r} \times \tilde{r}$, with $\tilde{r}$ denoting the rank of $\tilde{J}$.
Now, this problem is not too difficult to solve: we can do exactly the same as in the original algorithm when matrix $\begin{pmatrix} T_k & S_k \end{pmatrix}$ is augmented by an extra block row containing $\sqrt{\lambda}D_k\pi_k$ (see Sect. 2, the comments to step III), that is, we can premultiply by (orthogonal) Givens transformations in order to obtain the required form. As a matter of fact, there is no need for a postmultiplying permutation matrix (which is not so surprising, since one can always apply QR-decomposition also without column pivoting). But notice that in case one does apply an extra column permutation, the structure of the augmented matrix (which is already almost in triangular form) gets distorted, so that the number of required Givens transformations can increase considerably. (In the present situation we only need n of them.)
After having obtained $\bar{Q}$ (and possibly $\bar{\pi}$) we can calculate the new over-all matrices $\tilde{Q}$ and $\tilde{\pi}$ as

$$\tilde{Q} = \bar{Q} \begin{pmatrix} Q & 0 \\ 0 & 1 \end{pmatrix}, \qquad \tilde{\pi} = \pi\bar{\pi} \tag{B.5}$$

Next we proceed with matrices $\tilde{T}$ and $\tilde{S}$, after adding a next row, in a similar fashion.
Remark that we can keep the amount of required memory space for storage of variables limited, because we know that the rank of the augmented matrices can never exceed n. Furthermore, the zero block row is of no interest for the calculations and can be deleted, provided we introduce some bookkeeping. This should involve especially the calculation of u_k. Indeed, if J_k is obtained by adding rows, then in a similar way f_k is obtained by adding more and more elements. Premultiplication by the orthogonal matrix that is obtained in a step of the recursive procedure sketched above should also be applied to vector f_k. After that operation the orthogonal matrix is no longer needed and can be deleted, leaving memory space for the orthogonal matrix to be obtained in the next step.

B.2 Calculations from the knowledge of $\Gamma_k = J_k^T J_k$ and $g_k = J_k^T f_k$

In [29] we discuss an application to the field of system identification, where Γ_k and g_k can be obtained with a reduced amount of required memory space, independent of m. We shall presently indicate how these quantities can be used as an alternative startingpoint for the calculations in the LM algorithm, instead of the quantities J_k and f_k.
In the notation of App. A, the calculations in iteration k of the LM algorithm are motivated by minimization of criterion $c_\lambda(p) = ||J_kp + f_k||^2 + \lambda||D_kp||^2$. We can rewrite this expression as

$$c_\lambda(p) = p^T J_k^T J_k p + 2p^T J_k^T f_k + f_k^T f_k + \lambda p^T D_k^T D_k p \tag{B.6}$$

Therefore, minimization of $c_\lambda(p)$ is equivalent to minimization of

$$p^T \Gamma_k p + 2p^T g_k + \lambda p^T D_k^T D_k p \tag{B.7}$$

Thus, the problem of minimizing $c_\lambda(p)$ is still well-defined if Γ_k and g_k are available instead of J_k and f_k. It is our purpose, however, to solve the latter problem in a way that is as closely related as possible to the LM algorithm of [27], in order to be able to take advantage of that approach. In the sequel we show how this goal can be achieved.

We have the following alternative algorithm for the calculation of $p_k(\lambda)$ from Γ_k and g_k, that should be compared with the algorithms given in the comments to steps II and III in Sect. 2.

Alternative calculation of $p_k(\lambda)$

* Determine Γ_k and g_k.

* Apply Choleski decomposition to Γ_k. This gives a factor G_k (upper triangular, with possibly zeros on its main diagonal) such that $\Gamma_k = G_k^T G_k$.

* Apply QR–decomposition with column pivoting to G_k. This yields a permutation matrix $\tilde{\pi}_k$, an upper triangular, nonsingular $r_k \times r_k$ matrix $\tilde{T}_k$, an $r_k \times (n - r_k)$ matrix $\tilde{S}_k$ and an orthogonal $n \times n$ matrix $\tilde{Q}_k$ such that

$$G_k = \tilde{Q}_k^T \begin{pmatrix} \tilde{T}_k & \tilde{S}_k \\ O & O \end{pmatrix} \tilde{\pi}_k^T \tag{B.8}$$

  where r_k denotes the rank of G_k, which is equal to the rank of Γ_k and also to the rank of J_k (accounting for the omission of a tilde).

* Calculate

$$\tilde{u}_k = \begin{pmatrix} \tilde{T}_k^{-T} & O \end{pmatrix} \tilde{\pi}_k^T g_k \tag{B.9}$$

* Use $\tilde{u}_k$ as u_k, $\tilde{\pi}_k$ as π_k and $\begin{pmatrix} \tilde{T}_k & \tilde{S}_k \end{pmatrix}$ as $\begin{pmatrix} T_k & S_k \end{pmatrix}$ in all further calculations, that is: define

$$\tilde{p}_k(0) = \tilde{\pi}_k \begin{pmatrix} \tilde{T}_k^{-1} \\ O \end{pmatrix} \tilde{u}_k \tag{B.10}$$

  to be the GN solution to the minimization problem (with $\lambda = 0$), and next apply further calculations for obtaining $p_k(\lambda)$ as described in the comments to Sect. 2, step III.

We have the following lemmas that are useful for proving the validity of the scheme above and pointing out some further relationships between the two alternatives.

Lemma B.1 *Let A and B be $r \times n$ matrices of full row rank $r(\leq n)$, such that $A^T A = B^T B$. Then there exists a (unique) orthogonal matrix Q of size $r \times r$ such that $A = QB$.*

Proof Matrix A has full rank r, so we can select r independent columns from A, or equivalently r independent rows from A^T. The i–th row of $A^T A$ is formed as the product of the i–th row of A^T with matrix A. Thus we see that the row space of $A^T A$ (the space spanned by its rows) is identical to the row space of A.
Similarly, we see that the row space of $B^T B$ is identical to the row space of B. Because, by assumption, $A^T A = B^T B$ we find that A and B have the same row space. This means that the rows of A can be expressed as a linear combination of the rows of B (and vice versa).
Therefore, there exists a (nonsingular) $r \times r$ matrix Q such that $A = QB$. (As a matter of fact we also see that Q is unique.)
Now the identity $A^T A = B^T B$ leads us to $B^T Q^T Q B = B^T B$, which can be written as $B^T(Q^T Q - I)B = O$, the zero $n \times n$ matrix.
From B we can select r independent columns. Putting them in an $r \times r$ matrix C we see that the product $C^T(Q^T Q - I)C$ forms a submatrix of $B^T(Q^T Q - I)B$ and therefore equals zero as well. But C is invertible (and C^T also) so that we find $Q^T Q - I = O$.
This shows that $Q^T Q = I$, whence Q is orthogonal. □

A special case of Lemma B.1 occurs for $r = n$, i.e. when we have square matrices A and B of full rank.
For matrices that are not of full rank we have the following result, for which its proof shows that we no longer have uniqueness of Q.

Lemma B.2 *Let A and B be $m \times n$ matrices of rank r, such that $A^T A = B^T B$. Then there exists an orthogonal matrix Q of size $m \times m$ such that $A = QB$.*

Proof We first make use of the well–known fact that one can apply QR–decomposition to matrix A, yielding an $m \times m$ orthogonal matrix Q_A and an $m \times n$ matrix R_A of the following structure:

$$R_A = \begin{pmatrix} T_A & S_A \\ O & O \end{pmatrix} \tag{B.11}$$

where T_A is $r \times r$, upper triangular, nonsingular and S_A is $r \times (n-r)$.
Similarly, one can apply QR–decomposition to B, yielding matrices Q_B, R_B, T_B and S_B.
We have

$$\begin{pmatrix} T_A & S_A \end{pmatrix}^T \begin{pmatrix} T_A & S_A \end{pmatrix} = A^T A = B^T B = \begin{pmatrix} T_B & S_B \end{pmatrix}^T \begin{pmatrix} T_B & S_B \end{pmatrix} \tag{B.12}$$

Application of Lemma B.1 to $\begin{pmatrix} T_A & S_A \end{pmatrix}$ and $\begin{pmatrix} T_B & S_B \end{pmatrix}$ then yields the existence of a (unique) orthogonal matrix $\bar{Q}$ of size $r \times r$ such that $\begin{pmatrix} T_A & S_A \end{pmatrix} = \bar{Q} \begin{pmatrix} T_B & S_B \end{pmatrix}$. Thus, R_A and R_B are related, for instance, by

$$R_A = \begin{pmatrix} \bar{Q} & O \\ O & I \end{pmatrix} R_B \tag{B.13}$$

(Notice that instead of block I one can put *any* orthogonal matrix of the appropriate size. This shows the nonuniqueness mentioned earlier.) We find

$$A = Q_A R_A = Q_A \begin{pmatrix} \bar{Q} & O \\ O & I \end{pmatrix} R_B = Q_A \begin{pmatrix} \bar{Q} & O \\ O & I \end{pmatrix} Q_B^T B \tag{B.14}$$

Defining Q by

$$Q = Q_A \begin{pmatrix} \bar{Q} & O \\ O & I \end{pmatrix} Q_B^T \tag{B.15}$$

we finally see that $A = QB$, where Q is $m \times m$ orthogonal. □

We now can prove the following result, which is essential for the correctness of the alternative algorithm.

Lemma B.3 *Let J_k be an $m \times n$ matrix of rank r_k. Define $\Gamma_k = J_k^T J_k$, of size $n \times n$. Let K_k be any matrix such that $K_k^T K_k = \Gamma_k$.*
Apply QR-decomposition with column pivoting to K_k, yielding an appropriately sized orthogonal matrix $\tilde{Q}_k$, an $n \times n$ permutation matrix $\tilde{\pi}_k$ and an appropriately sized matrix $\tilde{R}_k$ with the structure $\tilde{R}_k = \begin{pmatrix} \tilde{T}_k & \tilde{S}_k \\ O & O \end{pmatrix}$, where $\tilde{T}_k$ is $r_k \times r_k$, upper triangular and invertible, with r_k denoting the rank of K_k (equal to the rank of J_k), such that holds $K_k = \tilde{Q}_k^T \tilde{R}_k \tilde{\pi}_k^T$.
Then there exists an $m \times m$ orthogonal matrix $\bar{Q}_k$ such that $J_k = \bar{Q}_k^T R_k \tilde{\pi}_k^T$, where R_k is defined as the $m \times n$ matrix $\begin{pmatrix} \tilde{T}_k & \tilde{S}_k \\ O & O \end{pmatrix}$ (so with the zero blocks in $\tilde{R}_k$ adapted to the appropriate size).

Proof We can write

$$\Gamma_k = K_k^T K_k = \tilde{\pi}_k \tilde{R}_k^T \tilde{R}_k \tilde{\pi}_k^T \tag{B.16}$$

so that

$$\tilde{\pi}_k^T \Gamma_k \tilde{\pi}_k = \tilde{R}_k^T \tilde{R}_k \tag{B.17}$$

Notice that, by definition of R_k, we also have

$$\tilde{\pi}_k^T \Gamma_k \tilde{\pi}_k = R_k^T R_k \tag{B.18}$$

Introduce $m \times n$ matrix L_k, to be defined as $L_k = J_k \tilde{\pi}_k$. Then L_k and R_k are of the same size, with $L_k^T L_k = R_k^T R_k$. Application of Lemma B.2 then yields the existence of an $m \times m$ orthogonal matrix $\bar{Q}_k$ such that $R_k = \bar{Q}_k L_k$.
This leads to

$$J_k = L_k \tilde{\pi}_k^T = \bar{Q}_k^T R_k \tilde{\pi}_k^T \tag{B.19}$$

which proves the proposition. □

Theorem B.4 *The alternative algorithm for calculating $p_k(\lambda)$ is correct.*

Proof From Lemma B.3 we see that application of the alternative algorithm leads to matrices $\tilde{T}_k$, $\tilde{S}_k$ and $\tilde{\pi}_k$ that might have occurred in the original version also, when QR–decomposing J_k with column pivoting. Since the column selection strategy is of no importance for the validity of the further calculations (it is only the structure of the achieved decomposition that matters), there is no incorrectness in using these matrices instead of their counterparts without tilde.
Further inspection of the original calculations shows that what is further needed to obtain $\tilde{p}_k(0)$ (and in at later stage $p_k(\lambda)$), is vector $\tilde{u}_k$, consisting of the first r_k components of $\bar{Q}_k f_k$ (in the notation of Lemma B.3).
Now, notice that we have the gradient g_k at our disposal. We can write

$$g_k = J_k^T f_k = \tilde{\pi}_k R_k^T \bar{Q}_k f_k \tag{B.20}$$

so that $\tilde{u}_k$ is obtained as

$$\tilde{u}_k = \begin{pmatrix} \tilde{T}_k^{-T} & O \end{pmatrix} \tilde{\pi}_k^T g_k = \begin{pmatrix} \tilde{T}_k^{-T} & O \end{pmatrix} \begin{pmatrix} \tilde{T}_k^T & O \\ \tilde{S}_k^T & O \end{pmatrix} \bar{Q}_k f_k = \begin{pmatrix} I & O \end{pmatrix} \bar{Q}_k f_k \tag{B.21}$$

This shows that $\tilde{u}_k$ as calculated in the alternative algorithm represents the first r_k components of $\bar{Q}_k f_k$ and is indeed playing the role of u_k in the original calculations. Hence, the alternative algorithm is correct. □

The importance of the above theorem, showing that we can perform the required calculations for the LM algorithm with a reduced amount of computer memory capacity for data storage that is independent of the value of m has already been discussed before. However, one still may argue that the alternative procedure given above might have numerical properties that are inferior to those of the original one. Apart from the numerical round–off stemming from the explicit calculation of Γ_k that might affect the final results, we shall show that there is in general good reason to expect the alternative algorithm to yield results similar to those of the original one. The point we will make in Theorem B.6 is that the quantities $\tilde{T}_k$, $\tilde{S}_k$, $\tilde{\pi}_k$ and $\tilde{u}_k$ are in fact *identical* to their counterparts without tilde, provided the selection strategy applied when performing QR–decomposition with column pivoting is based on the principle of selecting the column of which the remaining part that is of interest has the largest norm, and moreover an additional sign convention procedure is applied. This strategy (apart from the sign convention procedure) is a very common one and actually we have been using it in the experiments of [29]. (It enables one pretty easily to determine the rank of a matrix in the presence of numerical noise by setting a level for deciding on significant difference from zero.) Before stating the actual theorem we shall discuss the algorithm for QR–decomposition with column pivoting based on this strategy in some more detail.
The procedure can be sketched as follows. In the first iteration, select the column with the largest norm of the matrix to be operated on. Interchange this column with the first one by postmultiplication with a suitable permutation matrix. Next, premultiplicate with a suitable Householder transformation (as in standard QR–decomposition) to obtain a first column that is a multiple of the first unit column vector e_1. In subsequent iterations we proceed in a similar way, now leaving the rows and columns that have already been handled out of it. Each postmultiplying permutation matrix is now formed as a 2 by 2 block matrix with the (1,1)–block equal to I, the (1,2)–block and (2,1)–block equal to zero and the (2,2)–block containing the actual permutation (all of appropriate size). Each premultiplying orthogonal matrix has a similar structure, with the (2,2)–block containing the actual Householder transformation. (Notice that each 2 by 2 block matrix in such a case is itself still a Householder transformation matrix.) When selecting a column in a new iteration we only calculate the norms of the lower parts of the columns that are still of interest, and compare these. As soon as the last column has been handled we stop. If we encounter a situation where the largest norm (of the remaining parts of the columns) is less than a certain specified tolerance value (e.g. 10^{-12}), we set all remaining matrix elements to zero. All postmultiplying permutations then build the overall permutation matrix π, and all premultiplying Householder transformation matrices build the overall orthogonal matrix Q. The matrix itself has been transformed into the upper triangular factor R.

Lemma B.5 *Let J be an $m \times n$ matrix and K be a $p \times n$ matrix such that $J^TJ = K^TK$. Then if we apply QR–decomposition with column pivoting to J and K respectively, based on the column selection strategy described above, we find (in the same notation as before) that $J = Q^TR\pi^T$ and $K = \tilde{Q}^T\tilde{R}\tilde{\pi}^T$, where $\tilde{\pi} = \pi$, $\tilde{T} = DT$ and $\tilde{S} = DS$, with D a diagonal matrix with all elements on its main diagonal from the set $\{-1, 1\}$.*

Proof By induction. To start with, notice that the squared values of the lengths of the columns of J appear on the main diagonal of J^TJ in the same order. Because $J^TJ = K^TK$ we see that the selection procedure, when applying QR–decomposition with column pivoting to J and K, leads in both cases to the same initial permutation matrix by which is postmultiplied. Denoting the first post- and premultiplying matrices with a subscript 1, we have that after the first iteration of the QR–algorithm we are left with

$$Q_1J\pi_1 = \begin{pmatrix} x_1 & y_1^T \\ 0 & J_1 \end{pmatrix} \quad \text{and} \quad \tilde{Q}_1K\tilde{\pi}_1 = \begin{pmatrix} \tilde{x}_1 & \tilde{y}_1^T \\ 0 & K_1 \end{pmatrix} \tag{B.22}$$

where $\pi_1 = \tilde{\pi}_1$, and with x_1 and $\tilde{x}_1$ denoting nonzero scalars (provided J and K are nonzero matrices).
Since $J^TJ = K^TK$ and $\pi_1 = \tilde{\pi}_1$, whereas Q_1 and $\tilde{Q}_1$ are orthogonal, we see that

$$(Q_1J\pi_1)^TQ_1J\pi_1 = (\tilde{Q}_1K\tilde{\pi}_1)^T\tilde{Q}_1K\tilde{\pi}_1 \tag{B.23}$$

Hence

$$\begin{pmatrix} x_1^2 & x_1y_1^T \\ x_1y_1 & J_1^TJ_1 + y_1y_1^T \end{pmatrix} = \begin{pmatrix} \tilde{x}_1^2 & \tilde{x}_1\tilde{y}_1^T \\ \tilde{x}_1\tilde{y}_1 & K_1^TK_1 + \tilde{y}_1\tilde{y}_1^T \end{pmatrix} \tag{B.24}$$

Define d_1, the first diagonal element of D, as $\tilde{x}_1/x_1$. From the matrix identity above we have that $d_1 \in \{-1, 1\}$, since $x_1^2 = \tilde{x}_1^2$. As a result we also have that $\tilde{y}_1 = d_1y_1$. Therefore $y_1y_1^T = \tilde{y}_1\tilde{y}_1^T$, whence $J_1^TJ_1 = K_1^TK_1$.
In the next iteration of the QR–algorithm we proceed with J_1 and K_1 in a similar way. The obtained elements x_1, $\tilde{x}_1$, y_1 and $\tilde{y}_1$ remain unaffected. Thus, we can conclude (since the squared values of the lengths of the remaining parts of the columns, i.e. of the columns of J_1 and K_1 respectively, appear on the main diagonal of $J_1^TJ_1 = K_1^TK_1$), that $\pi_2 = \tilde{\pi}_2$.
By induction we obviously find that $\pi = \tilde{\pi}$. Moreover, from the way d_1 is obtained we see, by induction, that $D = \text{diag}\{d_1, \ldots, d_r\}$ will satisfy the required properties, while $\tilde{T} = DT$ and $\tilde{S} = DS$.
The latter conclusion can alternatively be obtained from the result that $\pi = \tilde{\pi}$ in the following way.
If $\pi = \tilde{\pi}$ we have from $J^TJ = K^TK$ that

$$\begin{pmatrix} T^T \\ S^T \end{pmatrix} \begin{pmatrix} T & S \end{pmatrix} = \begin{pmatrix} \tilde{T}^T \\ \tilde{S}^T \end{pmatrix} \begin{pmatrix} \tilde{T} & \tilde{S} \end{pmatrix} \tag{B.25}$$

Therefore $T^TT = \tilde{T}^T\tilde{T}$. By Lemma B.1 we see that there exists an orthogonal matrix $\tilde{D}$ such that $\tilde{T} = DT$. But T is invertible and upper triangular, so its inverse is also upper triangular. Consequently, $D = \tilde{T}T^{-1}$ is upper triangular. Hence, also the inverse of D is upper triangular. But since D is orthogonal we know that $D^{-1} = D^T$, which must be lower triangular. So D must be diagonal. Orthogonality of the diagonal matrix D then implies that all its elements are from $\{-1, 1\}$, since $D^2 = I$.
Next, we use that $T^TS = \tilde{T}^T\tilde{S} = T^TD^T\tilde{S}$, whence $\tilde{S} = DS$. This completes the proof. □

The next theorem then easily follows.

Theorem B.6 *When applying the alternative algorithm for calculating $p_k(\lambda)$, the quantities $\tilde{T}_k$, $\tilde{S}_k$, $\tilde{\pi}_k$ and $\tilde{u}_k$ are identical to their counterparts without tilde, provided the selection strategy applied when performing QR–decomposition with column pivoting is as described above, i.e. based on the principle of selecting the column of which the remaining part of interest has the largest norm, with the modification that we also apply a sign convention procedure, making sure that the diagonal elements of T_k and $\tilde{T}_k$ are all positive.*

Proof This is an immediate consequence of Lemma B.5, since we have that $J_k^T J_k = \Gamma_k = K_k^T K_k$ and we apply QR–decomposition with column pivoting to either J_k or K_k. The extra sign convention consists of choosing matrix D (in Lemma B.5) appropriately, that is, by requiring the diagonal elements of T_k and $\tilde{T}_k$ to be positive. $\square$

Remark In the first case that has been studied in this appendix, where a recursion was applied to the rows of J_k using Givens rotations, we shall in general *not* obtain the same matrices T and S as for the original algorithm.

In order to complete the discussion about these approaches for calculating the LM steps, using a reduced amount of memory space for the storage of variables, we must also pay attention to the following. In the LM algorithm implementation of [27] there is also a step where the trust–region size is adapted, based on the experienced validity of the linearization around the current iterate. For that, use is made of the quantities $\|J_k p_k\|^2$ and $\|f_k\|^2$, in particular in the definitions of ρ_k and γ_k and in the convergence criterion (see Sect. 2, the comments to step IV, V and VI). Notice that $\frac{1}{2}\|f_k\|^2$ denotes the current criterion value, which can be computed recursively in our application as well (being the sum of squares). Also $\|J_k p_k\|^2$ can be computed without explicit knowledge of J_k, since it is identical to $p_k^T J_k^T J_k p_k = p_k^T \Gamma_k p_k$, where Γ_k is available. Therefore, steps IV, V and VI do not require the explicit knowledge of J_k and f_k either, making the above approaches applicable.

Appendix C : Analysis of Moré's scaling strategies

In Sect. 2, in the comments to step VII of the LM algorithm, we have claimed that the scaling strategies proposed in [27] are unsatisfactory and not properly based. The purpose of this appendix is to motivate this claim by presenting the results of experiments carried out with the same four test–problems as studied in [27]. The second objective of this appendix is to investigate and compare the performance of the stopping criteria of Moré and Marquardt (see Sect. 2, step V).
The test–problems under consideration are the following.

1. Fletcher and Powell [10], $n = 3$ and $m = 3$.
   The residual mapping f is described by

$$f^1(x^1,x^2,x^3) = 10[x^3 - 10\theta(x^1,x^2)],$$

$$f^2(x^1,x^2,x^3) = 10[\sqrt{[x^1]^2+[x^2]^2} - 1],$$

$$f^3(x^1,x^2,x^3) = x^3,$$

   with

$$\theta(x^1,x^2) = \begin{cases} \frac{1}{2\pi}\arctan\left(\frac{x^2}{x^1}\right) & \text{if } x^1 > 0, \\ \frac{1}{2\pi}\arctan\left(\frac{x^2}{x^1}\right) + \frac{1}{2} & \text{if } x^1 < 0. \end{cases}$$

   The starting point is

$$x_0 = (-1,0,0).$$

   There exists a global minimum at (1,0,0) with residual norm 0. There is a discontinuity along the plane $x^1 = 0$, which must be crossed in order to get to the optimum.

2. Kowalik and Osborne [19], $n = 4$ and $m = 11$.
   Here f is given by

$$f^i(x^1,x^2,x^3,x^4) = y^i - \frac{x^1([u^i]^2 + x^2u^i)}{([u^i]^2 + x^3u^i + x^4)} \quad (i = 1,\ldots,11)$$

   where

$$(u^1,\ldots,u^{11}) = (4,2,1,.5,.25,.167,.125,.1,.0823,.0714,.0625),$$

$$(y^1,\ldots,y^{11}) = (.1957,.1947,.1735,.1600,.0844,.0627,.0456,.0342,.0323,.0235,.0246).$$

   The starting point is

$$x_0 = (.25,.39,.415,.39).$$

   A global minimum occurs at (.1928,.1938,.1246,.1370) with corresponding residual norm $1.76188 \cdot 10^{-2}$. (In [27] the value $1.7536 \cdot 10^{-2}$ for the optimal residual norm is given, which is equal to the value specified in [19]. There the optimum is said to be located at (.1928,.1916,.1234,.1362). These small differences are probably caused by numerical round–off contained by the printed data in [19].) This is a data fitting problem with small residual norm. There exist infima at infinity for which some parameters are drifting away and others remain bounded.

3. Bard [2], $n = 3$ and $m = 15$.
   In this case f is given by

$$f^i(x^1,x^2,x^3) = y^i - \left[x^1 + \frac{u^i}{(x^2v^i + x^3w^i)}\right] \qquad (i = 1,\dots,15)$$

where u^i, v^i and w^i are defined as

$$u^i = i, \quad v^i = 16 - i, \quad w^i = \min(u^i, v^i),$$

and y^i is given by

$$(y^1,\dots,y^{15}) = (.14,.18,.22,.25,.29,.32,.35,.39,.37,.58,.73,.96,1.34,2.10,4.39)$$

The starting point is

$$x_0 = (1,1,1).$$

This is again a data fitting problem for which infima at infinity exist. Its global minimum occurs at $(.0824, 1.1330, 2.3437)$ and leads to a residual norm 9.063596.

4. Brown and Dennis [6], $n = 4$ and $m = 20$.
For this problem f is given by

$$f^i(x^1,x^2,x^3,x^4) = (x^1 + x^2t^i - \exp(t^i))^2 + (x^3 + x^4\sin(t^i) - \cos(t^i))^2, \qquad (i = 1,\dots,20)$$

where $t^i = (.2)i$.
The starting point is

$$x_0 = (25, 5, -5, 1).$$

The global minimum occurs at $(-11.5944, 13.2036, -.4034, .2368)$ with residual norm 292.9543. There are no other local minima. This is a problem with a large residual.

We have carried out experiments with these test–problems, starting from three different starting points: x_0, $10x_0$ and $100x_0$. We investigated four different scaling strategies, being

1. *No Scaling.* Here we choose $D_k = I$ at each iteration, irrespective of the value of x_k.
2. *Initial Scaling.* We choose $D_k = D_0$, with D_0 chosen as explained in Sect. 2, steps I and VII.
3. *Adaptive Scaling.* See Sect. 2, step VII.
4. *Continuous Scaling.* See also Sect. 2, steps I and VII. D_k is chosen at x_k like D_0 is chosen at x_0 for strategy 2.

Furthermore we have compared the effect of the stopping criteria of Moré and Marquardt on the number of iterations required to reach an optimum. The results of the experiments are given in Tables 1–4. In a number of situations FC is specified, indicating "failed convergence." The experiments were carried out using the MATLAB environment, so that certain automatic warnings in case of bad conditioning of matrices or division by zero were issued. In principle, all experiments were started with Δ_0 (the initial trust–region size parameter) set to 1. In contrast with Figure 1 of Sect. 2 we have counted an iteration also in case the step was not accepted (i.e. $\rho < 10^{-4}$). This in order to make a fair comparison with the results of [27].

Test–problem 1.

[1] In the last iteration a "division by zero" warning was issued by MATLAB. After 17 iterations we had a residual norm $2.0134 \cdot 10^{-15}$ at $(1, -1.265 \cdot 10^{-16}, 0)$. This coincides with the result after 16 iterations (see the result obtained for Marquardt's stopping criterion), so that apparently no step was accepted in the 17–th iteration.

[2] In the last iteration a "division by zero" warning was issued by MATLAB. After 26 iterations we had a residual norm $8.9754 \cdot 10^{-17}$ at $(1, 5.6394 \cdot 10^{-8}, 0)$.

[3] FC indicated because convergence was very slow. (Several thousands of iterations required.) Typical gradient–like behaviour takes place: zigzagging of some variables and slow, smooth convergence of other ones. Obviously, the problem is now badly scaled.

starting	scaling	Moré's stopping criteria		Marquardt's stopping criterium		Moré's
point	strategy	# iterations	residual norm	# iterations	residual norm	results
x_0	1	15	$2.7878 \cdot 10^{-45}$	13	$4.3160 \cdot 10^{-14}$	—
x_0	2	15	0	14	$1.7919 \cdot 10^{-25}$	12
x_0	3	15	$1.2261 \cdot 10^{-32}$	14	$8.4338 \cdot 10^{-17}$	11
x_0	4	18 [1]	0	16	$2.0134 \cdot 10^{-15}$	12
$10x_0$	1	19	0	18	$6.0804 \cdot 10^{-18}$	—
$10x_0$	2	27	$1.1419 \cdot 10^{-41}$	26	$6.4282 \cdot 10^{-26}$	34
$10x_0$	3	23	$3.6540 \cdot 10^{-40}$	22	$2.5022 \cdot 10^{-24}$	20
$10x_0$	4	26	0	25	$4.4882 \cdot 10^{-20}$	14
$100x_0$	1	27 [2]	0	25	$8.9754 \cdot 10^{-17}$	—
$100x_0$	2	FC [3]	—	FC [3]	—	FC
$100x_0$	3	101	0	100	$8.5246 \cdot 10^{-20}$	19
$100x_0$	4	227	0	226	$1.9686 \cdot 10^{-26}$	176

Table 1. Results for test–problem 1.

starting	scaling	Moré's stopping criteria		Marquardt's stopping criterium		Moré's
point	strategy	# iterations	residual norm	# iterations	residual norm	results
x_0	1	23	$1.7619 \cdot 10^{-2}$	25	$1.7619 \cdot 10^{-2}$	—
x_0	2	10	$1.7619 \cdot 10^{-2}$	13	$1.7619 \cdot 10^{-2}$	19 [4]
x_0	3	10	$1.7619 \cdot 10^{-2}$	13	$1.7619 \cdot 10^{-2}$	18 [4]
x_0	4	10	$1.7619 \cdot 10^{-2}$	13	$1.7619 \cdot 10^{-2}$	18 [4]
$10x_0$	1	33	$1.7619 \cdot 10^{-2}$	35	$1.7619 \cdot 10^{-2}$	—
$10x_0$	2	44	$1.7619 \cdot 10^{-2}$	46	$1.7619 \cdot 10^{-2}$	81
$10x_0$	3	42	$1.7619 \cdot 10^{-2}$	44	$1.7619 \cdot 10^{-2}$	79
$10x_0$	4	53	$1.7619 \cdot 10^{-2}$	56	$1.7619 \cdot 10^{-2}$	63
$100x_0$	1	99	$1.7619 \cdot 10^{-2}$	101	$1.7619 \cdot 10^{-2}$	—
$100x_0$	2	113	$1.7619 \cdot 10^{-2}$	115	$1.7619 \cdot 10^{-2}$	365 [4]
$100x_0$	3	111	$1.7619 \cdot 10^{-2}$	113	$1.7619 \cdot 10^{-2}$	348 [4]
$100x_0$	4	FC [5]	$4.2317 \cdot 10^{-2}$	FC [5]	$4.2317 \cdot 10^{-2}$	FC [4]

Table 2. Results for test–problem 2.

Test–problem 2.

[4] Moré states that in all these cases convergence to an infimum at infinity took place.

[5] In these cases "division by zero" occurred after 106 iterations. The residual norm reported corresponds to the estimate then; this is $(11.6, 1.066 \cdot 10^4, 5.335 \cdot 10^5, 2.169 \cdot 10^5)$. A logarithmic plot of the estimates shows that x^2, x^3 and x^4 are drifting to infinity at almost the same rate. Convergence to an infimum at infinity seems to take place, but notably a different one as specified in [27].

starting	scaling	Moré's stopping criteria		Marquardt's stopping criterium		Moré's
point	strategy	# iterations	residual norm	# iterations	residual norm	results
x_0	1	5	$9.0636 \cdot 10^{-2}$	5	$9.0636 \cdot 10^{-2}$	—
x_0	2	7	$9.0636 \cdot 10^{-2}$	6	$9.0636 \cdot 10^{-2}$	8
x_0	3	7	$9.0636 \cdot 10^{-2}$	6	$9.0636 \cdot 10^{-2}$	8
x_0	4	6	$9.0636 \cdot 10^{-2}$	6	$9.0636 \cdot 10^{-2}$	8
$10x_0$	1	14	$9.0636 \cdot 10^{-2}$	14	$9.0636 \cdot 10^{-2}$	—
$10x_0$	2	48 [7]	4.17477	48 [7]	4.17477	37 [6]
$10x_0$	3	48 [8]	4.17477	48 [8]	4.17477	37 [6]
$10x_0$	4	7 [9]	4.17477	7 [9]	4.17477	FC [6]
$100x_0$	1	23	$9.0636 \cdot 10^{-2}$	23	$9.0636 \cdot 10^{-2}$	—
$100x_0$	2	19 [10]	4.17477	19 [10]	4.17477	14 [6]
$100x_0$	3	19 [11]	4.17477	19 [11]	4.17477	14 [6]
$100x_0$	4	FC [12]	—	FC [12]	—	FC [6]

Table 3. Results for test–problem 3.

starting	scaling	Moré's stopping criteria		Marquardt's stopping criterium		Moré's
point	strategy	# iterations	residual norm	# iterations	residual norm	results
x_0	1	27	292.9543	31	292.9543	—
x_0	2	424	292.9543	382 [14]	292.9546	268
x_0	3	424	292.9543	382 [14]	292.9546	268
x_0	4	FC [13]	—	FC [13]	—	FC
$10x_0$	1	35	292.9543	41	292.9543	—
$10x_0$	2	441	292.9543	1089 [15]	292.9543	423
$10x_0$	3	288 [16]	292.9543	60 [15]	292.9543	57
$10x_0$	4	FC [13]	—	FC [13] [15]	—	FC
$100x_0$	1	37	292.9543	43	292.9543	—
$100x_0$	2	1108	292.9543	750 [17]	292.9543	FC
$100x_0$	3	169 [18]	292.9543	91 [17]	292.9543	229
$100x_0$	4	FC [13]	—	FC [13] [17]	—	FC

Table 4. Results for test–problem 4.

Test–problem 3.
[6] Moré states that in all these cases convergence to an infimum at infinity took place.
[7] MATLAB issued "division by zero" and "matrix singular to working precision" warnings since iteration 42. Convergence to infinity, as specified in [27], took place. We notice that x^1 converges to $\frac{1261}{1500} = 0.84066667$, and the other coordinates to infinity. The fact that the number of iterations is equal for both stopping criteria is a consequence of the way MATLAB proceeds after the warnings being issued. Stopping occurred because the norm of the proposed step p was of order 10^{-8}.
[8] As under [7], now with warnings since iteration 43.
[9] As under [7], now with warnings since iteration 13.
[10] As under [7], again with warnings since iteration 13.
[11] Convergence as under [7] took place. Warnings were issued only during iteration 5. In these cases the estimate of x^1 after 7 iterations was very close to the specified asymptotic value.
[12] Numerical inaccuracies detected since iteration 4. (Condition numbers of order 10^{-19}.) A crash occurred after 8 iterations.

Test–problem 4.
[13] FC indicated because convergence was extremely slow (thousands of iterations required). In these situations the scaling is very bad.
[14] Notice that convergence was detected with less accuracy than in all other cases.
[15] The initial value for Δ_0 was taken to be 100. For the standard choice $\Delta_0 = 1$ termination after 1 iteration occurred. Compare with the remarks to step V in Sect. 2.
[16] The initial value for Δ_0 was 1. For $\Delta_0 = 100$ only 60 iterations were needed.
[17] The initial value for Δ_0 was taken to be 10000. For the choices $\Delta_0 = 1$ and $\Delta_0 = 100$ termination after 1 iteration occurred. Compare with the remarks to step V in Sect. 2.
[18] The initial value for Δ_0 was 1. For $\Delta_0 = 10000$ only 84 iterations were needed.

From these experiments we draw the following conclusions.

1. The scaling strategies of [27] are unsatisfactory. Nonscaling generally has led to superior convergence in all difficult situations. Of course one cannot state that nonscaling is optimal, because the "initial scaling" strategy can be considered as corresponding to nonscaling in a situation where one started with other coordinates. But apparently the proposed scaling strategies did not exhibit good behaviour.

2. There is little difference between the numbers of iterations required for Moré's stopping criteria and those required for Marquardt's criterium. Moré's criteria are more conveniently interpreted, but Marquardt's has the advantage of being applicable to other optimization routines as well, making a comparison between different methods easier.

3. The matter of choosing an initial value for Δ_0 deserves more attention, as it turns out to play an important role in some experiments with test–problem 4.

References

[1] R.A. Abraham, J.E. Marsden, *Foundations of Mechanics* (2nd ed.). Reading, Mass.: Benjamin & Cummings, 1978.

[2] Y. Bard, Comparison of gradient methods for the solution of nonlinear parameter estimation problems, *SIAM J. Num. Anal.* 7, 157–186, 1970.

[3] Y. Bard, *Nonlinear Parameter Estimation.* New York: Academic Press, 1974.

[4] G.J. Bierman, *Factorization Methods for Discrete Sequential Estimation.* New York: Academic Press, 1977.

[5] W.M. Boothby, *An Introduction to Differentiable Manifolds and Riemannian Geometry.* New York: Academic Press, 1975.

[6] K.M. Brown and J.E. Dennis, New computational algorithms for minimizing a sum of squares of nonlinear functions, Department of Computer Science, Report 71–6. New Haven, Connecticut: Yale University, 1971.

[7] J.E. Dennis, Jr. and R.B. Schnabel, *Numerical Methods for Unconstrained Optimization and Nonlinear Equations.* Englewood Cliffs: Prentice–Hall, 1983.

[8] R. Fletcher, A modified Marquardt subroutine for nonlinear least squares, *Atomic Energy Research Establishment, Report R6799*, Harwell, England, 1971.

[9] R. Fletcher, *Practical Methods of Optimization*, Vol. 1: Unconstrained Optimization. New York: John Wiley and Sons, 1980.

[10] R. Fletcher and M.J.D. Powell, A rapidly convergent descent method for minimization, *Comput. J.* 6, 163–168, 1963.

[11] D. Gabay, Minimizing a Differentiable Function over a Differentiable Manifold, *J. of Optimiz. Th. and Appl.* 37, 177–219, 1982.

[12] B. Hanzon, On a Gauss–Newton identification method that uses overlapping parametrizations, *IFAC Identification and System Parameter Estimation 1985, York, UK*, 1671–1676, 1985.

[13] B. Hanzon, *Identifiability, Recursive Identification and Spaces of Linear Dynamical Systems*, CWI Tracts 63, 64. Amsterdam: Centre for Mathematics and Computer Science, 1989.

[14] B. Hanzon and R.L.M. Peeters, On the Riemannian Interpretation of the Gauss–Newton Algorithm, in: M. Kárný and K. Warwick (eds), *Preprints of the IFAC Workshop MICC '92*, 65–70. Prague, 1992. (To appear in the proceedings.)

[15] M. Hazewinkel, Moduli and Canonical Forms for Linear Dynamical Systems II: The Topological Case, *Mathematical Systems Theory* 10, 363–385, 1977.

[16] M.D. Hebden, An algorithm for minimization using exact second derivatives, *Atomic Energy Research Establishment, Report TP515*, Harwell, England, 1973.

[17] A. Kamath and N. Karmarkar, A Continuous Method for Computing Bounds in Integer Quadratic Optimization Problems, *Journal of Global Optimization* 2, 229–241, 1992.

[18] N. Karmarkar, Riemannian Geometry Underlying Interior–Point Methods for Linear Programming, *Contemporary Mathematics* 114, 51–75, 1990.

[19] J. Kowalik and M.R. Osborne, *Methods for Unconstrained Optimization Problems.* New York: American Elsevier, 1968.

[20] C.L. Lawson and R.J. Hanson, *Solving Least Squares Problems.* Englewood Cliffs: Prentice–Hall, 1974.

[21] K. Levenberg, A method for the solution of certain nonlinear problems in least squares, *Quart. Appl. Math.* 2, 164–168, 1944.

[22] A. Lichnewsky, Une méthode de gradient conjugué sur des variétés; application à certains problèmes de valeurs propres non linéaires, *Numer. Funct. Anal. and Optimiz.*, Vol. 1, 515–560, 1979.

[23] A. Lichnewsky, *Minimisation des Fonctionnelles Définies sur une Variété par la Méthode du Gradient Conjugué*, Thèse de Doctorat d'Etat. Paris: Université de Paris–Sud, 1979.

[24] L. Ljung, *System Identification: Theory for the User.* Englewood Cliffs: Prentice–Hall, 1987.

[25] D.G. Luenberger, The Gradient Projection Method along Geodesics, *Management Science* 18, 620–631, 1972.

[26] D.W. Marquardt, An algorithm for least-squares estimation of nonlinear parameters, *SIAM J. Appl. Math.* 11, 431–441, 1963.

[27] J.J. Moré, The Levenberg–Marquardt algorithm: implementation and theory, in: G.A. Watson (ed.) *Numerical Analysis, Lecture Notes in Mathematics* 630, 105–116. Berlin: Springer Verlag, 1977.

[28] R.L.M. Peeters, Identification on a Manifold of Systems, *Serie Research Memoranda* 1992–7. Amsterdam: Free University, Faculty of Economics and Econometrics, 1992.

[29] R.L.M. Peeters, Application of the Riemannian Levenberg–Marquardt Algorithm to Off–line System Identification, *Serie Research Memoranda* 1993–12. Amsterdam: Free University, Faculty of Economics and Econometrics, 1993.

[30] R.L.M. Peeters, *Ph.D. Thesis.* Under preparation.

[31] R.L.M. Peeters and B. Hanzon, The Riemannian Interpretation of Gauss–Newton and Scoring, with Application to System Identification, *Serie Research Memoranda* 1992–22. Amsterdam: Free University, Faculty of Economics and Econometrics, 1992.

[32] T. Söderström and P. Stoica, *System Identification.* New York: Prentice–Hall, 1989.

[33] G. Strang, *Linear Algebra and Its Applications.* New York: Academic Press, 1976.