Science & Technology Facilities Council

Preprint

RAL-P-2017-010

A higher order method for solving nonlinear least-squares problems

NIM Gould, T Rees, JA Scott

November 2017

Submitted for publication in SIAM Journal on Scientific Computing

RAL Library
STFC Rutherford Appleton Laboratory
R61
Harwell Oxford
Didcot
OX11 0QX

Tel: +44(0)1235 445384
Fax: +44(0)1235 446403
email: libraryral@stfc.ac.uk

ISSN 1361- 4762

A HIGHER ORDER METHOD FOR SOLVING NONLINEAR LEAST-SQUARES PROBLEMS*

NICHOLAS I.M. GOULD†, TYRONE REES†, AND JENNIFER A. SCOTT†‡

Abstract. We consider the solution of nonlinear least-squares problems. Such problems have traditionally been solved using a Gauss-Newton or Newton approximation, which is in turn globalized by either solving within a trust region, or by solving a regularized problem. We describe a hybrid method that combines these two approaches.

While Newton's method uses second derivative information, we argue that, due to to the sum-of-squares nature of the problem, a method we call the tensor-Newton approximation makes better use of this information. We show how the subproblem here can be solved using standard nonlinear least-squares techniques, e.g., our hybrid Gauss-Newton/Newton solver.

We describe each of these methods, and give numerical results from our nonlinear least-squares solver RALFit, comparing the proposed solver with the commonly used implementation in the GNU Scientific Library.

Key words. nonlinear least-squares, Levenberg Marquardt, Trust region methods, Data fitting

AMS subject classifications. 65K05, 90C30

1. Introduction. One of the central problems in computational mathematics is to fit a suitable model to observed data [2, 14, 36, 37]. Although other measures of fit are sometimes appropriate, most fits are performed in the Euclidean norm $\|\cdot\|$, or an appropriately weighted version thereof. In this paper, we consider the more general setting of minimizing $\|r(x)\|$, where $r : \mathbb{R}^n \longrightarrow \mathbb{R}^m$ is a smooth function of the parameters $x \in \mathbb{R}^n$, or equivalently of minimizing

$$\Phi(x) := \frac{1}{2}\|r(x)\|^2;$$

for simplicity, we tacitly assume that any weights have been absorbed in r.

We focus on methods that return a local solution to our minimization problem, and for this $\nabla_x\Phi(x) = J^T(x)r(x) = 0$, where the Jacobian $J(x) = \nabla_x r(x)$, provided that $r \in C^1$. In practice, we prefer the equivalent criticality conditions

$$\|r(x)\| = 0 \text{ or } \|J^T(x)r(x)\|/\|r(x)\| = 0 \tag{1}$$

for the function $\|r(x)\|$, since these can be applied within an iterative method to stop the kth iteration at a point x_k which satisfies

$$\|r(x_k)\| \leq \epsilon_r \text{ or } \|J^T(x_k)r(x_k)\|/\|r(x_k)\| \leq \epsilon_g \tag{2}$$

for appropriately chosen (small) ϵ_r, $\epsilon_g > 0$.

Although $\Phi(x)$ may be considered as a general objective function, it is usual to exploit its sum-of-squares nature. We consider iterative methods in which we seek an improvement $x_k + s_k$ to the current estimate x_k of a critical point of the objective function. There are many different ways of selecting an appropriate s_k, and each of these result in a different method.

---

*Submitted version

Funding: This work was funded by EPSRC Grant EP/M025179/1

†STFC Rutherford Appleton Laboratory, (nick.gould@stfc.ac.uk, tyrone.rees@stfc.ac.uk, jennifer.scott@stfc.ac.uk)

‡University of Reading.

1.1. Gauss-Newton based methods. The best-known approach is the Gauss Newton method. Here $r(x_k + s)$ is replaced by its first-order Taylor approximation, $r(x_k) + J(x_k)s$, and we choose

$$s_k = \arg\min_{s\in\mathbb{R}^n} m_k^L(s) := \frac{1}{2}\|r(x_k) + J(x_k)s\|^2.$$

We denote this model by $m_k^L(s)$, since this in a linear approximation to the residual $r(x_k + s)$. This search direction, when combined with a suitable line search strategy, is often enough to achieve convergence. However, in the case where $J(x_k)$ is rank deficient convergence cannot be guaranteed, as the linear least squares problem has multiple solutions.

The usual way to alleviate this issue is to switch from a line-search to a trust-region globalization strategy. This is either implemented by explicitly solving the minimization problem in a trust region, taking

$$s_k = \arg\min_{s\in\mathbb{R}^n,\ \|s\|\le\Delta_k} m_k^L(s) \tag{3}$$

for an appropriate trust-region radius $\Delta_k > 0$ or, equivalently [30], solving the regularized problem, taking

$$s_k = \arg\min_{s\in\mathbb{R}^n} m_k^{LR}(s) := m_k^L(s) + \frac{1}{2}\sigma_k\|s\|^2, \tag{4}$$

for an appropriate regularization weight $\sigma_k > 0$.[1] Here we have appended the label $m_k^L(s)$ with a R to denote the addition of the regularization term into the model. The parameter Δ_k or σ_k, as appropriate, is increased or decreased on subsequent iterations and the predictions accepted ($x_{k+1} = x_k + s_k$) or rejected ($x_{k+1} = x_k$) depending on the agreement of our model with the function $\Phi(x)$, and it is straightforward to devise methods of adjusting Δ_k and σ_k to ensure global convergence [30]. These are variants of the Levenberg Marquardt (LM) method[2] [25, 28, 33].

1.2. Newton's method. While we can ensure global convergence with Gauss-Newton based methods, the asymptotic rate may be linear unless $r(x_k)$ converges to zero [24]. Thus, an alternative when $r \in C^2$ is to consider $\Phi(x)$ as a general function, and to model $\Phi(x_k + s)$ by its 2nd-order Taylor approximation

$$m_k^T(s) := \Phi(x_k) + s^T\nabla_x\Phi(x_k) + \frac{1}{2}s^T\nabla_{xx}\Phi(x_k)s \equiv \frac{1}{2}\|r(x_k) + J(x_k)s\|^2 + \frac{1}{2}s^T B_k s,$$

where $B_k = \sum_{i=1}^m r_i(x_k)\nabla_{xx}r_i(x_k)$ [11, 18]. We denote this model by $m_k^T(s)$, since it is a Newton approximation of the cost function $\Phi(x + s)$.

Since Newton's method is only locally convergent, to make a robust method we require a globalization strategy. As with the Gauss-Newton approximation, this can be provided effectively by solving the minimization problem either within a trust-region [9],

$$s_k = \arg\min_{s\in\mathbb{R}^n,\ \|s\|\le\Delta_k} m_k^T(s), \tag{5}$$

[1] In practice, both the trust-region and regularization norms can, indeed should, be weighted to account for different variable scalings [30].

[2] See also [35] for other historical antecedents and precedents.

or by solving a regularized problem [6, 21],

$$s_k = \arg\min_{s\in\mathbb{R}^n} m_k^{TR}(s) := m_k^T(s) + \frac{1}{3}\sigma_k\|s\|^3. \tag{6}$$

If required, we could replace the exact matrix B_k with a surrogate that is easier to calculate than the true second derivative matrix. Asymptotic superlinear or quadratic convergence can be shown under suitable assumptions on the approximation B_k [6, §4.2]

1.3. Tensor-Newton methods. It is readily observed that $m_k^T(s)$, the quadratic approximation to $\Phi(x_k + s)$, may not necessarily be bounded from below. This is in contrast to the model $m_k^L(s)$, obtained from a first order approximation to $r(x_k + s)$. Thus, one may argue that $m_k^T(s)$ provides a less satisfactory model of a norm-based objective function. A more "natural" second order model is to replace each component $r_i(x_k + s)$ by a second-order Taylor approximant

$$t_i(x_k, s) := r_i(x_k) + s^T g_i(x_k) + \frac{1}{2}s^T H_i(x_k)s, \tag{7}$$

where $g_i(x) = \nabla_x r_i(x)$ is the i-th column of $J^T(x)$ and $H_i(x) = \nabla_{xx} r_i(x)$. Writing $\underline{\mathrm{H}}(x)$ as the $n \times n \times m$ tensor formed by stacking these $H_i(x)$, we use the notation $s^T(\underline{\mathrm{H}}(s) \bullet s)$ to denote the vector in $\mathbb{R}^m$ with $\left(s^T(\underline{\mathrm{H}}(x) \bullet s)\right)_i = s^T H_i(x)s$. Then a suitable model of $f(x_k + s)$ is

$$m_k^Q(s) := \frac{1}{2}\|r(x_k) + J(x_k)s + s^T(\underline{\mathrm{H}}(x_k) \bullet s)\|^2 \equiv \frac{1}{2}\sum_{i=1}^m t_i^2(x_k, s). \tag{8}$$

We highlight that this model has been denoted by $m_k^Q(s)$ because it is based on a quadratic model of $r(x_k + s)$. Note that $m_k^Q(s)$ is a quartic function of s, and unlike (3)–(6), it is unclear that there are efficient methods to find its global minimizer. Regardless, we might aim to solve the regularized problem

$$\min_{s\in\mathbb{R}^n} m_k^{QR}(s) := m_k^Q(s) + \frac{1}{p}\sigma_k\|s\|_p^p, \tag{9}$$

for some $p > 0$, but accept any value s_k for which

$$m_k^{QR}(s_k) < m_k^{QR}(0) \text{ and } \|\nabla_s m_k^{QR}(s_k)\| \le \epsilon_k \tag{10}$$

for some suitably small $\epsilon_k > 0$.

We will show that, crucially, the subproblem (9) can be formulated itself as a least-squares problem, and thus we may apply any of the model-based methods (3)–(6) to it. Although approximately solving (9) may require significant effort, it only requires the computation of the values and (approximate) derivatives of $r(x)$ at x_k. We give more details in Section 3.1.

Higher-order models for least-squares problems have previously appeared in the literature. Bouaricha and Schnabel [3, 4] are the first authors we are aware of to consider this. To ease the affect of small singular values of the Jacobian, they suggest using a quadratic model of the residual evaluated at $x_k + s$ that approximates the Hessians by a low-rank matrix. More recently, an approach known as geodesic acceleration has been developed [38, 39], which applies a correction to Gauss-Newton

like steps in a way that allows for higher-order derivatives. There has also been some work recently describing derivative-free methods that aim to build quadratic models of $r(x_k + s)$ [40,41], although these ultimately have the flavour of Gauss-Newton variants. While these methods use a quadratic model of the residual, and so can be seen to improve on Gauss-Newton based methods, none makes full use of the residual Hessians as proposed here.

1.4. Outline. In this paper, we consider all of the variants mentioned above. In Section 2, we consider how Gauss-Newton and Newton variants compare in practice. In particular, we describe a software package, `RALFit`, that offers a variety of Gauss-Newton/Newton hybrids for problems involving a modest number of variables. We indicate that the new package performs favourably with the widely-used GSL/Minpack nonlinear least-squares software [17,32]. In Section 3, we turn to methods based on the tensor-Newton model (9), and describe how this is implemented in `RALFit`.

2. Gauss Newton, Newton and hybrids. In this section we consider the Gauss-Newton approach (which is the basis of the celebrated LM algorithm) and the Newton approach.

2.1. Gauss-Newton. First, consider the subproblem to be solved when using a Gauss-Newton approach with regularization. It is easy to see that the solution of (4) is given by solving the linear system

$$(J(x_k)^T J(x_k) + \sigma_k I)s_k = -J(x_k)^T r(x_k). \tag{11}$$

The alternative is to solve the trust-region subproblem (3). This is (theoretically) equivalent to using regularization (4), in that there is some λ such that the solution to (3) is equivalent to solving

$$(J(x_k)^T J(x_k) + \lambda I)s_k = -J(x_k)^T r(x_k) \tag{12}$$

$$\lambda(\Delta - \|s_k\|) = 0. \tag{13}$$

This tells us that either the solution lies on the boundary of the trust region, or λ vanishes, i.e., $\sigma_k = \lambda$ or $\sigma_k = 0$ respectively in (11). We can therefore think of a trust-region method as a form of regularization, where the regularization parameter σ_k is updated in a nonlinear way. For example, solving the problem within a trust-region switches off regularization when sufficiently close to a solution, something that is desirable (but hard to detect) when using LM.

The solution of the trust-region subproblem (12)-(13) has been well studied, both in the dense and the sparse cases and, although there are still open questions about the best way to do this (particularly in the sparse case), we will treat this as a problem that can be solved with existing algorithms. In particular, this may be achieved by either matrix factorization or using a suitable preconditioned iterative method [23,30]. Alternatively, specialized iterative methods may be applied directly to (3)-(4) [5]. A derivative-free Gauss-Newton method that solves the model in a trust region has recently been developed [7].

2.2. Newton. The Newton trust-region subproblem (5) is solved similarly. It can be shown that there is some λ such that

$$\left(J(x_k)^T J(x_k) + B_k + \lambda I\right) s_k = -J(x_k)^T r(x_k), \tag{14}$$

$$\lambda(\Delta - \|s_k\|) = 0. \tag{15}$$

Dealing with the cubic term in the regularization parameter in (6) is slightly more complicated than what is needed in the Gauss-Newton case, but this subproblem can be efficiently solved by using techniques borrowed from solving the trust-region subproblem [6, 23]

2.3. A hybrid algorithm. It is frequently observed that Gauss-Newton tends to work well far away from the solution, whereas Newton is provably superior sufficiently close to the solution (especially for problems with a large residual at the solution). A discussion of this phenomenon is given, for example, by Chen [8].

A way around this is to use a hybrid method; see, for example, Madsen [27]. This method starts by using Gauss-Newton and switches to Newton if, for n_s consecutive iterations,

$$\|J(x_k)^T r(x_k)\|_2 \leq \epsilon_h \frac{1}{2}\|r(x_k)\|^2,$$

where n_s and ϵ_h are pre-defined parameters. Experiments suggest that taking $n_s = 1$ and $\epsilon_h = 2$ gave good results, and we use these values in our tests below. Furthermore, if on a subsequent iteration

$$\frac{1}{2}\|r(x_k + s_k)\|^2 > \frac{1}{2}\|r(x_k)\|^2,$$

then we interpret this as being because we were not sufficiently close to the solution for the superlinear convergence of Newton's method to be apparent. In this case, the hybrid algorithm switches back to using the Gauss-Newton model.

The exact second derivatives may not be available and in this case we can often do better than Gauss-Newton by approximating the missing second-order term B_k. Common approaches to this include the method based on BFGS by Fletcher and Xu [16], [15, Chapter 6], or the update procedure based on the secant method described by Dennis, Gay and Welsch [12], both of which are hybrid schemes that switch to Gauss-Newton when the problem has a small residual.

2.4. Numerical Results. We now compare three different algorithms on a set of 101 test problems. These examples are all available through `CUTEst` [20], and comprise 27 problems from the NIST StRD nonlinear regression data sets, started from both 'hard' and 'easy' initial points [10] (giving us 54 test problems in total), 26 problems from the Moré-Garbow-Hillstrom test set [31], 8 problems from the ISIS synchotron at the Rutherford Appleton Laboratory [1], 9 problems described by Luksan [26], and 4 problems from a model of brain tissue [29]. The problems, and their dimensions, are listed in Table 1.

Our implementation of Gauss-Newton and the hybrid method described in Section 2.3, both using a trust-region globalisation strategy, is part of our RALFit software package. RALFit is written in modern Fortran, and is available (along with the code to run these tests) on GitHub. We compare the performance of the implementation of these two solvers in RALFit against the implementation of LM in the GNU Scientific Library (GSL). We use the stopping test (2) for both methods, with $\epsilon_r = \max(10^{-5}, 10^{-8}\|r(x_0)\|)$ and $\epsilon_g = \max(10^{-5}, 10^{-8}\|J^T(x_0)r(x_0)\|/\|r(x_0)\|)$. We set the maximum number of iterations to 5000. Here, and in the following, we call a test a failure if either the solver returns an error code, or the maximum number of iterations has been reached.

The tests were run on a desktop PC with Intel Core i7 CPU (3.10GHz) with 8GB of memory, running Linux Mint 17.2 Rafaela. Figure 1 shows a performance profile [13] comparing function evaluations. Here we see that GSL fails to find the

Problem	Source	n	m	Problem	Source	n	m
ARGAUSS	M	3	15	LANCZOS3	N	6	24
ARGTRIG	M	10	10	LUKSAN11	L	100	198
BARDNE	M	3	15	LUKSAN12	L	98	192
BENNETT5	N	3	154	LUKSAN13	L	98	224
BIGGS6NE	M	6	13	LUKSAN14	L	98	224
BOX3NE	M	3	10	LUKSAN15	L	100	196
BOXBOD	N	2	6	LUKSAN16	L	100	196
BROWNALE	M	10	10	LUKSAN17	L	100	196
BROYDN3D	M	10	10	LUKSAN21	L	100	100
BROYDNBD	M	10	10	LUKSAN22	L	100	198
CERI651A	I	7	61	MEYER3NE	M	3	16
CERI651B	I	7	66	MGH09	N	4	11
CERI651C	I	7	56	MGH10	N	3	16
CERI651D	I	7	67	MGH17	N	5	33
CERI651E	I	7	64	MISRA1A	N	2	14
CHWIRUT1	N	3	214	MISRA1B	N	2	14
CHWIRUT2	N	3	54	MISRA1C	N	2	14
DANWOOD	N	2	6	MISRA1D	N	2	14
ECKERLE4	N	3	35	MOREBVNE	M	10	10
ENSO	N	9	168	NELSON	N	3	128
FBRAIN	B	2	2211	OSBORNE1	M	5	33
FBRAIN2	B	4	2211	OSBORNE2	M	11	65
FBRAIN3	B	6	2211	PENLT1NE	M	10	11
FREURONE	M	2	2	PENLT2NE	M	4	8
GAUSS1	N	8	250	POWELLBS	M	2	2
GAUSS2	N	8	250	POWELLSE	M	4	4
GAUSS3	N	8	250	RAT42	N	3	9
GBRAIN	B	2	2200	RAT43	N	4	15
GULFNE	M	3	99	ROSZMAN1	N	4	25
HAHN1	N	7	236	RSNBRNE	M	2	2
HELIXNE	M	3	3	THURBER	N	7	37
INTEQNE	M	12	12	VARDIMNE	M	10	12
JENSMPNE	M	2	10	VESUVIA	I	8	1025
KIRBY2	N	5	151	VESUVIO	I	8	1025
KOWOSBNE	M	4	11	VESUVIOU	I	8	1025
LANCZOS1	N	6	24	WATSONNE	M	12	31
LANCZOS2	N	6	24	WOODSNE	M	4	7

Table 1: Problems in the test set. Sources are:
B Mihai, *et al.* [29]; **I** ISIS problems; **L** Luksan [26]; **M** Moré-Garbow-Hillstrom [31]; **N** NIST test set [10]

solution within 5000 iterations nine times on this test set (returning an error code five of those times, and taking the maximum number of iterations for the remaining four), whereas our implementation of the Gauss-Newton method failed to converge within 5000 iterations for seven problems (with two of these returning errors), and the hybrid Gauss-Newton/Newton method only failed to converge four times (again

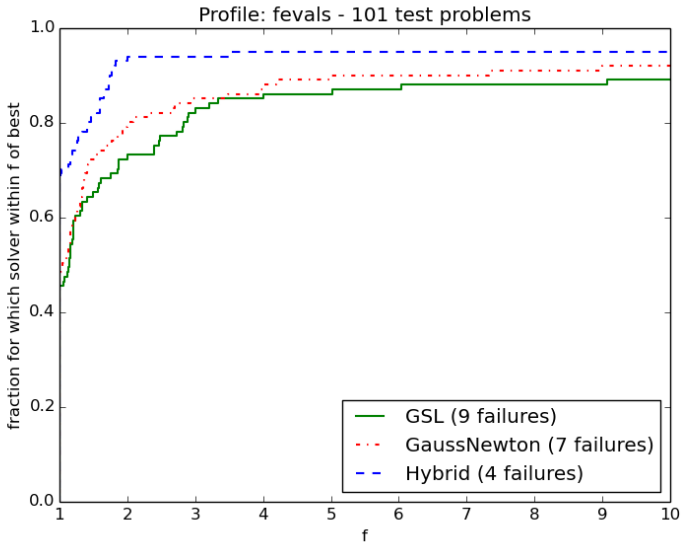

Fig. 1: Performance profile: function evaluations

with two of these returning error codes).

The maximum number of iterations was reached before any of the codes could solve `CERI651B`, and `CERI651E` also failed to be solved by any of the methods tested, with both Gauss Newton and hybrid methods returning an error message and GSL taking the maximum number of iterations. Only the hybrid method could solve `CERI651D` (with GaussNewton and GSL taking the maximum number of iterations) and `VESUVIA` (where both GaussNewton and GSL returned error codes). By contrast, GSL was the only method that could solve `PENLT1NE` and `LUKSAN12`, with GaussNewton and Hybrid not converging before 5000 iterations. GaussNewton returned an error code when solving `LUKSAN17`, which the others had no difficulty with. Finally, GSL was the only method that had trouble with `MGH17 (SP2)`, `NELSON (SP2)`, `VESUVIOU`, `CERI65A`, and `LUKSAN15`, terminating early with an error code in all but `CERI651A`, which reached 5000 iterations without convergence. These findings are summarised in Table 2.

The time performance profile in Figure 2 shows that Gauss Newton and particularly the hybrid method compare favourably in terms of wall clock time with GSL. This, together with the evaluation performance profile in Figure 1, suggests that including the second derivative information in the hybrid method leads to an improvement.

The results show that we get better performance, both in terms of the time taken and the number of solvable problems, by using the Hybrid method, which uses information from the Hessian by switching to Newton's method when close to the solution. However, there are still a number of problems that are not solved even when allowing a Newton step. We contend that the fundamental issue is that we are using the higher order derivatives to generate a model of the cost function, $\mu_k(s)$, while such a model may not reflect its true structure. If, instead, we use information from the Hessian to improve our model for *the residual*, then we can build up a higher

Problem	GSL	Gauss Newton	Hybrid
CERI651B	★	★	★
CERI651E	★	×	×
CERI651D	★	★	✓
VESUVIA	×	×	✓
PENLT1NE	✓	★	★
LUKSAN12	✓	★	★
LUKSAN17	✓	×	✓
MGH17 (SP2)	×	✓	✓
NELSON (SP2)	×	✓	✓
VESUVIOU	×	✓	✓
CERI65A	★	✓	✓
LUKSAN15	×	✓	✓

Table 2: Troublesome problems.
★ : 5000 iterations reached without convergence
× : error message returned before convergence
✓ : iteration converged successfully

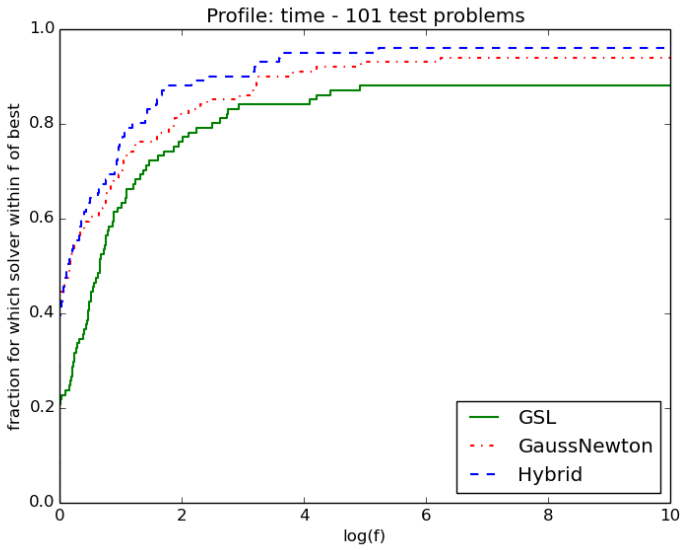

Fig. 2: Performance profile: time

order model of the cost function, $m_k^{QR}(s)$, and one that is furthermore always a least-squares problem. This will therefore likely be a more accurate model globally, and should give us a more robust method; we explore this idea further in the following section.

3. Tensor-Newton methods. In this section we consider tensor-Newton methods, which are based on the model $m_k^{QR}(s)$ from (9). We assume throughout that $r(x) \in C^2$, and that second derivatives are available. We note that much of what we

say also applies when $H_i(x_k)$ is replaced by a suitable (sparse, if necessary) approximation, B_{ik}.

3.1. Solving the subproblem. The subproblem (9) needs to be solved at each step of the iteration. For expository purposes, first consider the unregularized subproblem

$$\min_{s\in\mathbb{R}^n} m_k^Q(s) = \frac{1}{2}\|t(x_k,s)\|^2, \tag{16}$$

where $t(x_k,s)$ is the $m-$vector with entries as given by (7). As we will see in Sections 3.1.1 and 3.1.2, the method of solving (16) can be adapted as in (9) to incorporate regularization in a straightforward way.

Crucially, note that the problem (16) is itself a nonlinear least-squares problem, and so can be solved using any of methods from Section 2. These require access to the Jacobian and Hessian of the function $t(x_k,s)$. From the definition (7), it is easy to see that the Jacobian $\nabla_s t(x_k,s) =: A(x_k,s)$ has columns

$$\nabla_s t_i(x_k,s) = g_i(x_k) + H_i(x_k)s, \tag{17}$$

while the weighted Hessian

$$\sum_{i=1}^m t_i(x_k,s)\nabla_{ss}t_i(x_k,s) = \sum_{i=1}^m t_i(x_k,s)H_i(x_k) =: W(x_k,s). \tag{18}$$

Thus, unsurprisingly since we are considering second-order approximations, the values $t_i(x_k,s)$ and gradients $\nabla_s t_i(x_k,s)$ require access to $H_i(x_k)$ in some form.

Observe that evaluating the Jacobian and Hessian of $t_i(x_k,s)$ in the subproblem does not require re-evaluating the Jacobian and Hessian of the original function $r(x)$ at a different point, but simply re-using the data from the point x_k in matrix-vector products. We highlight this as a fundamental feature of the tensor-Newton method. Typically, evaluating $J(x_k)$ and (especially) $\nabla_{xx}r_i(x_k)$ are the most expensive operations per iteration, and this algorithm allows us to get more use out of these operations, reducing the number of points at which these need to be evaluated. For problems involving a small number of variables, it is usually feasible to calculate and store the Jacobian $J(x_k)$ and m Hessians $\nabla_{xx}r_i(x_k)$ (or approximations thereof) as dense matrices, and thus to calculate $t_i(x_k,s)$ and its required derivatives. For large problems, where we must take into account sparsity, we may take advantage of the fact that only matrix-vector products with these matrices are required.

3.1.1. Regularization where $p = 2$. Here we describe how to solve the subproblem (9) when $p = 2$. Note that, in this case, the function $m_k^{QR}(s)$ is itself a sum-of-squares, since we can write

$$m_k^{QR}(s) = \frac{1}{2}\left(\sum_{i=1}^m t_i^2(x_k,s) + \sum_{j=1}^n (\sqrt{\sigma_k}s_i)^2\right) = \frac{1}{2}\sum_{i=1}^{n+m}\widehat{t}^{\,2}(x_k,s),$$

where the function $\widehat{t}$ satisfies

$$\widehat{t_i}(x_k,s) = \begin{cases} t_i(x_k,s) & 1\le i\le m,\\ \sqrt{\sigma}s_i & m+1\le i\le n+m.\end{cases}$$

This problem can now tackled by any of the methods from Section 2. In addition to the function values, we need a Jacobian and some more information about the Hessian. For our modified function, the Jacobian, $\widehat{A}(x_k, s)$, is

$$\widehat{A}(x_k, s) = \begin{bmatrix} A(x_k, s) \\ \sqrt{\sigma} I \end{bmatrix},$$

and the weighted Hessian, $\widehat{W}(x_k, s)$, is given by

$$\widehat{W}(x_k, s) = \sum_{i=1}^{n+m} \widehat{t}_i(x_k, s)\, \nabla_{ss} \widehat{t}_i(x_k, s) = \sum_{i=1}^{n} t_i(x_k, s)\, \nabla_{ss} t_i(x_k, s) = W(x_k, s).$$

3.1.2. General regularization powers. For a more general p, we cannot write (9) as a sum of squares in such a simple way. However, since the regularization term $\frac{\sigma_k}{p}\|s\|_2^p$ is non-negative, we can write

$$m_k^{QR}(s) = \frac{1}{2}\left(\|t(x_k, s)\|^2 + \left(\left(\frac{2\sigma}{p} \right)^{1/2} \|s\|^{p/2} \right)^2 \right),$$

thereby defining a new nonlinear least squares problem involving the function $\bar{t}$ such that

$$\bar{t}_i(s) = \begin{cases} t_i(x_k, s) & 1 \le i \le m, \\ \frac{2\sigma}{p}\|s\|^{p/2} & i = m+1. \end{cases}$$

The Jacobian, $\bar{A}(x_k, s)$, for this new function is given by

$$\bar{A}(x_k, s) = \begin{bmatrix} A(x_k, s) \\ \left(\frac{\sigma p}{2}\right)^{1/2} \|s\|^{(p-4)/2} s^T \end{bmatrix},$$

and the second derivatives are given by

$$\nabla_{ss} \bar{t}_{m+1} = \left(\frac{\sigma p}{2} \right)^{1/2} \|s\|^{(p-4)/2} \left(I + \frac{s\, s^T}{\|s\|^2} \right).$$

3.2. The algorithm. A summary of the essential features of the algorithm is given below as Algorithm 1. Typical values of the algorithm's parameters might be $\sigma_0 = 100$, $\sigma_{\min} = 10^{-16}$, $\theta = 1$, $\gamma_1 = 10^{-2}$, $\gamma_2 = 1$, $\gamma_3 = 2$, $\eta_1 = 10^{-8}$, $\eta_2 = 0.9$ and $\epsilon_r = \epsilon_g = 10^{-6}$.

The computation of s_k is well defined as $m_k^Q(s)$ is bounded from below by zero, and $m_k^{QR}(s)$ grows to infinity as $\|s\|_p$ grows since $\sigma_k > 0$, and hence has a minimizer. We may find s_k by, for example, applying the methods in Sections 3.1.1 or 3.1.2, as appropriate. More sophisticated weight updates than suggested in Algorithm 1 are also possible [21].

3.3. Convergence. Details of the convergence of this algorithm are given in [22], but for completeness we summarise the main findings here. We make the following assumption:

ASSUMPTION 1. *Each component $r_i(x)$ and its first two derivatives are Lipschitz continuous on an open set containing the intervals $[x_k, x_k + s_k]$ generated by Algorithm 1.*

Algorithm 1 Adaptive regularization using a tensor-Newton model for nonlinear least-squares problems

Input: x_0, $\sigma_0 \geq \sigma_{\min} > 0$, ϵ_r, $\epsilon_g > 0$.
Set parameters $\theta > 0$, $\gamma_3 \geq \gamma_2 > 1 > \gamma_1 > 0$ and $1 > \eta_2 \geq \eta_1 > 0$
Evaluate $r(x_0)$, and hence $f(x_0)$.
for $k = 0, 1, \ldots,$ **do**
  Evaluate $J(x_k) := \nabla_x r(x_k)$, and hence $\nabla_x f(x_k)$.
  if $\|r(x_k)\| \leq \epsilon_r$ or $\frac{\|J^T(x_k) r(x_k)\|}{\|r(x_k)\|} \leq \epsilon_g$ **then**
    Converged: terminate algorithm
  end if
  Compute second derivatives of $r(x)$ at x_k.
  Compute a step s_k by approximately minimizing $m_k^{QR}(s)$ so that $m_k^{QR}(s_k) < m_k^{QR}(0)$ and $\|\nabla_s m_k^{QR}(s_k)\| \leq \theta \|s_k\|^{p-1}$
  Compute $r(x_k + s_k)$, and hence $f(x_k + s_k)$
  Set $\rho_k = \frac{f(x_k) - f(x_k + s_k)}{m_k^Q(0) - m_k^Q(s_k)}$.
  if $\rho_k \geq \eta_1$ **then**
    $x_{k+1} = x_k + s_k$
  else
    $x_{k+1} = x_k$
  end if
  if $\rho_k \geq \eta_2$ **then**
    Set $\sigma_{k+1} \in [\max(\sigma_{\min}, \gamma_1 \sigma_k), \sigma_k]$ *(very successful iteration)*
  else if $\eta_1 \leq \rho_k < \eta_2$ **then**
    Set $\sigma_{k+1} \in [\sigma_k, \gamma_2 \sigma_k]$ *(successful iteration)*
  else
    Set $\sigma_{k+1} \in [\gamma_2 \sigma_k, \gamma_3 \sigma_k]$ *(unsuccessful iteration)*
  end if
end for

In the case were $2 \leq p \leq 3$, we can show that the algorithm converges to a first-order critical point of $\Phi(x)$.

THEOREM 2 (Theorem 3.9, Gould, Rees and Scott [22]). *Suppose that Assumption 1 holds and $2 \leq p \leq 3$. Then the iterates $\{x_k\}$ generated by Algorithm 1 satisfy*

$$\lim_{k \to \infty} \inf \|\nabla_x \Phi(x_k)\| = 0$$

if no non-trivial termination test is provided, i.e., unless (1) is satisfied.

Using the stopping criterion (2) within Algorithm 1, we can also provide an evaluation complexity result for when $2 < p \leq 3$.

THEOREM 3 (Theorem 3.12, Gould Rees and Scott [22]). *Suppose that Assumption 1 holds, $2 < p \leq 3$ and that the integer*

$$i \geq i_0 := \left\lceil \log_2 \left(\frac{p-1}{p-2} \right) \right\rceil$$

is given. Then Algorithm 1 requires at most

$$\left\lceil \kappa_u \max \left(\kappa_c^{-1}, \kappa_g^{-1} \epsilon_g^{-p/(p-1)}, \kappa_r^{-1} \epsilon_r^{-1/2^i} \right) \right\rceil + \kappa_s + 1$$

evaluations of $r(x)$ and its derivatives to find an iterate x_k for which the termination test (2) is satisfied for given $\epsilon_r > 0$ and $\epsilon_g > 0$, where κ_u, κ_s, κ_c, κ_g and κ_r are constants that depend on the properties of $r(x)$, and $\beta \in (0,1)$ is a fixed problem-independent constant.

If $i < i_0$, then we can also show a weaker bound, which includes the case of $p = 2$, of $\mathcal{O}\left(\max(1, \epsilon_g^{p/(p-1)} \cdot \epsilon_r^{\left(p/(p-1)-(2^{i+1}-1)/2^i\right)}, \epsilon_r^{1/2^i})\right)$. Furthermore, in the case where $p > 3$, convergence and complexity results have been shown with minor adjustments to algorithm; for details see [22, Section 4].

3.4. Results. Here we test four methods: GSL, the hybrid method from Section 2.3, and two tensor-Newton methods, one with $p = 2$ and the other with $p = 3$ (using the methods in Sections 3.1.1 and 3.1.2, respectively). We run this on the same test set as in Section 2.4. Our findings are presented in Figures 3 and 4.

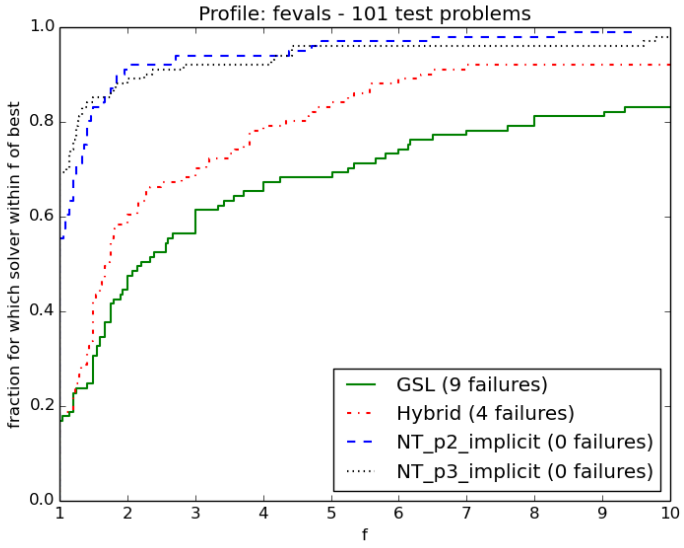

Fig. 3: Performance profile: function evaluations

We highlight the fact that both variants of the tensor-Newton method are more robust than the other schemes: whereas GSL failed to solve 9 problems within 5000 iterations, and the hybrid method failed to solve 4, both the tensor-Newton methods returned a local minimum for all problems in the test. In addition to being more robust, it is clear from Figure 3 that the tensor-Newton models require fewer function evaluations on average (although it is less clear whether it is better to choose $p = 2$ or $p = 3$). The tensor-Newton methods are, however, more expensive. Since the cost per iteration is higher, fewer function evaluations need not mean a quicker solution time, with the main exception being problems where function evaluation times dominate. Figure 4 shows a time performance profile.

4. Conclusions. We have described the tensor-Newton method, an approach for solving nonlinear least-squares problems that uses second derivative information to improve our approximation to the residual, hence leading to a higher order model

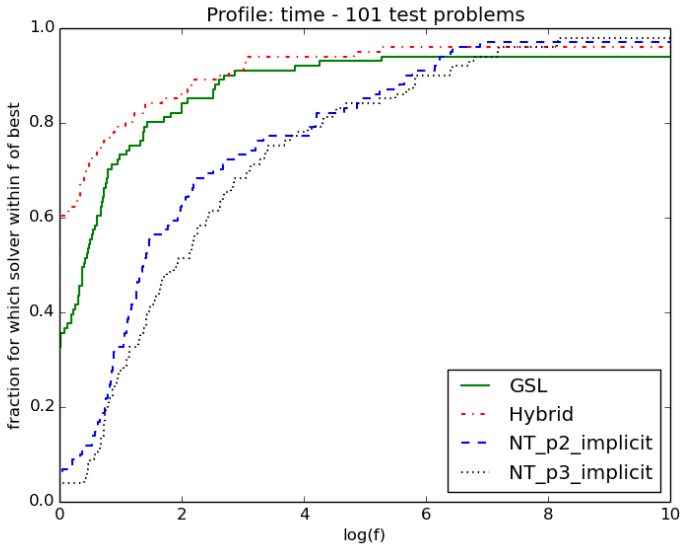

Fig. 4: Performance profile: time

of the cost function. We argue that this will lead to a more robust numerical method for solving such problems, and our numerical tests confirm this.

The subproblem that needs to be solved at each iteration in the tensor-Newton method is itself a nonlinear least-squares problem. We solve this using a hybrid Gauss-Newton/Newton method. This method, described in Section 2, has proved to be faster and more robust than the state-of-the-art nonlinear least squares solving routine in GSL. Although the tensor-Newton method is more robust, it is generally slower, so the Hybrid method may be preferable as a general purpose solver, with the tensor-Newton method used for the most challenging problems. This effect is less obvious when the function evaluation is expensive, and a prototype implementation of the method (GALAHAD_NLS) for large, sparse problems within GALAHAD [19] suggests that this is so.

The methods we have introduced here have been made available in the open-source package RALFit, which is available to download from GitHub [34].

REFERENCES

[1] Website of the ISIS facility. https://www.isis.stfc.ac.uk. Accessed: 2017-10-11.
[2] Å. Björck. *Numerical Methods for Least Squares Problems*. SIAM, Philadelphia, USA, 1996.
[3] A. Bouaricha and R. B. Schnabel. Algorithm 768: TENSOLVE: a software package for solving systems of nonlinear equations and nonlinear least-squares problems. *ACM Transactions on Mathematical Software*, 23(2):174–195, 1997.
[4] A. Bouaricha and R. B. Schnabel. Tensor methods for large, sparse nonlinear least squares problems. *SIAM Journal on Scientific and Statistical Computing*, 21(4):1199–1221, 1999.
[5] C. Cartis, N. I. M. Gould, and Ph. L. Toint. Trust-region and other regularisations of linear least-squares problems. *BIT*, 49(1):21–53, 2009.
[6] C. Cartis, N. I. M. Gould, and Ph. L. Toint. Adaptive cubic regularisation methods for unconstrained optimization. Part I: motivation, convergence and numerical results. *Mathematical Programming, Series A*, 127(2):245–295, 2011.

[7] C. Cartis and L. Roberts. A derivative-free Gauss-Newton model. arXiV.1710.11005, 2017.
[8] P. Chen. Hessian matrix vs. Gauss–Newton Hessian matrix. *SIAM Journal on Numerical Analysis*, 49(4):1417–1435, 2011.
[9] A. R. Conn, N. I. M. Gould, and Ph. L. Toint. *Trust-Region Methods*. SIAM, Philadelphia, 2000.
[10] NIST Nonlinear Regression Datasets. http://www.itl.nist.gov/div898/strd/nls/nls_main.shtml, (accessed March 2017).
[11] J. E. Dennis, D. M. Gay, and R. E. Welsh. An adaptive nonlinear least squares algorithm. *ACM Transactions on Mathematical Software*, 7(3):348–368, 1981.
[12] J. E. Dennis Jr, D. M Gay, and R. E. Walsh. An adaptive nonlinear least-squares algorithm. *ACM Transactions on Mathematical Software (TOMS)*, 7(3):348–368, 1981.
[13] E. D. Dolan and J. J. Moré. Benchmarking optimization software with performance profiles. *Mathematical programming*, 91(2):201–213, 2002.
[14] R. W. Farebrother. *Fitting Linear Relationships: A History of the Calculus of Observations 1750–1900*. Springer Verlag, Heidelberg, Berlin, New York, 1999.
[15] R. Fletcher. *Practical methods of optimization*. John Wiley & Sons, 2013.
[16] R. Fletcher and C. Xu. Hybrid methods for nonlinear least squares. pages 371,389, 1987.
[17] M. Galassi, J. Davies, J. Theiler, B. Gough, G. Jungman, P. Alken, M. Booth, F. Rossi, and R. Ulerich. GNU Scientific Library reference manual, Edition 2.1. http://www.gnu.org/software/gsl/, January 2015.
[18] P. E. Gill and W. Murray. Algorithms for the solution of the nonlinear least squares problem. *SIAM Journal on Numerical Analysis*, 15(5):977–992, 1978.
[19] N. I. M. Gould, D. Orban, and Ph. L. Toint. GALAHAD, a library of thread-safe Fortran 90 packages for large-scale nonlinear optimization. *ACM Transactions on Mathematical Software (TOMS)*, 29(4):353–372, 2003.
[20] N. I. M. Gould, D. Orban, and Ph. L. Toint. CUTEst: a constrained and unconstrained testing environment with safe threads. Technical Report RAL-TR-2013-005, STFC Rutherford Appleton Laboratory, 2013.
[21] N. I. M. Gould, M. Porcelli, and Ph. L. Toint. Updating the regularization parameter in the adaptive cubic regularization algorithm. *Computational Optimization and Applications*, 53(1):1–22, 2012.
[22] N. I. M. Gould, T. Rees, and J. A. Scott. Convergence and evaluation-complexity analysis of a regularized tensor-Newton method for solving nonlinear least-squares problems. Technical Report RAL-P-2017-009, STFC Rutherford Appleton Laboratory, 2017.
[23] N. I. M. Gould, D. P. Robinson, and H. S. Thorne. On solving trust-region and other regularised subproblems in optimization. *Mathematical Programming Computation*, 2(1):21–57, 2010.
[24] S. Gratton, A. S. Lawless, and N. K. Nichols. Approximate Gauss-Newton methods for nonlinear least squares problems. *SIAM Journal on Optimization*, 18(1):106–132, 2007.
[25] K. Levenberg. A method for the solution of certain problems in least squares. *Quarterly of Applied Mathematics*, 2(2):164–168, 1944.
[26] L. Lukšan. Hybrid methods for large sparse nonlinear least squares. *Journal of Optimization Theory and Applications*, 89(3):575–595, 1996.
[27] K. Madsen. A combined Gauss-Newton and Quasi-Newton method for non-linear least squares. 1988.
[28] D. Marquardt. An algorithm for least-squares estimation of nonlinear parameters. *SIAM Journal on Applied Mathematics*, 11(2):431–441, 1963.
[29] L. A. Mihai, S. Budday, G. A. Holzapfel, E. Kuhl, and A. Goriely. A family of hyperelastic models for human brain tissue. *Journal of the Mechanics and Physics of Solids*, 2017.
[30] J. J. Moré. The Levenberg-Marquardt algorithm: implementation and theory. In G. A. Watson, editor, *Numerical Analysis, Dundee 1977*, number 630 in Lecture Notes in Mathematics, pages 105–116, Heidelberg, Berlin, New York, 1978. Springer Verlag.
[31] J. J. Moré, B. S. Garbow, and K. E. Hillstrom. Testing unconstrained optimization software. *ACM Transactions on Mathematical Software (TOMS)*, 7(1):17–41, 1981.
[32] J. J. Moré, D. C. Sorensen, K. E. Hillstrom, and B. S. Garbow. The MINPACK project. In W. J. Cowell, editor, *Sources and Development of Mathematical Software*, Englewood Cliffs, New Jersey, 1984. Prentice-Hall.
[33] D. D. Morrison. Methods for nonlinear least squares problems and convergence proofs. In J. Lorell and F. Yagi, editors, *Proceedings of the Seminar on Tracking Programs and Orbit Determination*, pages 1–9, Pasadena, USA, 1960. Jet Propulsion Laboratory.
[34] RALFit: A nonlinear least-squares solver. https://github.com/ralna/RALFit, (accessed October 2017).
[35] J. Pujol. The solution of nonlinear inverse problems and the Levenberg-Marquardt method.

Geophysics, 72(4):W1–W16, 2007.

[36] G. A. F. Seber and C. J. Wild. *Nonlinear regression*. J. Wiley and Sons, Chicester and New York, 1989.

[37] T. Strutz. *Data Fitting and Uncertainty (A practical introduction to weighted least squares and beyond)*. Springer Verlag, Heidelberg, Berlin, New York, 2nd edition, 2016.

[38] M. K. Transtrum, B. B. Machta, , and J. P. Sethna. Why are nonlinear fits to data so challenging? *Physical Review Letters*, 104(6):060201, 2010.

[39] M. K. Transtrum and J. P. Sethna. Geodesic acceleration and the small-curvature approximation for nonlinear least squares. arXiV.1207.4999, 2012.

[40] H. Zhang and A. R. Conn. On the local convergence of a derivative-free algorithm for least-squares minimization. *Computational Optimization and Applications*, 51(2):481–507, 2012.

[41] H. Zhang, A. R. Conn, and K. Scheinberg. A derivative-free algorithm for least-squares minimization. *SIAM Journal on Optimization*, 20(6):3555–3576, 2010.