

A REGULARIZED GAUSS–NEWTON TRUST REGION APPROACH TO IMAGING IN DIFFUSE OPTICAL TOMOGRAPHY*

ERIC DE STURLER[†] AND MISHA E. KILMER[‡]

Abstract. We present a new algorithm for the solution of nonlinear least squares problems arising from parameterized imaging problems with diffuse optical tomographic data [D. Boas et al., *IEEE Signal Process. Mag.*, 18 (2001), pp. 57–75]. The parameterization arises from the use of parametric level sets for regularization [M. E. Kilmer et al., *Proc. SPIE*, 5559 (2004), pp. 381–391], [A. Aghasi, M. E. Kilmer, and E. L. Miller, *SIAM J. Imaging Sci.*, 4 (2011), pp. 618–650]. Such problems lead to Jacobians that have relatively few columns, a relatively modest number of rows, and are ill-conditioned. Moreover, such problems have function and Jacobian evaluations that are computationally expensive. Our optimization algorithm is appropriate for any inverse or imaging problem with those characteristics. In fact, we expect our algorithm to be effective for more general problems with ill-conditioned Jacobians. The algorithm aims to minimize the total number of function and Jacobian evaluations by analyzing which spectral components of the Gauss–Newton direction should be discarded or damped. The analysis considers for each component the reduction of the objective function and the contribution to the search direction, restricting the computed search direction to be within a trust region. The result is a truncated SVD-like approach to choosing the search direction. However, we do *not necessarily* discard components in order of decreasing singular value, and some components may be scaled to maintain fidelity to the trust region model. Our algorithm uses the Basic Trust Region Algorithm from [A. R. Conn, N. I. M. Gould, and Ph. L. Toint, *Trust-Region Methods*, SIAM, Philadelphia, 2000]. We prove that our algorithm is globally convergent to a critical point. Our numerical results show that the new algorithm generally outperforms competing methods applied to the DOT imaging problem with parametric level sets, and regularly does so by a significant factor.

Key words. nonlinear least squares, Gauss–Newton, Levenberg–Marquardt, optimization, regularization, diffuse optical tomography

AMS subject classifications. 65K05, 65F22, 90C30, 90C90

DOI. 10.1137/100798181

1. Introduction. To solve an inverse problem for a given system, we must compute the input that was responsible for generating a set of measured data. This solution is based on an assumed model of the relationship between the input and output of a system. Mathematically, this is represented as

$$(1.1) \quad y = h(p) + \eta,$$

where the vector p denotes the input, y is the output *data* vector, and η is the unknown noise in the measured data. In this paper, we are interested in models $h(p)$ that are nonlinear in p . In particular, we use regularization by parametric level sets [18, 1] (see section 2), which reduces the imaging problem to the nonlinear optimization of a regularized (well-posed) problem for the vector of nonlinear parameters p .

A typical approach to estimating p is to solve the nonlinear least squares problem

$$(1.2) \quad \min_p \frac{1}{2} \|W(h(p) - y)\|_2^2 = \min_p \frac{1}{2} r(p)^T r(p) = \min_p F(p),$$

*Received by the editors June 10, 2010; accepted for publication (in revised form) August 17, 2011; published electronically October 27, 2011.

<http://www.siam.org/journals/sisc/33-5/79818.html>

[†]Mathematics Department, Virginia Tech, Blacksburg, VA 24061 (sturler@vt.edu). This author's research was supported by NSF grant DMR-0325939.

[‡]Department of Mathematics, Tufts University, Medford, MA 20155 (misha.kilmer@tufts.edu). This author's research was supported by NSF grants 0139968 and 0342559.

with $r(p) = Wh(p) - Wy$, using an iterative nonlinear solver. As the objective function contains noise from measured data, we use a discrepancy principle-based stopping criterion to avoid wasting computational effort trying to resolve the noise. The discrepancy principle states that one should stop iterating when the norm of the residual reaches the norm of the (weighted) noise vector [17]. Here, W denotes a diagonal weighting matrix. The purpose of W is to scale the noise such that $W\eta$ is white, that is, the power spectrum of the vector $W\eta$ is flat.

As our regularization and parameterization approach has been introduced and discussed in [18, 1], in this paper, we focus on the resulting optimization (1.2). We briefly describe the parameterized diffuse optical tomographic (DOT) models in section 2. We focus on solving problems of the form (1.2) where the unknown p is of low to modest dimension, the Jacobian $J(p)$ for $r(p)$ is of modest size but modestly to severely ill-conditioned, evaluating $h(p)$ and/or $J(p)$ is computationally expensive, and computing the Hessian is either computationally intractable or not worth the cost. Specifically, we concentrate on the problem of reconstructing images from DOT data, using parametric level sets for regularization, where these features are common. However, we stress that the nonlinear least squares method introduced here should be applicable to more general problems for which our assumptions are satisfied. It is important to notice that the use of parametric level sets leads to a well-posed optimization problem, which does not need further regularization in the optimization algorithm. This is in contrast to the use of conventional level set methods in image reconstruction, and similar inverse problems, where regularization is an important issue; see [13, 9, 5, 24, 27, 26, 25].

Several well-known nonlinear iterative methods can be applied to solve (1.2). The most popular are the Gauss–Newton (GN), damped Gauss–Newton (DGN), also known as Gauss–Newton with a line search, and the Levenberg–Marquardt (LM) methods [8]. Since we assume that computing the Hessian is infeasible or intractable, a standard Newton approach is not applicable here. Quasi-Newton methods are also possible candidates, as are so-called full Newton modifications to the standard Gauss–Newton approach [8, 22] in which an extra term is added to $J^T J$ prior to computing the search direction with either GN or LM.

In this paper, we illustrate why the standard GN, DGN, and LM methods tend to be inefficient for (1.2) under the given assumptions. To produce a more efficient algorithm, we combine a trust region approach with the ideas underlying the truncated singular value decomposition (TSVD) to compute a (more) effective solution update from the trust region subproblem (see section 4). The underlying idea to our approach is that the trust region algorithm deals effectively with the nonlinearity of the optimization, but it does not account for the ill-conditioning of the Jacobian and the noise in the residual. To deal with the ill-conditioning and the noise, we use a combination of the TSVD and a criterion to balance the relative importance of an SVD component to reducing the trust region model with its contribution to the length of the solution update. Although this is not its usual application, we use the generalized cross validation (GCV) method for this criterion. In [15], with a similar motivation, the GCV method is combined with DGN and Tikhonov regularization to solve the (original) ill-posed, nonlinear least squares problem. In [15], the GCV method is used to compute (update in each iteration) a global regularization parameter to deal with the noise, whereas the line search in DGN imposes a local regularization to deal with the nonlinearity of the model. In [13], in the same context, the use of the GCV method in combination with a trust region method is also briefly mentioned. However,

an important difference with our approach is that in [15, 13] the parameter computed by the GCV is necessary for (Tikhonov) regularization, whereas in our approach the nonlinear problem is already regularized. So, neither the TSVD nor the GCV is used for regularization, and the cut-off computed by the GCV does not always lead to discarding components below the cut-off. Similarly, on some occasions no truncation based on the SVD is actually done.

We note that truncating the SVD of the Jacobian to find a better search direction is not new; see, for example, [10, 2]. However, these methods typically rely on truncating the singular values in order (i.e., traditional truncation) to combat the ill-conditioning of J . In [10], for instance, the authors truncate based on a measure of the so-called grade of J . Further, the methods of [10, 2] rely on information from an estimate of $H - J^T J$, where H denotes the Hessian, to improve the search direction. Our technique is unique in that we analyze individual spectral contributions of the residual to decide which components of the SVD are discarded or kept (but possibly filtered). Additionally, unlike the methods of [10, 2], our method is tailored for use with a trust region approach as opposed to a line search, and does not require estimates of the “missing” part of the Hessian. In [23], a trust region approach is applied to large-scale, discrete ill-posed problems, formulating the trust region subproblem as a parameterized eigenvalue problem.

This paper is organized as follows. In section 2, we briefly state the DOT models of interest. For details and further discussion of regularization by parametric level sets we refer the reader to [18, 1]. In section 3, we analyze the potential pitfalls of DGN, LM, and TSVD for the optimization problem (1.2) arising from the class of problems that is our focus. Our algorithm for determining the optimization step is given in section 4. We give theoretical results for our algorithm in section 5, followed by numerical results in section 6. Conclusions and future work are the subject of section 7.

2. DOT models. In DOT imaging, near infrared light is shined into the body, and three-dimensional (3D) images of the diffusion and/or absorption of light inside the tissue are reconstructed from the measured photon flux. We refer to the image of diffusion in vector form as f_d and the image of absorption as f_a , where the vectors are obtained by ordering the 3D images by column and across all slices. We use the vector f to formally denote the vector of unknowns: if both absorption and diffusion images are desired, then $f = [f_a^T, f_d^T]^T$; otherwise, $f = f_a$ or $f = f_d$, depending on the task.

Both the linear and the nonlinear forward models that describe the mapping of image data to measurements on the surface have been used extensively in the literature (for background, see [3, 4]). We consider both cases in this work. Note that due to the parameterization we employ for the images, even the linear forward model leads to an optimization problem that is *nonlinear in terms of the unknown parameter vector*. First, we describe briefly the parameterization of the image(s) in three dimensions. Then we describe the forward model problems and the corresponding optimization problems.

2.1. Image parameterization. In DOT imaging, the physics dictates that the spatial resolution of the imaging problem is limited [4]. However, one can recover regions of large inhomogeneities in the absorption and/or diffusion coefficient, which may reveal the presence of some physical disorder (e.g., cancer). Therefore, we often use a parametric imaging model to target these regions for recovery [18, 20, 1], reducing the number of unknowns from the (very large) number of pixels/voxels in the

image to the number of parameters. In this paper, we aim to recover images that are almost piecewise constant, looking for anomalous regions of diffusion and absorption on a possibly unknown constant background.

Our specific model is the parametric level set model, introduced in [18] for brain imaging limited to the cortex (an essentially two-dimensional (2D) projected problem), and expanded to the fully 3D case as presented in [1]. We describe the model, with some minor modifications, for the 3D case. This parameterization leads to a regularized problem, and the image reconstruction, with respect to this model, is now a well-posed optimization problem.

Consider the n th degree polynomial, $q(x, y, z) = \sum_{i,j,k} c_{ijk} x^i y^j z^k$, where $i + j + k \leq n$. We can evaluate this polynomial at discrete grid points using the matrix-vector product Pc , where c is the vector of coefficients c_{ijk} under an appropriate ordering and P is a matrix with corresponding entries $x^i y^j z^k$, evaluated at grid points. We define the diffusion image using the zero-level set of this function,

$$(2.1) \quad f_d = \frac{\nu \tanh(\alpha^{(d)})}{2} (e + \tanh(-\beta Pc^{(d)})) + \gamma^{(d)} e,$$

where $c^{(d)}$ is the vector of polynomial coefficients for diffusion, e is a vector of all ones, $\gamma^{(d)}$ is the estimate of the background diffusion value, $\gamma^{(d)} + \nu \tanh(\alpha^{(d)})$ is the estimate of the diffusion coefficient inside the anomaly with ν an appropriately chosen constant, and β is a known (constant) scaling parameter to make the transition of the tanh function “sharp.” Similarly, we define the absorption image as $f_a = \frac{\nu \tanh(\alpha^{(a)})}{2} (e + \tanh(-\beta Pc^{(a)})) + \gamma^{(a)} e$. In this formulation, aiming to recover images that are almost piecewise constant, we do not need a large number of parameters. Hence the number of columns in the Jacobian, $J(p)$, of $r(p)$ will be modest. As the number of sources, detectors, and frequencies tends to be limited, the size of the Jacobian in DOT is not very large.

2.2. Linear forward model for DOT. In frequency domain imaging with a linear model for both absorption and diffusion and using (for example) two frequencies, we have $Af + \eta = y$ (cf. (1.1)), where A has the following block structure:

$$(2.2) \quad A = \begin{bmatrix} A_{R1a} & A_{R1d} \\ A_{I1a} & A_{I1d} \\ A_{R2a} & A_{R2d} \\ A_{I2a} & A_{I2d} \end{bmatrix} = [A_a \quad A_d].$$

The subscripts translate as follows: R stands for real part, I stands for imaginary part, 1 and 2 refer to two different modulation frequencies, a stands for absorption, and d stands for diffusion. Consequently, $y = [y_{R1}^T \ y_{I1}^T \ y_{R2}^T \ y_{I2}^T]^T$. We note that A is dense, and for most problems would have far more columns than rows (underdetermined problem).

In the linear model, we assume the background absorption and/or diffusion are known, and that the images we wish to obtain are in fact images of the perturbation of the absorption and diffusion on this known background, i.e., $\gamma^{(a)}$ and $\gamma^{(d)}$ are both zero. Under the parametric model, the vector of unknowns is the parameter vector

$$p = [(c^{(a)})^T, \alpha^{(a)}, (c^{(d)})^T, \alpha^{(d)}]^T,$$

and the imaging problem becomes of the form (1.2), with

$$(2.3) \quad h(p) = Af(p),$$

where we have used the function notation $f(p)$ to indicate that the entries of the vector of pixel values in the two images depend on the parameter vector p .

It is well known that A_a and A_d are ill-conditioned, and that the ill-conditioning in these matrices results in $J(p)$ being ill-conditioned as well.¹

2.3. Nonlinear forward model for DOT. We assume that the region to be imaged is a rectangular region with a limited number of sources, N_s , on the top and a limited number of detectors, N_d , on the top or the bottom or both. We caution the reader that, in this section only, r , x , y , and z refer to spatial variables. We use the diffusion model for photon flux/fluence $\phi_{s,\omega}(r)$ given input $g_s(r)$ from [3]:

$$\begin{aligned}
 -\nabla \cdot (D(r)\nabla\phi_{s,\omega}(r)) + \mu_a(r)\phi_{s,\omega}(r) + i\frac{\omega}{\nu}\phi_{s,\omega}(r) &= 0 \\
 \text{for } r = (x, y, z) \text{ and } -a < x < a, -b < y < b, 0 < z < c, \\
 \phi_{s,\omega}(r) &= 0 \quad \text{for } 0 \leq z \leq c \\
 &\text{and either } x = -a, x = a, y = -b, \text{ or } y = b, \\
 .25\phi_{s,\omega}(r) + \frac{D(r)}{2} \frac{\partial\phi_{s,\omega}(r)}{\partial\xi} &= g_s(r) \quad \text{for } z = 0 \text{ or } z = c.
 \end{aligned}$$

Here, $D(r)$ and $\mu_a(r)$ denote the diffusion and absorption coefficients, ξ denotes the outward unit normal, $i = \sqrt{-1}$, ω represents the frequency modulation of light, and ν is the speed of light in the medium. The integer subscript s indicates the model with a single source at a known position. Knowing the source and the functions $D(r)$ and $\mu_a(r)$ (given by (2.1) and the analogous expression for the absorption), we can compute the corresponding $\phi_{s,\omega}(r)$ everywhere, in particular at the detectors, i.e., at a subset of grid points with $z = 0$ or $z = c$.

For given f_d, f_a , and ω , the photon flux measured at the detectors due to source s is estimated by

$$(2.4) \quad \psi_{f_d, f_a, \omega, s} = Q_{\text{det}} \phi_{f_d, f_a, \omega, s}, \text{ where } A_{f_d, f_a, \omega} \phi_{f_d, f_a, \omega, s} = g_s,$$

and where Q_{det} contains the rows of the identity matrix that correspond to detector locations, and the linear system on the right represents the discretization of the PDE for a fixed source. Therefore, for given f_d and f_a , the vector of *computed* measurements for frequency ω_j and source s_i is

$$h_j^{(i)} = \begin{bmatrix} \text{Re}(\psi_{f_d, f_a, \omega_j, s_i}) \\ \text{Im}(\psi_{f_d, f_a, \omega_j, s_i}) \end{bmatrix}.$$

The images f_d and f_a are represented in terms of the parameter vector p , so if we stack all data vectors for the n_s sources and n_ω frequencies, we obtain the computed results from (1.1):

$$h(p) = [h_1^{(1)T}, h_1^{(2)T}, \dots, h_1^{(n_s)T}, h_2^{(1)T}, \dots, h_{n_\omega}^{(n_s)T}]^T.$$

Therefore, the computational cost associated with a single function evaluation is the cost of solving the large, sparse system (2.4) for all sources and frequencies.

¹Clearly, if the polynomial degree is large, some of the ill-conditioning in J is attributable to the choice of a monomial basis, and better polynomial bases may be advantageous. However, the predominant difficulty is the presence of the matrices A_a and A_d , and ill-conditioning in J is therefore unavoidable.

Although there are methods for solving multiple such systems efficiently during the course of a nonlinear solve for the images [19], it is clear that any optimization routine for solving (1.2) must minimize the number of function evaluations.

In our application, the cost of a Jacobian evaluation is typically similar to the cost of a function evaluation. Constructing the Jacobian is done using an adjoint-type (or costate) approach that exploits the fact that the number of detectors is not large (roughly equal to the number of sources), as discussed in [14] and [28, p. 88]. In the interest of space, we provide only a brief outline. Using (2.4) and differentiating $\psi_{f_d, f_a, \omega_j, s_i}$ with respect to parameter p_k gives

$$(2.5) \quad \frac{\partial \psi_{f_d, f_a, \omega_j, s_i}}{\partial p_k} = -Q_{\det} A_{f_d, f_a, \omega_j}^{-1} \frac{\partial A_{f_d, f_a, \omega_j}}{\partial p_k} \phi_{f_d, f_a, \omega_j, s_i}.$$

Since A_{f_d, f_a, ω_j} does not depend on the source, s_i , we can compute $Q_{\det} A_{f_d, f_a, \omega_j}^{-1}$ for each frequency, ω_j , by solving

$$(2.6) \quad A_{f_d, f_a, \omega_j}^T G_j^T = Q_{\det}^T.$$

Moreover, the matrices $(\partial/\partial p_k)A_{f_d, f_a, \omega_j}$ have to be computed only once (per frequency). Thus, the computational cost for evaluating $J(p)$ after computing $\phi_{f_d, f_a, \omega_j, s_i}$ for the function evaluation is mainly the cost of (2.6), which amounts to solving a number of linear systems equal to the number of detectors times the number of frequencies. Solving such a sequence of linear systems can be done efficiently using Krylov subspace recycling [19], which substantially reduces the cost even further. Computing the sparse matrix-vector products $[(\partial/\partial p_k)A_{f_d, f_a, \omega_j}]\phi_{f_d, f_a, \omega_j, s_i}$ introduces only a modest additional cost as long as the number of parameters is not large.

Therefore, assuming that the number of sources and the number of detectors are roughly equal and that the number of parameters is relatively small, the cost of a Jacobian evaluation and a function evaluation are about the same.

3. Background and motivation. We now discuss the problems for nonlinear least squares algorithms arising from ill-conditioned Jacobians (of the nonlinear residual) and motivate our new algorithm. In the application considered here, DOT, in spite of the regularization, the ill-posedness of the underlying problem causes the Jacobians to be ill-conditioned, though typically not (nearly) singular.

In general, we use a local quadratic model of the objective function $F(p)$ in (1.2) to compute an update to the current iterate. This is based on the reasonable assumption that there is a region around the current iterate in which such a local model is sufficiently accurate to produce a useful update. We refer to this region as the *trust region*,² and we want our update to lie inside the trust region.

Since the second derivatives of the nonlinear residual $r(p)$ with respect to p are usually hard or very expensive to compute, the full quadratic approximation to $F(p)$ is rarely used. Especially in the *small residual case*, when $\|r(p)\|$ is small compared with $\|\nabla r(p)\|$ near a regularized solution, this is warranted by the fact that near a regularized solution $\nabla^2 F(p) \approx \nabla r(p)^T \nabla r(p)$. Ignoring the term $\nabla^2 r(p)$ in the quadratic model leads to the GN model at the current approximate solution p_c [8], $F(p) \approx m_{\text{GN}}(p)$:

$$(3.1) \quad m_{\text{GN}}(p) = \frac{1}{2} r^T r + r^T J(p - p_c) + \frac{1}{2} (p - p_c)^T J^T J(p - p_c),$$

²In an actual algorithm, the trust region is the current estimate of such a region rather than the region itself.

where $r = r(p_c)$ and $J = \nabla r(p_c)$.

In the remainder of this section, we analyze three well-known approaches to approximately solve the GN model problem. The methods can be analyzed in terms of how they approximate the solution of the model problem, or, alternatively, how their updates are defined exactly by approximations to the model problem. In particular, the update s computed by each of the three methods can be formally described using the reduced SVD of J in (3.1) and a diagonal matrix of (nonnegative) filter factors Ψ . In the following, the dependence of r , J , and $m_{GN}(p)$ on the current iterate p_c is understood. Let the reduced SVD of the $m \times n$ current Jacobian J with rank \hat{n} be

$$(3.2) \quad J = U\Sigma V^T = \sum_{i=1}^{\hat{n}} \sigma_i u_i v_i^T,$$

where $U \in \mathbb{R}^{m \times \hat{n}}$, $\Sigma \in \mathbb{R}^{\hat{n} \times \hat{n}}$ with ordered, positive, diagonal coefficients, $\sigma_1 \geq \sigma_2 \geq \dots \geq \sigma_{\hat{n}} > 0$, and $V \in \mathbb{R}^{\hat{n} \times \hat{n}}$. The search direction or solution update for any of the three methods can be expressed as

$$(3.3) \quad s_\Psi = - \sum_{i=1}^{\hat{n}} v_i \frac{u_i^T r}{\sigma_i} \cdot \psi_i = -V\Psi\Sigma^{-1}U^T r,$$

where $\Psi = \text{diag}(\psi_1, \dots, \psi_{\hat{n}})$. The methods differ only in the definition of these *filter factors* ψ_i , which therefore provide a convenient tool for analysis. Next, we show that the filter factors ψ_i for three standard approaches, DGN, LM, and TSVD,³ can be less effective for the problems we consider in this paper, and we argue that the method we propose in section 4 follows a more effective strategy for filtering SVD components. Observe that in terms of s_Ψ ,

$$(3.4) \quad m_{GN}(p_c + s_\Psi) = \frac{1}{2} \sum_{i=\hat{n}+1}^m (u_i^T r)^2 + \frac{1}{2} \sum_{i=1}^{\hat{n}} (u_i^T r)^2 (1 - \psi_i)^2.$$

So, the update s_Ψ leads to a reduction in the GN model of

$$(3.5) \quad R(s_\Psi) = m_{GN}(p_c) - m_{GN}(p_c + s_\Psi) = \frac{1}{2} \sum_{i=1}^{\hat{n}} (u_i^T r)^2 \psi_i (2 - \psi_i).$$

$R(s_\Psi)$ gives the *estimated reduction of the objective function*, a notion that plays an important role in the trust region method discussed later.

In the following, we discuss each of the methods DGN, LM, and TSVD, noting the difficulties with each approach as it is applied to a problem with an ill-conditioned Jacobian, J .

3.1. (Damped) Gauss Newton. Using the reduced SVD of J from (3.2), minimizing m_{GN} with respect to p leads to

$$(3.6) \quad J^T J s_{GN} = -J^T r \quad \Rightarrow \quad s_{GN} = - \sum_{i=1}^{\hat{n}} v_i \frac{u_i^T r}{\sigma_i} \quad (\psi_i = 1 \text{ for } i = 1, \dots, \hat{n}),$$

³This refers to a truncated SVD approximation to the Gauss-Newton direction combined with a trust region approach.

where s_{GN} is the Gauss–Newton update. The estimated reduction of the objective function from (3.5) is

$$(3.7) \quad R(s_{\text{GN}}) = \frac{1}{2} \sum_{i=1}^{\hat{n}} (u_i^T r)^2.$$

If J is ill-conditioned, with one or more very small singular values, σ_i , and the corresponding components, $u_i^T r$, are not comparably small, then the Gauss–Newton update will be large and, in general, outside the trust region. A classical remedy is to use the damped GN (DGN) method [8], where we replace the update $p = p_c + s_{\text{GN}}$ by $p = p_c + \lambda s_{\text{GN}}$, and λ satisfies some line search criteria. Hence the ψ_i in (3.3) for s_{DGN} are given by $\psi_i = \lambda$, $i = 1, \dots, \hat{n}$. However, in this case, the Gauss–Newton search direction is often almost completely determined by the components corresponding to the smallest singular values, and these lead to the least reliable improvements of the (possibly highly) nonlinear objective function, as they are likely far outside the trust region (i.e., the region where the model is reasonably accurate). Typically, a sufficiently small line search parameter leads to an acceptable step, but it likely leads to disproportionately small components in *useful* directions corresponding to the large(r) singular values. Hence, the method may perform poorly, making many small steps; see also [14]. Moreover, the line search leads to additional function evaluations per step.

To demonstrate this potential difficulty with DGN, consider the hypothetical case of an ill-conditioned Jacobian with one very small singular value ($\sigma_{\hat{n}}$) and components $u_i^T r$ of roughly equal magnitude. The component $v_{\hat{n}}(u_{\hat{n}}^T r)/\sigma_{\hat{n}}$ is very large and (we assume) outside the trust region. For the sake of argument, we assume that the remainder of the update, $\tilde{s}_{\text{GN}} \equiv \sum_{i=1}^{\hat{n}-1} v_i(u_i^T r)/\sigma_i$, lies within the trust region. Replacing s_{GN} by $s_{\text{DGN}} = \lambda s_{\text{GN}}$ gives

$$m_{\text{GN}}(p_c + s_{\text{DGN}}) = \frac{1}{2} \sum_{\hat{n}+1}^m (u_i^T r)^2 + \frac{1}{2} \sum_{i=1}^{\hat{n}} (u_j^T r)^2 (1 - \lambda)^2 \quad \text{and}$$

$$R(s_{\text{DGN}}) = \frac{1}{2} \sum_{i=1}^{\hat{n}} (u_j^T r)^2 \lambda (2 - \lambda).$$

So, for very small λ , the reduction in m_{GN} is very small. This example illustrates why in applications like the one we consider here, where function evaluations may be expensive and the Jacobians are ill-conditioned, DGN is typically not a good approach; see also section 6 for numerical results.

3.2. Levenberg–Marquardt. The Levenberg–Marquardt (LM) approach replaces (3.6) with the following problem:

$$(3.8) \quad (J^T J + \mu I) s_{\text{LM}} = -J^T r,$$

which is clearly well-defined for any $\mu > 0$. LM corresponds to (3.1) with $J^T J$ replaced by $J^T J + \mu I$, which can be thought of as regularization. For papers where LM is applied to the ill-posed problem, we refer the reader to [5, 6, 16, 25]. The choice of μ controls the step size and corresponds to estimating the appropriate size of the trust region. To make LM a globally convergent method, typically an update is analyzed to determine whether the estimated reduction of the objective function is

sufficiently close to the actual reduction, and μ is updated accordingly. Formally, LM is equivalent to the trust region method,

$$(3.9) \quad \min_s m_{GN}(p_c + s) \quad \text{subject to } \|s\|_2 \leq \delta,$$

where δ is the (current) trust region radius.

Substituting (3.2) into (3.8) and solving for s , we see that the LM solution update s_{LM} satisfies

$$(3.10) \quad s_{LM} = - \sum_{i=1}^{\hat{n}} v_i \frac{u_i^T r}{\sigma_i} \psi_i, \quad \psi_i = \frac{\sigma_i^2}{\sigma_i^2 + \mu}.$$

LM works much better for our class of problems than DGN, because it uses a different filter factor for each singular value component, and the filtering strongly favors the larger singular values. Yet it suffers from a potential problem in the context of ill-conditioned Jacobian matrices. Since all damping factors depend on a single parameter, LM cannot balance for individual components the relative size of $|u_i^T r|$ and the reduction of the GN model with the corresponding contribution to the length of s_{LM} . Given the value for μ , all components with σ_i^2 near μ or smaller will be damped significantly irrespective of whether this is required given the value of $|u_i^T r|$, potentially leading to a significantly diminished reduction of the (estimated) objective function compared with a filter factor ψ_i close to 1. The use of a single parameter to determine the filter factors means that singular values that are close have similar filter factors irrespective of the relative lengths of the corresponding residual components. Hence, LM cannot sufficiently tune individual filter factors to account for this case.

In short, in the LM method, the components corresponding to the larger singular values (but typically not the largest few) can be overdamped. Moreover, relatively large components in the residual corresponding to small singular values are almost completely ignored (strongly overdamped).

3.3. Modified truncated SVD. A popular approach to dealing with ill-conditioned matrix equations is to use a truncated SVD (TSVD); see [10, 2] in the context of ill-conditioned Jacobians and [17, Chapter 3 and section 3.5] in the context of regularization (both are relevant here). A straightforward extension to the current setting is to modify the TSVD to comply with the trust region constraint. Using the reduced SVD of J , this amounts to computing the TSVD update as

$$(3.11) \quad s_{TSVD} = - \sum_{i=1}^{k-1} v_i \frac{u_i^T r}{\sigma_i},$$

where k is such that $\|s_{TSVD}\| \leq \delta$ but adding $-v_k(u_k^T r/\sigma_k)$ would make s_{TSVD} too long. Since we want the approximate solution to lie on the trust region boundary when the Gauss-Newton step falls outside the trust region, we include the direction v_k , but scale this direction such that the solution update has length δ :

$$(3.12) \quad s_{MTSVD} = s_{TSVD} - v_k c \frac{u_k^T r}{\sigma_k},$$

where c is determined so that $\|s_{MTSVD}\| = \delta$. We compare this modified approach with DGN, LM, and our new method in the numerical results section. In [13], the GCV method is proposed to compute a cut-off for a TSVD. However, this is in the

context of the (original) ill-posed problem. In our experiments, k is typically quite small, i.e., only a few components are included, and hence quite a few directions are excluded. For the update s_{MTSVD} we have

$$(3.13) \quad m_{\text{GN}}(s_{\text{MTSVD}}) = \frac{1}{2} \sum_{i=\hat{n}+1}^m (u_i^T r)^2 + \frac{1}{2} (u_k^T r)^2 (1-c)^2 + \frac{1}{2} \sum_{i=k+1}^{\hat{n}} (u_i^T r)^2,$$

which corresponds to $\psi_i = 1$ (no damping) for $i = 1, \dots, k-1$, $\psi_k = c$, and $\psi_i = 0$ for $i = k+1, \dots, \hat{n}$.

Clearly, both the TSVD and MTSVD approaches can suffer from a similar drawback as the LM method: components in the residual for which $|u_i^T r|$ is relatively large but for which $i > k$ do not contribute to the sum (3.5), and therefore the reduction at a single step is not nearly as large as it could be. In general, the approach is too greedy.

4. A more general TSVD trust region approach. The discussion in the previous section illustrates that we need a method with more flexibility to vary the filter factors by considering components more or less individually, except for the overall step length constraint. We now discuss how to compute an effective choice of filter factors ψ_i for the problems of interest to us, yielding an s satisfying (3.3) such that $\|s\| \leq \delta$. The key is to balance the relative importance of the components to minimizing $m_{\text{GN}}(p_c + s)$ with the contribution of these components to the (length of the) solution update, which may only be partially related to the presence of noise. Thus, the choice of the filter factors needs to take the following considerations into account:

1. Emphasize the large singular values as they give a large reduction of the objective function for a small change in the length of the update s .
2. Avoid adding terms which are less important in the sense that $|u_i^T r|$ is relatively small, but whose contribution $|u_i^T r|/\sigma_i$ to the length of s would be unduly large and put us outside the trust region. Components with $|u_i^T r|$ larger than some threshold ε_{GCV} (see section 4.2 for how ε_{GCV} is automatically determined) are considered *critical components*; the remaining components are considered *noncritical components*.
3. To ensure a sufficient reduction of the (presumably accurate) model (3.1) and hence of the objective function, give priority to the critical components and make at least a damped update for each critical component with priority decreasing with decreasing singular value.
4. Make some (modest) update for noncritical components when there is some trust left over, especially for those corresponding to large singular values.
5. If the Gauss–Newton step fits inside the current trust region, take the full Gauss–Newton step.

The remainder of this section is organized as follows. First, we give the algorithm that implements this strategy for determining a solution update, s . The specifics of the GCV-like criterion that we use to determine which components are critical is described in detail in subsection 4.2. In subsection 4.3, we give a globally convergent optimization algorithm that employs the proposed solution update within the Basic Trust Region Algorithm (BTR) from [7, p. 116].

4.1. Solution update algorithm. The algorithm first checks if the full Gauss–Newton step would fit inside the trust region. If this is the case, the Gauss–Newton step is chosen. If this is not the case, the algorithm computes the truncated SVD of J (3.2) based on a parameter τ . The parameter $\tau > 0$ serves mainly a theoretical

purpose and can be chosen arbitrarily small. This parameter is discussed in subsection 5.2; see (5.3) and below. Considering the (reduced) SVD in (3.2), let \hat{j} be such that $\sigma_i \geq \tau$ for $i = 1, \dots, \hat{j}$ and $\sigma_i < \tau$ for $i > \hat{j}$. We define J_τ , U_τ , Σ_τ , and V_τ as follows:

$$(4.1) \quad J_\tau = \sum_{i=1}^{\hat{j}} \sigma_i u_i v_i^T = U_\tau \Sigma_\tau V_\tau^T.$$

The special cases— $\hat{j} = 0$, in which all singular values are less than the threshold, and $\hat{j} = \hat{n}$, in which all singular values are larger than or equal to the threshold—are allowed. Components with index $i > \hat{j}$ will be ignored in the update (if the Gauss-Newton step is too large for the trust region); $\psi_i = 0$ for $i > \hat{j}$.

Next, the algorithm partitions the terms in (3.6) for $i = 1, \dots, \hat{j}$ into critical components and noncritical components, according to their relative importance for reducing m_{GN} (3.1), using a GCV-like condition (described in section 4.2). We use \mathcal{I} to denote the set of indices for critical components and \mathcal{I}^c for its complement. To guarantee a sufficient reduction in all critical components, we divide the trust region into two parts, trust region 1 (TR1) with radius $\delta_1 < \delta$, and trust region 2 (TR2) with radius $\delta_2 \equiv \delta$. When adding a critical component would make the partial solution update⁴ longer than δ_1 , we add all remaining critical components in an optimal fashion such as to minimize m_{GN} while remaining inside TR2.

The algorithm considers the update components in order of decreasing singular value. If adding a component would not make the partial update longer than δ_1 , it is added (critical or not). This helps guarantee we use the full Gauss-Newton step when possible. However, if adding a component would put the (partial) update outside TR1, the algorithm proceeds in one of the following two ways. If the component is critical, the algorithm adds to the (partial) update a combined step in all remaining critical components (including the current one) that minimizes m_{GN} while keeping the length of the update less than or equal to δ_2 . If the component is not critical, we skip it, but we keep track of the largest such component for possible (scaled) inclusion in the final update. Once all \hat{j} indices have been visited, we consider the remaining noncritical components. If the (partial) update does not fully exploit TR2 (the full trust region), we first add the largest noncritical component that has not been added yet (scaled if necessary) to the update. Finally, if any trust remains (we still have not reached the border of TR2), we add the remaining noncritical components in order of decreasing singular value. This leads to our solution update algorithm, Algorithm SU.

Algorithm SU (Input: SVD threshold: τ (5.4); reduced SVD of current J : U, Σ, V (3.2); trust region radius: δ ; current residual: r ; Output: trial update s ;)

Compute update coefficients $t_k = u_k^T r / \sigma_k$ for $k = 1, \dots, \hat{n}$;

if $\sum_{k=1}^{\hat{n}} t_k^2 \leq \delta^2$, { Gauss-Newton update fits inside trust region }

do full Gauss-Newton update; $\psi_k = 1$ for $k = 1, \dots, \hat{n}$;

else

discard components u_k, σ_k, v_k where $\sigma_k < \tau$; { work with $J_\tau = U_\tau \Sigma_\tau V_\tau^T$ }

$\delta_1 = \nu_{\text{crit}} \delta$; $\delta_2 = \delta$; { we use $\nu_{\text{crit}} = 0.75$; experimentation may lead to better values }

$[\mathcal{I}, \mathcal{I}^c] = \text{GCV}(U_\tau, \Sigma_\tau, V_\tau, r)$; { see section 4.2 GCV Partitioning }

$s = 0$; $c_{\text{max}} = 0$; $\text{cnt} = 0$; **for** $k = 1, \dots, \hat{n}$, $\psi_k = 0$; **end**

⁴The term *partial solution update* refers to the estimate of s_Ψ in (3.3) before all \hat{j} components have been visited.

```

for  $k = 1, \dots, \widehat{j}$ ,
  if  $k$ th component not already included in partial update,
    if  $\|s - t_k v_k\| \leq \delta_1$  and  $t_k \neq 0$ , { update in TR1, so add component whether
      critical or not }
       $s = s - t_k v_k$ ;  $\psi_k = 1$ ; mark component  $k$  as included (add to index set  $\mathcal{J}_1$ )
    else
      if  $k \in \mathcal{I}$ , { update outside TR1, but component is critical }
         $\gamma = (\delta_2^2 - \|s\|^2)^{1/2}$ ; { distance to boundary TR2 }
         $\mathcal{J}_2 = \{j \geq k, j \in \mathcal{I}\}$ ; { index set of remaining critical components }
        compute optimal  $\psi_j$  for all  $j \in \mathcal{J}_2$  { LM step for remaining
          critical components - see Optimal  $\psi_j$  below }
        for all  $j \in \mathcal{J}_2$ ,
           $s = s - v_j \frac{u_j^T r}{\sigma_j} \psi_j$ ; mark component  $j$  as included in update
          (add to index set  $\mathcal{J}_1$ )
        end { for all }
      else { update outside TR1 and component is not critical }
        cnt=cnt+1; { save info on most important noncritical component }
        if  $|u_k^T r| > c_{max}$ ,  $c_{max} = |u_k^T r|$ ;  $k_{max} = k$ ;  $t_{k_{max}} = t_k$ ; end
        end { if component critical }
      end { if update in TR1 }
    end { if component not already included }
  end { for }
if cnt > 0, { some updates skipped }
   $\gamma = (\delta_2^2 - \|s\|^2)^{1/2}$ ; { distance to boundary TR2 }
  if  $\gamma > 0$ , { consider adding the most important of noncritical components }
     $\psi_{k_{max}} = \min(\gamma/t_{k_{max}}, 1)$ ;  $s = s - \psi_{k_{max}} t_{k_{max}} v_{k_{max}}$ ;
    mark component  $k_{max}$  as included (add to index set  $\mathcal{J}_1$ )
  end
if trust left,
  add skipped, noncritical components in order of decreasing singular value
  taking the minimum of  $t_k$  and distance to boundary of TR2
  end { if noncritical components skipped }
end { if Gauss-Newton update fits }

```

Optimal ψ_j . The optimal filter factors, ψ_j , for all $j \in \mathcal{J}_2$ (see above) are computed such that we minimize the GN model, m_{GN} , over the remaining critical components within the remaining length γ . If the algorithm reaches this step, $s = -\sum_{i \in \mathcal{J}_1} \frac{u_i^T r}{\sigma_i} v_i$ has already been computed for all $i \in \mathcal{J}_1$, the indices of included components.⁵ We wish to update s so that it has the form

$$(4.2) \quad s = -\sum_{i \in \mathcal{J}_1} \frac{u_i^T r}{\sigma_i} v_i - \sum_{i \in \mathcal{J}_2} \psi_i \frac{u_i^T r}{\sigma_i} v_i,$$

where \mathcal{J}_2 , defined above, denotes the set of remaining critical indices. The algorithm first checks whether damping is needed. If s in (4.2) with $\psi_i = 1$ for all $i \in \mathcal{J}_2$ is inside the trust region TR2, no damping is done. If s is outside TR2, $\|s\|_2 > \delta_2$, we compute filter factors corresponding to \mathcal{J}_2 that minimize m_{GN} over the distance to the boundary of TR2. This is essentially an LM step, except that we use the distance

⁵Note that $\psi_i = 1$ for these components.

to the boundary, γ , in place of the trust region radius. Hence

$$(4.3) \quad \psi_i = \frac{\sigma_i^2}{\sigma_i^2 + \mu}, \quad i \in \mathcal{J}_2,$$

where μ satisfies

$$(4.4) \quad \gamma^2 - \sum_{i \in \mathcal{J}_2} \psi_i^2 \frac{(u_i^T r)^2}{\sigma_i^2} = \gamma^2 - \sum_{i \in \mathcal{J}_2} \frac{\sigma_i^2 (u_i^T r)^2}{(\sigma_i^2 + \mu)^2} = 0.$$

We solve the latter equation using MATLAB's `fzero` routine. Note that it is easy to find a bracket for this zero.

4.2. GCV partitioning. Although we aim to emphasize the solution components associated with the largest singular values, we also need to enforce a sufficient reduction of the local model. Hence, we must ensure that large components (called critical here) of the residual along the left singular vectors u_i are taken into account. We determine the critical components with a GCV-like condition [11, 12]. Just as the GCV functional determines the regularization parameter by minimizing the “errors” introduced by leaving out a particular measurement [11], our GCV-like functional determines a discrete parameter ε_{GCV} that minimizes the residual of the Gauss–Newton solution relative to the number of SVD components included in the approximate Jacobian (see below). Intuitively, this provides a modest number of components that captures a large part of the residual component in $\text{Range}(J_\tau)$. As described in the previous section, critical components are guaranteed a partial correction.

Given the least squares problem $J_\tau s \approx -r$ (where $J_\tau = U_\tau \Sigma_\tau V_\tau^T$), the GCV functional [11, 12] is given by

$$G(\varepsilon) = \frac{\|J_\tau s_\varepsilon + r\|_2}{m \text{trace}(I - J_\tau J_\varepsilon^\dagger)},$$

where ε is the cut-off value that defines the matrix J_ε^\dagger ,

$$J_\varepsilon^\dagger \equiv \sum_{i: |u_i^T r| > \varepsilon} \sigma_i^{-1} v_i u_i^T,$$

in terms of the SVD of J_τ introduced in (4.1), so $\sigma_i \geq \tau$, and $s_\varepsilon = J_\varepsilon^\dagger(-r)$.

We use the GCV functional to balance the relative importance of components to minimizing $m_{\text{GN}}(p_c + s)$ with the contribution of these components to the solution update, which may only be partially related to the presence of noise. As proposed in [21], we use the values $|u_i^T r|$ sorted in descending order as the discrete set of parameters ε_i .

The value $\varepsilon_{\text{GCV}} = \arg \min_i G(\varepsilon_i)$ determines our partitioning for the current Jacobian and residual vector: the indices i , for which $|u_i^T r| > \varepsilon_{\text{GCV}}$, belong to the set of critical indices \mathcal{I} . We note that the GCV condition is used *only* as a first step in partitioning the indices. As our optimization problem is regularized by parametric level sets, we are using the GCV as a convenient tool for a purpose other than its usual application. Unlike traditional TSVD regularization for linear, discrete, ill-posed problems, which would preclude terms below the GCV threshold from the solution (see, e.g., [13]), our algorithm may include some terms corresponding to noncritical indices. This allows noncritical components with large singular values (accurate reduction at a small increase in the length of the solution update) to be included and

provides robustness if the GCV curve does not have a well-defined minimum (i.e., the curve is flat).

We provide a brief example of how this approach improves upon (M)TSVD in section 6.1.

4.3. Trust region algorithm. We combine the Basic Trust Region Algorithm from [7] with Algorithm SU (above) to construct a globally convergent optimization algorithm for solving (1.2). We call the resulting algorithm TREGS (pronounced as the dinosaur abbreviation T. Rex, indicating that our algorithm has “teeth”), for Trust region algorithm with REGularized model Solution. The global convergence proof is given in the next section. Our algorithm avoids unnecessary function and Jacobian evaluations. In addition, following [8], we reduce the number of Jacobian evaluations and reduced SVD computations by doubling the trust region and trying a larger step from the current solution iterate after a *very successful* step ($\rho \geq \eta_2$; see below). We provide the pseudocode for our algorithm below. The Boolean variable `newSVD` is set to 1 when the SVD of the Jacobian should be calculated and to 0 otherwise. The Boolean variable `newJAC` is set to 1 when a new Jacobian needs to be calculated and to 0 otherwise. The last Boolean variable `doublestep` is set to 1 when the algorithm doubles the size of the trust region and tries a larger step at the same solution iterate. If `doublestep` = 1 and the new, larger, step fails, we accept the previous, smaller, step from the same current iterate. As noted in [7], the parameters η_1, η_2, γ_1 , and γ_2 must satisfy the following relations:

$$\begin{aligned} 0 < \eta_1 &\leq \eta_2 < 1, \\ 0 < \gamma_1 &\leq \gamma_2 < 1. \end{aligned}$$

We use the default values suggested in [7, p. 117], $\eta_1 = 0.01$, $\eta_2 = 0.9$, and $\gamma_1 = \gamma_2 = 0.5$. Experimenting with these parameters may yield better results, but we have not done so for this paper.

Algorithm: TREGS

```

Choose initial approximate solution  $p_c$ ;  $r_c = r(p_c)$ ;  $F_c = F(p_c)$ ;  $m_c = F_c$ ;  $J_c = J(p_c)$ 
Choose starting  $\delta$ 
newSVD=1; newJAC=0; doublestep=0;
while not converged,
  if newSVD,  $[U, S, V] = \text{red\_svd}(J_c)$ ; newSVD = 0; end
  compute trial solution update  $s$  using Algorithm SU with
    input  $r_c, U, \Sigma, V, \tau$ , and  $\delta$ 
  compute trial solution  $p_t = p_c + s$ ,  $r_t = r(p_t)$ ,  $F_t = F(p_t)$ 
  use SU output to compute (efficiently)  $m_t = m_{GN}(p_t)$  from (3.1)
   $\rho = (F_c - F_t)/(m_c - m_t)$ 
  if  $\rho \geq \eta_2$  and  $s$  is not a Gauss–Newton step, { very successful step }
    double trust region,  $\delta = 2\delta$ , but continue to use  $p_c, F_c$ , SVD of  $J_c, r_c$ 
    save current trial solution values { in case larger step fails }
    doublestep = 1;
  elseif  $\eta_1 \leq \rho < \eta_2$  { successful step }
    accept the trial solution:  $p_c = p_t$ ;  $F_c = F_t$ ,  $r_c = r_t$ 
    newSVD=1; newJAC=1;
  elseif  $\rho < \eta_1$  { unsuccessful step }
    reject the trial solution and reduce trust region:  $\delta = \gamma_1 \delta$ 
  if doublestep,
    set  $p_c, J_c, r_c, F_c$  to saved trial solution values
    newSVD=1; newJAC=1;

```

```

else
  newJAC=0; newSVD=0;
end
elseif  $\rho \geq \eta_2$  and  $s$  is the Gauss-Newton step, { no need to try a larger step }
  accept trial solution:  $p_c = p_t; F_c = F_t; r_c = r_t;$ 
  newJAC=1; newSVD=1;
end
if newJAC,
   $J_c = J(p_c);$ 
  newJAC=0;
end
end { while }

```

5. Theory. The major result in this section is a proof of global convergence to a first-order critical point for our algorithm. The convergence proof is irrespective of the noise that may or may not be present in the data, and therefore is valid for nonlinear problems in general. Therefore, our algorithm is robust and is likely advantageous for other nonlinear problems with ill-conditioned Jacobians, as long as the Jacobian can be computed at reasonable cost and is not too large (as is the case for our problem). Therefore, to make the proof generally applicable, we do not use the discrepancy principle in our proof. Application of the discrepancy principle is useful for the DOT problem to avoid wasting effort resolving the noise, and we will use it in the experiments reported in section 6, but it may not be a useful stopping criterion in a general problem.

The proof involves showing that the proposed algorithm satisfies a set of sufficient conditions for global convergence given in [7]. In addition, this involves proving that the objective function to be minimized satisfies a set of sufficient conditions from [7]. We prove this result for the forward model in section 2.2; for the PDE-based problem described in section 2.3 the latter step is too involved for the present paper. Moreover, the main issue for this paper is that the algorithm itself satisfies the required conditions for convergence, so that it is robust for more general problems.

We start by showing that the objective function, $F(p)$, under the parametric DOT representation and given the forward model in section 2.2 possesses the required properties for global convergence.

5.1. Properties of the parametric imaging model. For simplicity, we consider the case where we optimize only for diffusion.

LEMMA 5.1. *When solving for a diffusion anomaly on a known background using the forward model (2.3) with the assumptions given in section 2.2, $F(p)$ is twice continuously differentiable and $F(p) \geq 0$.*

Proof. The first property follows immediately from standard calculus applied to the objective function, and the second property follows from the fact $F(p) = (1/2) r(p)^T r(p)$. \square

LEMMA 5.2. *The Jacobian matrix corresponding to (2.3) is bounded in norm.*

Proof. Let the parameter vector $p = [c; \alpha^{(d)}]$ have T components with the last component being $\alpha^{(d)}$ and the first $T - 1$ components being the coefficients of the polynomial as described in section 2.1. The i th component of the residual, r_i , satisfies $r_i = \sum_{j=1}^n A_{ij} f_j - y_i$, where $f_j = \frac{\nu \tanh(\alpha^{(d)})}{2} (1 + \tanh(-\beta P(j, :):c))$ (note $\gamma_d = 0$), $P \in \mathbb{R}^{n \times (T-1)}$ contains the values of the monomials over the grid, and c is the vector

of (polynomial) coefficients. Hence,

$$\frac{\partial f_j}{\partial p_k} = \begin{cases} -\frac{\nu \tanh(\alpha^{(d)})}{2} \beta P(j, k) \operatorname{sech}^2(-\beta P(j, :)c) & \text{for } 1 \leq k \leq T-1, \\ \frac{\nu \operatorname{sech}^2(\alpha^{(d)})}{2} (1 + \tanh(-\beta P(j, :)c)) & \text{for } k = T. \end{cases}$$

Because the functions $\operatorname{sech}^2(x)$ and $\tanh(x)$ are bounded, and since $\frac{\partial r_i}{\partial p_k} = \sum_{j=1}^n A_{ij} \frac{\partial f_j}{\partial p_k}$, it follows that $\|J\|_2$ is bounded independent of $\alpha^{(d)}$ and the entries in c . \square

LEMMA 5.3. *The Hessian matrix $H(p)$ for (2.3) is bounded in the 2-norm independent of p .*

Proof. The proof follows from writing $H = J^T J + S$, using the boundedness of J , and showing that the coefficients of S , too, involve only bounded hyperbolic trigonometric functions and values independent of $\alpha^{(d)}$ and the entries in c . The details are straightforward but tedious, and hence they are omitted here. \square

5.2. Global convergence. Next, we show that our algorithm for computing a trial solution update combined with the trust region algorithm produces a nonlinear least squares solver that converges globally to a first order critical point. Our trust region algorithm follows the Basic Trust Region Algorithm (BTR) in [7]. So, if we show that our algorithm satisfies the requirements for convergence of BTR (see below), the convergence proof in [7] applies.

We have three requirements on the objective function $F(p)$ that are sufficient to apply the convergence proof in [7]:

RQ 1: F is twice continuously differentiable.

RQ 2: F is bounded from below.

RQ 3: $\|\nabla_{pp} F\|$ (the norm of the Hessian of F) is bounded from above.

These three requirements, properties of the objective function, have been shown in the previous subsection.

In addition, the convergence theory in [7] makes the following four requirements on the local model $m_{\text{GN}}(p)$ (at iteration k). Note that our model is just the GN model at iteration k ; see (3.1). We give the requirements in terms of (3.1):

RQ 4: $m_{\text{GN}}(p)$ is twice differentiable on the trust region.

RQ 5: $m_{\text{GN}}(p_c) = F(p_c)$.

RQ 6: $\nabla_p m_{\text{GN}}(p_c) = J(p_c)^T r(p_c) = \nabla_p F(p_c)$.

RQ 7: $\|\nabla_{pp} m_{\text{GN}}(p)\| = \|J(p_c)^T J(p_c)\|$ is bounded at every step k by a constant (independent of k).

It follows from the definitions in section 2 that the residual $r(p)$ is always well-defined and bounded. In the previous subsection, we have shown that $J(p)$ is well-defined and bounded (in the 2-norm) for all p . Hence, requirements RQ 4, RQ 5, and RQ 6 follow immediately from the definition of the GN model. Observe that $\nabla_{pp} m_{\text{GN}}(p) = J(p_c)^T J(p_c)$ (independent of p), and this matrix is well-defined at every iteration and bounded based on the properties of F discussed in the previous subsection. Hence, requirement RQ 7 is satisfied.

The final requirement that our algorithm must satisfy in order to directly apply the BTR convergence theory to our algorithm states that, for all iterations k , the trial solution update s must satisfy

$$(5.1) \quad m_{\text{GN}}(p_c) - m_{\text{GN}}(p_c + s) \geq \kappa(m_{\text{GN}}(p_c) - m_{\text{GN}}(p^M)),$$

where $\kappa \in (0, 1]$ is a constant (over all k) and p^M is the minimizer of model m_{GN} over the trust region, the GN model at the k th approximate solution, p_c . In the notation

introduced in (3.1)–(3.5), we must have

$$(5.2) \quad R(s_{\text{TREGS}}) \geq \kappa(m_{\text{GN}}(p_c) - m_{\text{GN}}(p^M)),$$

where s_{TREGS} denotes the step produced by Algorithm SU at iteration k . In other words, the improvement obtained in our proposed solution update must be a fixed fraction (for all steps) of that obtained by the model minimizer (over the trust region). Since we use the GN model, the LM solution is the model minimizer over the trust region (coinciding with the Gauss–Newton solution if it lies inside the trust region). Hence, if the Gauss–Newton update is outside the trust region, we must compare with the improvement obtained by the LM update. If the Gauss–Newton step fits inside the trust region, Algorithm SU takes the full Gauss–Newton step for $J(p_c)$, which is obviously the model minimizer in this case, and so (5.2) will hold. Hence, *in the remainder of this section we tacitly assume that the Gauss–Newton step does not fit inside the trust region*, and we must show the following:

$$(RQ\ 8) \quad R(s_{\text{TREGS}}) \geq \kappa R(s_{\text{LM}})$$

for a fixed κ over all iterations, as long as the optimization has not converged.

The remainder of this section is devoted to proving that (RQ 8) is satisfied for the update s_{TREGS} computed in Algorithm SU, *assuming that the optimization has not converged*, that is, $\|\nabla F(p_c)\|_2 > \epsilon_g$ (gradient tolerance). Note that given the background of our optimization problem, we will use the discrepancy principle as an additional convergence criterion, but it plays no role in the convergence proof.

The proof of (RQ 8) proceeds in two stages. First, we establish the inequality $R(\tilde{s}_{\text{LM}}) \geq \kappa_1 R(s_{\text{LM}})$, where \tilde{s}_{LM} is the LM solution with J_τ replacing J . Second, we show there exists $\tilde{\kappa}$ such that $R(s_{\text{TREGS}}) \geq \tilde{\kappa} R(\tilde{s}_{\text{LM}})$. Together, these two results prove (RQ 8). The second inequality must be proved by considering several special cases regarding the execution of Algorithm SU.

We assume that an appropriate tolerance, ϵ_g , is given such that the optimization can stop if $\|\nabla F(p_c)\|_2 \leq \epsilon_g$. Given an initial solution p_0 , we define $F_0 = F(p_0)$ and $r_0 = r(p_0)$ according to (1.2). Next, we must choose the cut-off parameter τ for the truncated SVD of J (4.1),⁶ such that

$$(5.3) \quad 0 < \tau < \frac{\epsilon_g}{\sqrt{2F_0}}.$$

For convenience, we will use for our proofs

$$(5.4) \quad \tau = \frac{\epsilon_g}{\sqrt{4F_0}};$$

our choice for numerical experiments is given in section 6. Next, we define the τ -truncated SVD of J , J_τ , and \hat{j} as in (4.1). The special cases— $\hat{j} = 0$, in which all singular values are less than the threshold, and $\hat{j} = \hat{n}$, in which all singular values are larger than or equal to the threshold—are allowed. We define $\sigma_+ = \sup_p \|J(p)\|_2$. Lemma 5.2 guarantees that a finite σ_+ always exists.

Clearly, we cannot make an optimization step using J_τ if $\hat{j} = 0$. However, the following lemma shows that in that case we have converged.

LEMMA 5.4. *Let ϵ_g , τ , and J be defined as above, and let $\sigma_1 = \|J\|_2 < \tau$. Then $\|\nabla F\|_2 < \epsilon_g$.*

⁶The parameter $\tau > 0$ serves mainly a theoretical purpose and can be chosen arbitrarily small.

Proof. Since $\nabla F = J^T r = V \Sigma U^T r$ and $\sigma_1 < \tau$, we have

$$\begin{aligned} \|\nabla F\|_2 &= \left(\sum_{i=1}^{\hat{n}} \sigma_i^2 (u_i^T r)^2 \right)^{1/2} \leq \sigma_1 \left(\sum_{i=1}^{\hat{n}} (u_i^T r)^2 \right)^{1/2} \\ &< \tau \left(\sum_{i=1}^{\hat{n}} (u_i^T r)^2 \right)^{1/2} \leq \tau \|r\|_2. \end{aligned}$$

Since the overall algorithm leads to strictly decreasing residuals (the model minimizer must reduce the objective function unless the gradient is zero; see [7]), we also have $\|\nabla F\|_2 < \tau \|r_0\|_2 = \tau \sqrt{2F(p_0)} < \epsilon_g$, where r_0 is the initial (nonlinear) residual. Hence, the optimization has converged. \square

In the following, assume that the optimization algorithm has not yet converged; hence, $\sigma_1 \geq \tau$ and $J_\tau \neq 0$. Let \tilde{s}_{LM} be the LM update for J_τ . The following lemma provides a lower bound on the ratio $R(\tilde{s}_{\text{LM}})/R(s_{\text{LM}})$.

LEMMA 5.5. *Let $\kappa_1 = \frac{\epsilon_g^2}{4F_0\sigma_+^2}$ and $\|\nabla F\|_2 > \epsilon_g$. Then $R(\tilde{s}_{\text{LM}}) \geq \kappa_1 R(s_{\text{LM}})$.*

Proof. We have from (3.5)

$$(5.5) \quad R(s_{\text{LM}}) = \frac{1}{2} \sum_{i=1}^{\hat{n}} (u_i^T r)^2 \psi_i (2 - \psi_i),$$

$$(5.6) \quad R(\tilde{s}_{\text{LM}}) = \frac{1}{2} \sum_{i=1}^{\hat{j}} (u_i^T r)^2 \tilde{\psi}_i (2 - \tilde{\psi}_i),$$

where the $\tilde{\psi}_i = \sigma_i^2 / (\sigma_i^2 + \tilde{\mu})$ denote the filter coefficients corresponding to \tilde{s}_{LM} with $\tilde{\mu}$ defined as follows. The LM parameters μ and $\tilde{\mu}$ are defined by

$$(5.7) \quad \sum_{i=1}^{\hat{n}} \left(\frac{u_i^T r}{\sigma_i} \right)^2 \psi_i^2 = \sum_{i=1}^{\hat{n}} \left(\frac{u_i^T r}{\sigma_i} \right)^2 \left(\frac{\sigma_i^2}{\sigma_i^2 + \mu} \right)^2 = \delta^2.$$

$$(5.8) \quad \sum_{i=1}^{\hat{j}} \left(\frac{u_i^T r}{\sigma_i} \right)^2 \tilde{\psi}_i^2 = \sum_{i=1}^{\hat{j}} \left(\frac{u_i^T r}{\sigma_i} \right)^2 \left(\frac{\sigma_i^2}{\sigma_i^2 + \tilde{\mu}} \right)^2 \leq \delta^2,$$

where inequality holds in the latter equation only if $\tilde{\mu} = 0$, and hence $\tilde{\psi}_i = 1$ for $i = 1, \dots, \hat{j}$. In this case $\tilde{s}_{\text{LM}} = \sum_{i=1}^{\hat{j}} v_i (u_i^T r / \sigma_i)$ is also the Gauss–Newton update for J_τ . Obviously, we always have $\tilde{\mu} \leq \mu$, and hence $\tilde{\psi}_i \geq \psi_i$, which in turn implies that $\tilde{\psi}_i (2 - \tilde{\psi}_i) \geq \psi_i (2 - \psi_i)$ (for $\tilde{\psi}_i, \psi_i \in (0, 1]$ and $i = 1, \dots, \hat{j}$). Define

$$\eta = \sum_{i=1}^{\hat{n}} (u_i^T r)^2 \quad \text{and} \quad \eta_\tau = \sum_{i=1}^{\hat{j}} (u_i^T r)^2.$$

We complete the proof by contradiction. Assume that $R(\tilde{s}_{\text{LM}}) < \kappa_1 R(s_{\text{LM}})$, that is,

$$\sum_{i=1}^{\hat{j}} (u_i^T r)^2 \tilde{\psi}_i (2 - \tilde{\psi}_i) < \kappa_1 \sum_{i=1}^{\hat{j}} (u_i^T r)^2 \psi_i (2 - \psi_i) + \kappa_1 \sum_{i=\hat{j}+1}^{\hat{n}} (u_i^T r)^2 \psi_i (2 - \psi_i).$$

From $\sum_{i=1}^{\hat{j}} (u_i^T r)^2 \psi_i(2 - \psi_i) \leq \sum_{i=1}^{\hat{j}} (u_i^T r)^2 \tilde{\psi}_i(2 - \tilde{\psi}_i)$ and the inequality above, we derive

$$\begin{aligned} \sum_{i=1}^{\hat{j}} (u_i^T r)^2 \psi_i(2 - \psi_i) &< \kappa_1 \sum_{i=1}^{\hat{j}} (u_i^T r)^2 \psi_i(2 - \psi_i) + \kappa_1 \sum_{i=\hat{j}+1}^{\hat{n}} (u_i^T r)^2 \psi_i(2 - \psi_i) \Leftrightarrow \\ (1 - \kappa_1) \sum_{i=1}^{\hat{j}} (u_i^T r)^2 \psi_i(2 - \psi_i) &< \kappa_1 \sum_{i=\hat{j}+1}^{\hat{n}} (u_i^T r)^2 \psi_i(2 - \psi_i) \Rightarrow \\ (1 - \kappa_1) \psi_{\hat{j}+1}(2 - \psi_{\hat{j}+1}) \sum_{i=1}^{\hat{j}} (u_i^T r)^2 &< \kappa_1 \psi_{\hat{j}+1}(2 - \psi_{\hat{j}+1}) \sum_{i=\hat{j}+1}^{\hat{n}} (u_i^T r)^2 \Leftrightarrow \\ (1 - \kappa_1) \eta_\tau^2 &< \kappa_1 (\eta^2 - \eta_\tau^2) \Leftrightarrow \\ \eta_\tau^2 &< \kappa_1 \eta^2. \end{aligned} \tag{5.9}$$

Moreover,

$$\begin{aligned} \|\nabla F\|_2^2 &= \sum_{i=1}^{\hat{n}} (u_i^T r)^2 \sigma_i^2 = \sum_{i=1}^{\hat{j}} (u_i^T r)^2 \sigma_i^2 + \sum_{i=\hat{j}+1}^{\hat{n}} (u_i^T r)^2 \sigma_i^2 \\ &< \sigma_+^2 \sum_{i=1}^{\hat{j}} (u_i^T r)^2 + \tau^2 \sum_{i=\hat{j}+1}^{\hat{n}} (u_i^T r)^2 = \sigma_+^2 \eta_\tau^2 + \tau^2 (\eta^2 - \eta_\tau^2) \quad (\text{and using (5.9)}) \\ &< \sigma_+^2 \kappa_1 \eta^2 + \tau^2 \eta^2 \\ &< (\sigma_+^2 \kappa_1 + \tau^2) \|r\|_2^2 \\ &< (\sigma_+^2 \kappa_1 + \tau^2) 2F_0 = \epsilon_g^2. \end{aligned}$$

Hence, $R(\tilde{s}_{LM})/R(s_{LM}) < \kappa_1 \Rightarrow \|\nabla F\|_2 < \epsilon_g$. Since by assumption $\|\nabla F\|_2 > \epsilon_g$, we must have $R(\tilde{s}_{LM}) \geq \kappa_1 R(s_{LM})$. Finally, note that σ_+ , F_0 , and ϵ_g are constant over all the nonlinear iterations. \square

Next, we show that $R(s_{TREGS}) \geq \tilde{\kappa} R(\tilde{s}_{LM})$ for some $\tilde{\kappa} > 0$ independent of the iteration, assuming the method has not converged.

Recall that Algorithm SU uses two trust region radii, the full trust region (TR2) with radius δ and the smaller trust region (TR1) with radius $\nu_{\text{crit}}\delta$, where $\nu_{\text{crit}} \in (0, 1)$. First, we deal with the case that the Gauss-Newton step for J_τ fits inside TR2, that is, \tilde{s}_{LM} is the Gauss-Newton step for J_τ .

LEMMA 5.6. *Let \tilde{s}_{LM} be the Gauss-Newton step for J_τ . Then $R(s_{TREGS}) = \kappa_2 R(\tilde{s}_{LM})$ with $\kappa_2 = 1$.*

Proof. Algorithm SU produces the Gauss-Newton iterate for J_τ if this update fits inside TR2. Therefore, $s_{TREGS} = \tilde{s}_{LM}$, and $R(s_{TREGS}) = R(\tilde{s}_{LM})$. \square

The next two lemmas deal with the case that the Gauss-Newton update for J_τ does not fit inside the trust region TR2, so $\tilde{\mu} > 0$. Let \mathcal{I} be the set of critical indices; cf. section 4.1. We must consider two cases, $\mathcal{I} \neq \emptyset$ and $\mathcal{I} = \emptyset$. We consider the case $\mathcal{I} \neq \emptyset$ first.

LEMMA 5.7. *Let $\mathcal{I} \neq \emptyset$ and $\tilde{\mu} > 0$. Then there exists κ_3 such that $R(s_{TREGS}) \geq \kappa_3 R(\tilde{s}_{LM})$.*

Proof. The filter factors for *STREGS* will be denoted by $\hat{\psi}_i$ for $i = 1, \dots, \hat{j}$. We can prove the required result in a straightforward fashion by introducing two judiciously chosen LM updates with filter factors φ_i and $\tilde{\varphi}_i$ and LM parameters λ and $\tilde{\lambda}$, respectively, such that $\varphi_i \geq \tilde{\varphi}_i \geq \alpha\tilde{\psi}_i$ for $i = 1, \dots, \hat{j}$, where the $\tilde{\psi}_i$ are the filter factors for the LM step with J_τ and trust region radius δ introduced above, and α is a constant (over all nonlinear iterations) to be determined. Subsequently, we prove that $\hat{\psi}_i \geq \varphi_i \geq \alpha\tilde{\psi}_i$ for all $i \in \mathcal{I}$.

Let the $\varphi_i = \sigma_i^2 / (\sigma_i^2 + \lambda)$ be the filter factors for an LM step for J_τ with trust region radius $(1 - \nu_{\text{crit}}^2)^{1/2}\delta$, the minimum remaining distance for the update when TR1 will be exceeded. So, λ satisfies

$$(5.10) \quad \sum_{i=1}^{\hat{j}} \left(\frac{u_i^T r}{\sigma_i} \right)^2 \left(\frac{\sigma_i^2}{\sigma_i^2 + \lambda} \right)^2 = (1 - \nu_{\text{crit}}^2)\delta^2.$$

Let the $\tilde{\varphi}_i = \sigma_i^2 / (\sigma_i^2 + \tilde{\lambda})$ be the filter factors for an LM step for J_τ with parameter $\tilde{\lambda} = (\gamma - 1)\sigma_+^2 + \gamma\tilde{\mu}$ with $\gamma = (1 - \nu_{\text{crit}}^2)^{-1/2}$. First, we show that $\varphi_i \geq \tilde{\varphi}_i \geq \alpha\tilde{\psi}_i$ for $i = 1, \dots, \hat{j}$. We have

$$\begin{aligned} \frac{\tilde{\psi}_i}{\tilde{\varphi}_i} &= \frac{\sigma_i^2}{\sigma_i^2 + \tilde{\mu}} \cdot \frac{\sigma_i^2 + \gamma\tilde{\mu} + (\gamma - 1)\sigma_+^2}{\sigma_i^2} = \frac{\sigma_i^2 + \gamma\tilde{\mu} + (\gamma - 1)\sigma_+^2}{\sigma_i^2 + \tilde{\mu}} \\ &= \frac{\sigma_i^2 + \tilde{\mu} + (\gamma - 1)(\sigma_+^2 + \tilde{\mu})}{\sigma_i^2 + \tilde{\mu}} \\ &= 1 + (\gamma - 1) \frac{\sigma_+^2 + \tilde{\mu}}{\sigma_i^2 + \tilde{\mu}}. \end{aligned}$$

Since $\tau \leq \sigma_i \leq \sigma_+$, we get

$$(5.11) \quad \gamma \leq \frac{\tilde{\psi}_i}{\tilde{\varphi}_i} \leq 1 + (\gamma - 1) \frac{\sigma_+^2 + \tilde{\mu}}{\tau^2 + \tilde{\mu}} < 1 + (\gamma - 1) \frac{\sigma_+^2}{\tau^2} \quad \text{and hence}$$

The left inequality implies $\tilde{\varphi}_i \leq (1 - \nu_{\text{crit}}^2)^{1/2}\tilde{\psi}_i$, which gives

$$\sum_{i=1}^{\hat{j}} \left(\frac{u_i^T r}{\sigma_i} \right)^2 \tilde{\varphi}_i^2 \leq \sum_{i=1}^{\hat{j}} \left(\frac{u_i^T r}{\sigma_i} \right)^2 (1 - \nu_{\text{crit}}^2)\tilde{\psi}_i^2 = (1 - \nu_{\text{crit}}^2)\delta^2.$$

This, in turn, gives $\lambda \leq \tilde{\lambda}$ and hence $\varphi_i \geq \tilde{\varphi}_i$. The right inequality in (5.11) implies $\tilde{\varphi}_i \geq \alpha\tilde{\psi}_i$ with $\alpha = (1 + (\gamma - 1)\frac{\sigma_+^2}{\tau^2})^{-1}$.

Note that, by Algorithm SU, the remaining distance in the trust region for the update is at least $(1 - \nu_{\text{crit}}^2)^{1/2}\delta$. Moreover, some critical components may have already been added to the update (with $\hat{\psi}_i = 1$), so that the length of $(1 - \nu_{\text{crit}}^2)^{1/2}\delta$, or more, is only for the remaining critical components. Hence, $\hat{\psi}_i \geq \varphi_i$ for $i \in \mathcal{I}$, so

$\hat{\psi}_i \geq \varphi_i \geq \tilde{\varphi}_i \geq \alpha\tilde{\psi}_i$ for $i \in \mathcal{I}$. Therefore,

$$\begin{aligned} \frac{R(s_{\text{TREGS}})}{R(\tilde{s}_{\text{LM}})} &\geq \frac{\sum_{i \in \mathcal{I}} (u_i^T r)^2 \hat{\psi}_i (2 - \hat{\psi}_i)}{\sum_{i=1}^{\hat{j}} (u_i^T r)^2 \tilde{\psi}_i (2 - \tilde{\psi}_i)} \geq \frac{\sum_{i \in \mathcal{I}} (u_i^T r)^2 \alpha \tilde{\psi}_i (2 - \alpha \tilde{\psi}_i)}{\sum_{i=1}^{\hat{j}} (u_i^T r)^2 \tilde{\psi}_i (2 - \tilde{\psi}_i)} \\ &\geq \frac{\alpha \sum_{i \in \mathcal{I}} (u_i^T r)^2 \tilde{\psi}_i (2 - \tilde{\psi}_i)}{\sum_{i=1}^{\hat{j}} (u_i^T r)^2 \tilde{\psi}_i (2 - \tilde{\psi}_i)} = \frac{\alpha \sum_{i \in \mathcal{I}} (u_i^T r)^2 \tilde{\psi}_i (2 - \tilde{\psi}_i)}{\sum_{i \in \mathcal{I}} (u_i^T r)^2 \tilde{\psi}_i (2 - \tilde{\psi}_i) + \sum_{i \in \mathcal{I}_c} (u_i^T r)^2 \tilde{\psi}_i (2 - \tilde{\psi}_i)} \\ &= \alpha \left(1 + \frac{\sum_{i \in \mathcal{I}_c} (u_i^T r)^2 \tilde{\psi}_i (2 - \tilde{\psi}_i)}{\sum_{i \in \mathcal{I}} (u_i^T r)^2 \tilde{\psi}_i (2 - \tilde{\psi}_i)} \right)^{-1} \geq \alpha \left(1 + \frac{\tilde{\psi}_+(2 - \tilde{\psi}_+) \sum_{i \in \mathcal{I}_c} (u_i^T r)^2}{\tilde{\psi}_\tau (2 - \tilde{\psi}_\tau) \sum_{i \in \mathcal{I}} (u_i^T r)^2} \right)^{-1}, \end{aligned}$$

where $\tilde{\psi}_+ = \sigma_+^2 / (\sigma_+^2 + \tilde{\mu})$ and $\tilde{\psi}_\tau = \tau^2 / (\tau^2 + \tilde{\mu})$. In addition, $|u_i^T r| \leq \varepsilon_{\text{GCV}}$ for $i \in \mathcal{I}_c$ and $|u_i^T r| > \varepsilon_{\text{GCV}}$ for $i \in \mathcal{I}$. Let $\ell = |\mathcal{I}|$, the number of elements in \mathcal{I} . Then $|\mathcal{I}_c| = \hat{j} - \ell$. By assumption we have $\mathcal{I} \neq \emptyset$, and hence $\ell \geq 1$. Therefore,

$$\begin{aligned} \frac{R(s_{\text{TREGS}})}{R(\tilde{s}_{\text{LM}})} &\geq \alpha \left(1 + \frac{\tilde{\psi}_+(2 - \tilde{\psi}_+) \sum_{i \in \mathcal{I}_c} (u_i^T r)^2}{\tilde{\psi}_\tau (2 - \tilde{\psi}_\tau) \sum_{i \in \mathcal{I}} (u_i^T r)^2} \right)^{-1} \\ &\geq \alpha \left(1 + \frac{\tilde{\psi}_+(2 - \tilde{\psi}_+) (\hat{j} - \ell) \varepsilon_{\text{GCV}}^2}{\tilde{\psi}_\tau (2 - \tilde{\psi}_\tau) \ell \varepsilon_{\text{GCV}}^2} \right)^{-1} \\ &\geq \alpha \left(1 + \frac{\tilde{\psi}_+(2 - \tilde{\psi}_+) (\hat{j} - 1)}{\tilde{\psi}_\tau (2 - \tilde{\psi}_\tau)} \right)^{-1} \geq \alpha \left(1 + (\hat{j} - 1) \frac{2\sigma_+^2}{\tau^2} \right)^{-1} \\ (5.12) \quad &\geq \alpha \left(1 + (n - 1) \frac{2\sigma_+^2}{\tau^2} \right)^{-1}. \end{aligned}$$

The two last steps follow from

$$\begin{aligned} \frac{\tilde{\psi}_+(2 - \tilde{\psi}_+)}{\tilde{\psi}_\tau (2 - \tilde{\psi}_\tau)} &= \frac{(\tau^2 + \tilde{\mu})^2 \sigma_+^2 (\sigma_+^2 + 2\tilde{\mu})}{(\sigma_+^2 + \tilde{\mu})^2 \tau^2 (\tau^2 + 2\tilde{\mu})} \\ (5.13) \quad &= \frac{\sigma_+^2}{\tau^2} \cdot \frac{\tau^2 + \tilde{\mu}}{\tau^2 + 2\tilde{\mu}} \cdot \frac{\sigma_+^2 + 2\tilde{\mu}}{\sigma_+^2 + \tilde{\mu}} \cdot \frac{\tau^2 + \tilde{\mu}}{\sigma_+^2 + \tilde{\mu}} \leq 2 \frac{\sigma_+^2}{\tau^2}, \end{aligned}$$

and $\hat{j} \leq \hat{n} \leq n$. Note that although ε_{GCV} changes from one optimization step to the next, an appropriate value ε_{GCV} exists at every optimization step, and no assumption is made on the cut-off index \hat{j} . So, the bound $R(s_{\text{TREGS}}) \geq \kappa_3 R(\tilde{s}_{\text{LM}})$ holds independent of iteration with $\kappa_3 = \alpha (1 + 2(n - 1) \sigma_+^2 / \tau^2)^{-1}$ (note that α , n , σ_+ , and τ are all constant over the nonlinear iterations). \square

Next, we consider the second case.

LEMMA 5.8. *Let $\mathcal{I} = \emptyset$. Then there exists a $\kappa_4 > 0$, independent of iteration, such that $R(s_{\text{TREGS}}) \geq \kappa_4 R(\tilde{s}_{\text{LM}})$.*

Proof. In phase I of the algorithm, some components might be added if the updates fit inside TR1. Since we assume the Gauss-Newton update does not fit inside TR2 (the larger trust region), not all components are added. We use the remainder of TR2, which is at least $(1 - \nu_{\text{crit}}^2)^{1/2} \delta$, first for the maximum component, $\max_i |u_i^T r|$, with index m , unless it has been added already, and possibly for other remaining components. If the component with $\max |u_i^T r|$ has been added in phase I or if the update fits within TR2, we have $\hat{\psi}_m = 1$. Otherwise, the component fills the remaining distance to the boundary of TR2, and we have $1 > \hat{\psi}_m \geq \frac{\sigma_m}{|u_m^T r|} (1 - \nu_{\text{crit}}^2)^{1/2} \delta$.

If $\hat{\psi}_m = 1$, then

$$\begin{aligned}
 \frac{R(s_{\text{TREGS}})}{R(\tilde{s}_{\text{LM}})} &\geq \frac{(u_m^T r)^2}{\sum_{i=1}^{\hat{j}} (u_i^T r)^2 \tilde{\psi}_i (2 - \tilde{\psi}_i)} \geq \frac{(u_m^T r)^2}{\sum_{i=1}^{\hat{j}} (u_m^T r)^2 \tilde{\psi}_i (2 - \tilde{\psi}_i)} \\
 (5.14) \qquad &= \frac{1}{\sum_{i=1}^{\hat{j}} \tilde{\psi}_i (2 - \tilde{\psi}_i)} \geq \frac{1}{\hat{j}} \geq \frac{1}{n}.
 \end{aligned}$$

If $1 > \hat{\psi}_m \geq \frac{\sigma_m}{|u_m^T r|} (1 - \nu^2)^{1/2} \delta$, then

$$\begin{aligned}
 \frac{R(s_{\text{TREGS}})}{R(\tilde{s}_{\text{LM}})} &\geq \frac{(u_m^T r)^2 \hat{\psi}_m (2 - \hat{\psi}_m)}{\sum_{i=1}^{\hat{j}} (u_i^T r)^2 \tilde{\psi}_i (2 - \tilde{\psi}_i)} \geq \frac{(u_m^T r)^2 \hat{\psi}_m (2 - \hat{\psi}_m)}{\sum_{i=1}^{\hat{j}} (u_m^T r)^2 \tilde{\psi}_i (2 - \tilde{\psi}_i)} \\
 (5.15) \qquad &= \frac{\hat{\psi}_m (2 - \hat{\psi}_m)}{\sum_{i=1}^{\hat{j}} \tilde{\psi}_i (2 - \tilde{\psi}_i)} \geq \frac{\hat{\psi}_m (2 - \hat{\psi}_m)}{\hat{j} \tilde{\psi}_+ (2 - \tilde{\psi}_+)}.
 \end{aligned}$$

Next we show that $\hat{\psi}_m \geq (1 - \nu_{\text{crit}}^2)^{1/2} \tilde{\psi}_\tau$, and therefore $\hat{\psi}_m (2 - \hat{\psi}_m) \geq (1 - \nu_{\text{crit}}^2)^{1/2} \tilde{\psi}_\tau (2 - (1 - \nu_{\text{crit}}^2)^{1/2} \tilde{\psi}_\tau)$. Using (5.8) with equality (and $\tilde{\mu} > 0$), since we assume the Gauss–Newton update does not fit inside TR2, and $\tilde{\psi}_\tau \leq \tilde{\psi}_i$ for $i = 1, \dots, \hat{j}$, we have

$$\sum_{i=1}^{\hat{j}} \left(\frac{u_i^T r}{\sigma_i} \right)^2 \tilde{\psi}_\tau^2 \leq \delta^2,$$

whereas

$$\begin{aligned}
 \sum_{i=1}^{\hat{j}} \left(\frac{u_i^T r}{\sigma_i} \right)^2 \hat{\psi}_m^2 &\geq \sum_{i=1}^{\hat{j}} \left(\frac{u_i^T r}{\sigma_i} \right)^2 \left(\frac{\sigma_m}{|u_m^T r|} \right)^2 (1 - \nu_{\text{crit}}^2) \delta^2 = \sum_{i=1}^{\hat{j}} \frac{(u_i^T r)^2}{(u_m^T r)^2} \frac{\sigma_m^2}{\sigma_i^2} (1 - \nu_{\text{crit}}^2) \delta^2 \\
 &= (1 - \nu_{\text{crit}}^2) \delta^2 \left(1 + \sum_{i \neq m} \frac{(u_i^T r)^2}{(u_m^T r)^2} \frac{\sigma_m^2}{\sigma_i^2} \right) \geq (1 - \nu_{\text{crit}}^2) \delta^2.
 \end{aligned}$$

Hence,

$$\sum_{i=1}^{\hat{j}} \left(\frac{u_i^T r}{\sigma_i} \right)^2 \frac{\hat{\psi}_m^2}{1 - \nu_{\text{crit}}^2} \geq \delta^2,$$

and therefore

$$\begin{aligned}
 \frac{\hat{\psi}_m}{(1 - \nu_{\text{crit}}^2)^{1/2}} &\geq \tilde{\psi}_\tau \quad \Leftrightarrow \\
 (5.16) \qquad \hat{\psi}_m &\geq (1 - \nu_{\text{crit}}^2)^{1/2} \tilde{\psi}_\tau.
 \end{aligned}$$

From (5.15) and (5.16) we derive

$$\begin{aligned}
 \frac{R(s_{\text{TREGS}})}{R(\tilde{s}_{\text{LM}})} &\geq \frac{\hat{\psi}_m (2 - \hat{\psi}_m)}{\hat{j} \tilde{\psi}_+ (2 - \tilde{\psi}_+)} \\
 &\geq \frac{(1 - \nu_{\text{crit}}^2)^{1/2} \tilde{\psi}_\tau (2 - (1 - \nu_{\text{crit}}^2)^{1/2} \tilde{\psi}_\tau)}{\hat{j} \tilde{\psi}_+ (2 - \tilde{\psi}_+)} \geq \frac{(1 - \nu_{\text{crit}}^2)^{1/2}}{\hat{j}} \frac{\tilde{\psi}_\tau (2 - \tilde{\psi}_\tau)}{\tilde{\psi}_+ (2 - \tilde{\psi}_+)} \\
 &\geq \frac{(1 - \nu_{\text{crit}}^2)^{1/2}}{n} \frac{\tau^2}{2\sigma_+^2}.
 \end{aligned}$$

The last step follows from (5.13). Finally, we have $R(\text{stREGS}) \geq \kappa_4 R(\tilde{s}_{\text{LM}})$ with $\kappa_4 = \frac{(1-\nu_{\text{crit}}^2)^{1/2}}{n} \frac{\tau^2}{2\sigma_+^2}$ (again, note that all parameters are constant over the nonlinear iteration). \square

This brings us to the main result of this subsection.

THEOREM 5.9. *If the requirements RQ 1–RQ 7 on the objective function and the local model are satisfied, then there exists a $\kappa > 0$, independent of the iteration k , such that $m_k(p_k) - m_k(p_k + s_k) \geq \kappa(m_k(p_k) - m_k(p_k^M))$ is satisfied.*

Proof. The proof follows from the previous lemmas and taking $\kappa = \kappa_1 \min(\kappa_2, \kappa_3, \kappa_4)$. \square

COROLLARY 5.10. *Under the assumptions in the previous theorem, the algorithm TREGS is guaranteed to converge to a first-order critical point.*

Proof. Our algorithm satisfies all the requirements for the BTR algorithm from [7] to converge to a first-order critical point. For the remainder of the proof we refer the reader to [7, section 6.4]. \square

6. Numerical results. All numerical results were computed using MATLAB in double precision arithmetic. We split the numerical results into two subsections: those dealing with the linear DOT model and those dealing with the nonlinear DOT model. We present comparisons of our method with the LM and DGN implementations in MATLAB and with the MTSVD algorithm described in section 3.3.

Our comparisons are in terms of the total number of function evaluations and Jacobian evaluations until the discrepancy principle, our effective stopping criterion, is reached or a gradient tolerance is satisfied. A stopping criterion based on the discrepancy principle is relevant for our DOT problems, because of the presence of measurement noise. This avoids wasting computational effort trying to resolve the noise. The gradient tolerance and the related SVD cut-off τ were chosen as follows. Let ε_F be the tolerance from the discrepancy principle, given below for each test problem. Then $\varepsilon_g = 10^{-7}\varepsilon_F$ and $\tau = 0.1 \frac{\varepsilon_g}{\|r_0\|}$. Numbers are reported for various noise realizations, starting guesses, and configurations of the absorption and diffusion anomalies. In addition, for one experiment for the nonlinear forward problem, we give a qualitative assessment of the final image reconstruction of the diffusion image for both TREGS and LM.

6.1. Linear forward model. We tested our algorithm extensively on four different test problems, varying the starting guesses and the noise realizations, to get a complete picture of the behavior of our algorithm, TREGS, relative to LM, DGN, and MTSVD. This extensive testing strongly suggests that the demonstrated improvements by our algorithm are systematic and not the result of a lucky starting guess. In each of the four test problems, we are inverting jointly for absorption and diffusion perturbation images. The four test problems differ in the locations and sizes of the anomalies, as well as the values inside those anomalies. We mimic the setup in [18], where the goal is to locate anomalies on the cortex of the brain through the use of a one-to-one mapping from a region in \mathbb{R}^2 to the cortical surface. Thus, the 3D imaging problem becomes inherently 2D. We use polynomials of degree two or less. Therefore, the total number of unknown parameters is 14: 7 to describe the absorption image (6 polynomial parameters plus the value inside the anomaly) and 7 to describe the diffusion image.

In each of the four test problems, the “true” data are created as $y_{\text{true}} = Af$ (see section 2.2) using two modulation frequencies and splitting the real and imaginary components as described. To compute a single noise realization, we first generate

TABLE 6.1

Comparison of our proposed method (TREGS) with LM, DGN, and MTSVD on the first test problem, for two different starting guesses for each of three noise realizations. The * in the DGN column means that the method switched over to LM after a few iterations due to poor conditioning of the gradient. Err1 and Err2 give the relative error in the diffusion and absorption images in the 2-norm (for the image as a vector).

Config	TREGS				LM			
	FEV	JEV	Err1	Err2	FEV	JEV	Err1	Err2
1,1,NR1	82	45	0.126	0.098	247	56	0.195	0.166
1,1,NR2	25	13	0.071	0.090	224	52	0.125	0.223
1,1,NR3	48	25	0.093	0.067	175	40	0.125	0.165
1,2,NR1	18	11	0.172	0.075	91	21	0.075	0.044
1,2,NR2	23	11	0.223	0.055	74	17	0.075	0.125
1,2,NR3	19	10	0.159	0.119	79	18	0.073	0.035

Config	DGN				MTSVD			
	FEV	JEV	Err1	Err2	FEV	JEV	Err1	Err2
1,1,NR1	202*	46	0.098	0.090	91	54	0.148	0.064
1,1,NR2	241*	54	0.098	0.218	103	60	0.172	0.083
1,1,NR3	233*	53	0.162	0.131	102	51	0.191	0.100
1,2,NR1	136*	32	0.221	0.116	52	32	0.080	0.036
1,2,NR2	324*	75	0.295	0.146	53	28	0.077	0.039
1,2,NR3	154*	37	0.120	0.116	94	47	0.100	0.039

four noise subvectors using the `randn` function—each subvector corresponding to the conformal partitioning in (2.2) for the matrix A in (2.3). Each noise subvector was then scaled so that the relative noise level (for that subvector) was 1 percent, and then that noise subvector was added to the corresponding true data subvector. We generated three different noise realizations according to this method. The scaling values were saved and used to define the matrix W in order to whiten the data. That is, we use $A \leftarrow WA, y \leftarrow Wy = W(y_{true} + \eta)$. Hence, our stopping criterion, based on the discrepancy principle, was

$$\|Af - y\| < tol \approx \|W\eta\|,$$

where we used the same value for tol , $tol = 10.5$, for each noise realization. Of course, the choice of the tolerance in the stopping criterion can have a slight effect on the quality of the solution, since a value too small could mean unnecessarily trying to overfit for noise. However, in our experiments, underestimating a stopping tolerance does not dramatically affect the quality of the solution, primarily because the regularization is accomplished through the parameterization itself.

First, in Table 6.1, we compare the performance of LM, DGN, and MTSVD with our algorithm (TREGS) for the first test problem, two starting guesses, and three noise realizations. For the first starting guess, the shape was a circle centered at (0,0) with radius 1 and in the second case it was a circle centered at (0,0) with radius 0.5. Then we ran 3 optimization steps with the shape parameters fixed and inverted only for the 2 values inside the anomalies. The first observation from Table 6.1 is that DGN is not effective in choosing an appropriate search direction. In fact, MATLAB's DGN, implemented by the `lsqnonlin` routine, always switches to LM once it detects an ill-conditioned gradient and slow progress (indicated by * in Table 6.1).

In our implementation of the MTSVD approach, we used an algorithm from Dennis and Schnabel [8, Alg A.6.4.5] for updating the model trust region. Jacobian updates were done only once a step was accepted. As discussed in section 3.3, this

TABLE 6.2

Comparison of our proposed method, TREGS, to LM for test problems 2–4, for two different starting guesses for each of three noise realizations. *Err1* and *Err2* give the relative error in the diffusion and absorption images in the 2-norm (for the image as a vector).

Config	TREGS				LM			
	FEV	JEV	Err1	Err2	FEV	JEV	Err1	Err2
2,1,NR1	71	29	0.141	0.024	127	30	0.072	0.027
2,1,NR2	78	35	0.110	0.120	131	31	0.128	0.108
2,1,NR3	132	61	0.187	0.134	121	29	0.123	0.051
2,2,NR1	144	74	0.172	0.058	137	33	0.107	0.050
2,2,NR2	50	25	0.116	0.138	137	33	0.185	0.146
2,2,NR3	46	24	0.196	0.092	125	30	0.123	0.096
3,1,NR1	31	20	0.062	0.040	143	33	0.056	0.015
3,1,NR2	37	23	0.041	0.036	140	32	0.046	0.014
3,1,NR3	29	16	0.076	0.012	138	32	0.051	0.012
3,2,NR1	34	17	0.057	0.034	130	30	0.046	0.023
3,2,NR2	40	20	0.040	0.020	120	28	0.053	0.048
3,2,NR3	43	22	0.051	0.013	112	26	0.042	0.049
4,1,NR1	16	9	0.426	0.268	84	20	0.284	0.219
4,1,NR2	19	10	0.271	0.399	89	21	0.460	0.395
4,1,NR3	29	12	0.392	0.224	41	10	0.253	0.177
4,2,NR1	40	9	0.413	0.250	32	8	0.486	0.329
4,2,NR2	48	8	0.366	0.561	41	10	0.317	0.205
4,2,NR3	43	9	0.365	0.245	41	10	0.240	0.182

approach is still too greedy to perform well on these test problems compared with TREGS. However, MTSVD is a serious competitor for LM.

Table 6.1 also clearly shows that LM needs significantly more function evaluations and generally more Jacobian evaluations (though not by a similar factor) than our proposed method. The behavior of the four methods was, in this respect, consistent across all experiments. In the interest of space, and as LM is probably the most common method for these optimization problems, we report only the comparison of TREGS with LM for the remaining experiments. These results are presented in Table 6.2. We do report on MTSVD for the nonlinear forward problem in the next section.

In Figure 6.1, we demonstrate the effectiveness of the GCV-like condition in TREGS. On the top, for a representative step of TREGS (the second iteration of “1,1 NR3” in Table 6.1), we plot the magnitude of the residual components along the left singular vectors versus the rank of the singular values (from large to small). For the first four components, indicated by the disks, the full length of the correction (the Gauss–Newton update in the right singular value direction) is included in the solution update. The full correction in the direction of the fifth component would put the solution update outside the first trust region, triggering the update of all remaining critical components, indicated by the triangles. The GCV cut-off is indicated by the dotted line. The components corresponding to the pluses are ignored. Note that these are not the components corresponding to the smallest singular values. On the bottom, we plot the magnitude of the components of the solution update corresponding to the residual components along the left singular vectors. Note that components 3 to 12 contribute roughly equally to the length of the Gauss–Newton solution update, but TREGS selects from these the ones that contribute most to the residual reduction. MTSVD would include the full correction along the first four components and a damped correction along the fifth component. All other directions

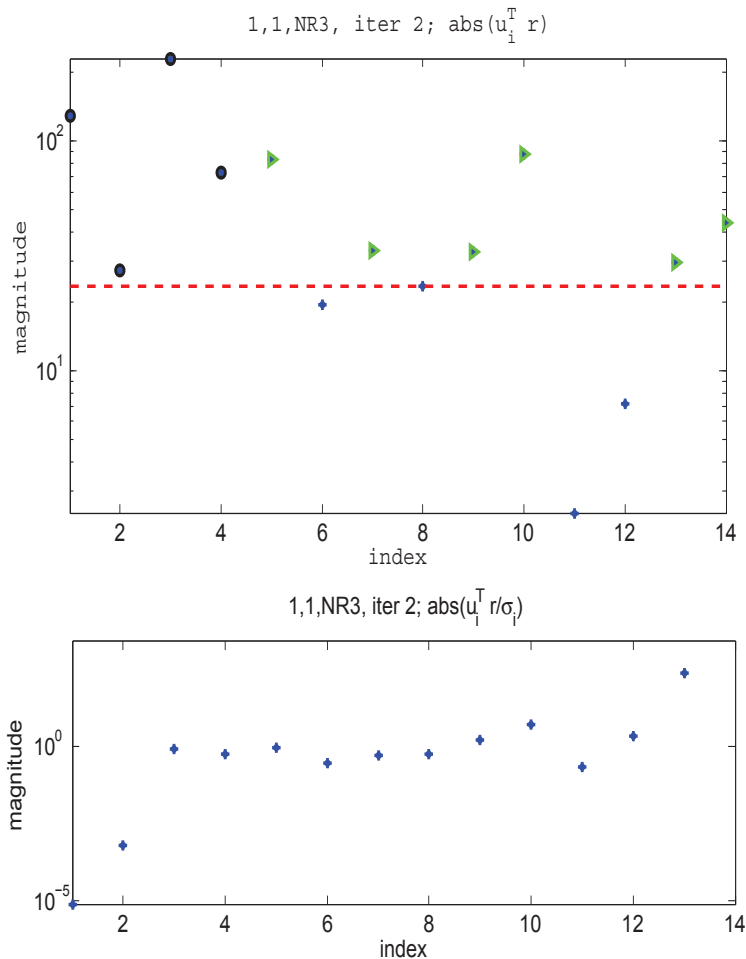


FIG. 6.1. Comparison of TREGS and MTSVD demonstrating the importance of the GCV-like condition for a representative step (in “1,1 NR3”). Top: the magnitude of the residual components along the left singular vectors versus the rank of the singular values (from large to small) for a representative step of TREGS. For the disks the full Gauss–Newton correction along the left singular vectors is included in the solution update. For the triangles a damped Gauss–Newton correction was made. The components corresponding to the pluses are ignored. Bottom: we plot the magnitude of the solution update components versus the rank of the singular values. Components 3 to 12 would contribute roughly equally to the length of the Gauss–Newton update, but TREGS selects from these the ones that contribute most to the residual reduction.

would be ignored, which includes multiple directions with large components in the residual. In the first two optimization steps, the objective function for TREGS goes from $1.39e5$ (initial value of the objective function) to $5.75e4$ to $2.69e4$, whereas for MTSVD, the objective function goes from $1.39e5$ to $5.92e4$ to $3.27e4$, reflecting that MTSVD is less effective in reducing the objective function. If we would apply the MTSVD to the TREGS solution after one iteration, we would get a model reduction (3.5) of $4.12e4$ versus $4.22e4$ for the model reduction that TREGS yields. The cumulative effect of these differences is that TREGS converges in 48 function evaluations (25 Jacobian evaluations), while MTSVD converges in 102 function evaluations (51

Jacobian evaluations); see Table 6.1.

6.2. Nonlinear forward model. In this section, we give comparisons for a few different configurations and noise realizations. Again, we provide extensive test results to demonstrate that the improvements by our algorithm are systematic and not the result of a lucky starting guess. In the first three experiments, the region of interest was $8\text{cm} \times 8\text{cm} \times 4\text{cm}$, discretized with $N \times N \times N_z$ grid points (the actual values are given below). In the next two experiments, the region of interest was $6\text{cm} \times 6\text{cm} \times 4\text{cm}$, discretized with $N \times N \times N_z$ grid points. In all experiments, data were taken at two frequencies, 0 and 50 MHz (recall that we obtain twice the amount of data at 50 MHz in the sense that we separate the real and imaginary contributions). Sources and detectors were located on both the top and the bottom of the box and were assumed to coincide with grid points for simplicity. In practice, the numbers of sources and detectors for a fixed region of interest are limited by the cost of the apparatus and the physics of the problem—that is, the spacing of sources and detectors for a fixed region of interest is bounded from below independent of the mesh spacing used in the forward solve.

In all examples, we invert for the parameters of the polynomials describing both the absorption and the diffusion anomalies, and we invert for the interior parameters. Thus, the length of the parameter vector was 22, giving a Jacobian that is still quite tall and skinny. In all experiments, our critical trust region in TREGS is 80 percent of the actual trust region ($\delta_1 = 0.8\delta_2$ in Algorithm SU). The stopping criterion in each example was $1.1\|W\eta\|$, where η is the noise vector. This choice was used to simulate a 10 percent error in the estimation of the noise level as well as to prevent wasted effort oversolving for the solution.

The experiments are described in detail below. Note that the experiments include examples on different grid spacings, with different noise levels, different noise realizations, and different starting guesses, with the intent of showing the robustness of our approach. The true absorption and diffusion anomalies were unchanged in experiments 1–3; the anomalies in 4 and 5 were smaller, but the same for each of those two experiments.

1. Here, $N = 21, N_z = 17$ on an $8\text{cm} \times 8\text{cm} \times 4\text{cm}$ region of interest. There was a 10×10 equispaced grid of sources on the top and bottom, and a 9×9 equispaced grid of detectors on the top and on the bottom. We used ellipses for the starting guesses for the absorption and diffusion anomalies, and then 4 optimization steps to determine loose estimates of the anomaly values inside these shapes. The starting shapes were different from each other, and in the same off-center directions as the actual centers, but larger than the actual objects. We ran this experiment for 1 percent and 5 percent noise.
2. In this comparison, $N = 26, N_z = 17$ on the $8\text{cm} \times 8\text{cm} \times 4\text{cm}$ region of interest. We simulated a 12×12 equispaced grid of sources on the top and bottom, and a 12×12 equispaced grid of detectors on the top and bottom (sources and detectors were not co-located). The starting guess was obtained in the same way as described in experiment 1. We report results for two different noise realizations, both at 5 percent noise.
3. For this experiment, $N = 33, N_z = 17$ on the $8\text{cm} \times 8\text{cm} \times 4\text{cm}$ region of interest. We simulated a 16×16 equispaced grid of sources on the top and bottom, and a 15×15 equispaced grid of detectors on the top and bottom. We used the same starting shapes as in the previous two experiments, but fixed the starting values for the interior of the diffusion and absorption anomalies

TABLE 6.3

Results for the first three configurations. The NL column indicates the noise level for that particular experiment. The * indicates the maximum number of function evaluations (75) was exceeded prior to convergence. Err1 and Err2 give the relative error in the diffusion and absorption images in the two-norm (for the image as a vector).

Ex	NL	TREGS				LM				MTSVD			
		FEV	JEV	Err1	Err2	FEV	JEV	Err1	Err2	FEV	JEV	Err1	Err2
1	1%	35	11	0.066	0.396	71	17	0.086	0.337	27	12	0.083	0.439
1	5%	29	10	0.280	0.681	26	7	0.345	0.743	*	*	*	*
2	5%	29	14	0.378	0.828	39	10	0.397	0.731	*	*	*	*
2	5%	21	12	0.111	0.411	26	7	0.334	0.701	*	*	*	*
3	10%	24	10	0.315	0.755	28	7	0.463	0.732	12	5	0.344	0.754

at 0.11cm^{-1} and 0.04cm^{-1} , respectively.

4. Here, the region of interest was $6\text{cm} \times 6\text{cm} \times 4\text{cm}$, with $N = 21, N_z = 17$. We use the same source/detector arrangement as in experiment 1, though the spacing between adjacent sources (detectors) was smaller (6mm). The starting shapes for both absorption and diffusion anomalies were the same, but chosen as the largest ellipse to fit in the box with axes oriented parallel with the coordinate axes. The starting internal anomaly values were determined by 4 optimization steps used to determine loose estimates of the anomaly values given those large starting shapes. We used 1 percent noise, and ran this for two different noise realizations.
5. We used the same region of interest as in the previous example, but $N = 30, N_z = 17$. We also used the same configuration of sources and detectors as in the previous example, even though the grid resolution is twice as fine in the x and y directions. Finally, we used the same starting ellipse shapes, but the starting value for the diffusion coefficient inside the diffusion anomaly was taken to be 0.12cm^{-1} , and the starting value for the absorption coefficient inside the absorption anomaly was taken to be 0.04cm^{-1} . We used 5 percent noise, and ran this for two different noise realizations.

Table 6.3 gives the numerical comparisons for the first three experiments. We tested our new method (TREGS), LM, and MTSVD. DGN consistently gave poor results, and the MATLAB implementation, via `lsqnonlin`, always warned that it would switch to LM partway through, so we do not report those numbers here. We included results for MTSVD for the first three experiments, because, for the linear problem, we observed that MTSVD often was more competitive with TREGS than LM. Here, however, that only seems to be the case for low noise levels and a reasonable starting guess. For the first three experiments, for runs at the higher noise levels, LM is competitive with TREGS in the number of function and Jacobian evaluations. However, the quality of the solutions is typically worse, indicating that perhaps too much damping of important directions lead LM away from the desired minimum.

We present comparisons between TREGS and LM for experiments 4 and 5 in Table 6.4. Recall that the starting guess is further from the solution in these experiments. For these cases, LM had a great deal of trouble—after a few productive iterations at the beginning, the reduction rate became almost stagnant and resulted in the triggering the failsafe stopping criterion of 100 function evaluations. This behavior was found both at the 1 percent (experiment 4) and 5 percent (experiment 5) noise levels, for all four noise realizations. In Figure 6.2, for experiment 5, we give the computed reconstructions of the diffusion image (in translucent blue) for TREGS and LM compared with the true diffusion image (in orange) as well as the starting

TABLE 6.4

Results for configurations 4 and 5. For configuration 4, both runs were for 1 percent noise at different noise realizations; for configuration 5, both runs were for 5 percent noise at different noise realizations. The * means that LM terminated at the first step after the maximum of 100 function evaluations was reached (so the results are for a nonconverged approximate solution). Err1 and Err2 give the relative error in the diffusion and absorption images in the two-norm (for the image as a vector).

Example	TREGS				LM			
	FEV	JEV	Err1	Err2	FEV	JEV	Err1	Err2
4, NR1	22	12	0.076	0.245	102*	24	0.212	0.532
4, NR2	31	14	0.132	0.539	102*	24	0.272	0.574
5, NR1	20	9	0.276	0.639	102*	24	0.669	0.826
5, NR2	21	9	0.294	0.656	102*	23	0.642	0.828

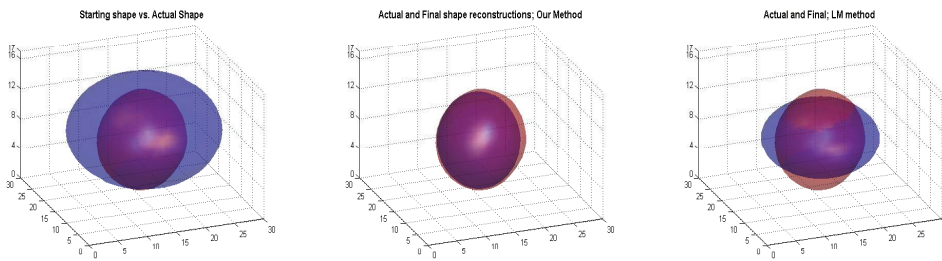


FIG. 6.2. Illustration corresponding to the last row of Table 6.4. Left: starting shape estimate for the diffusion anomaly. Middle: final shape estimate for the diffusion anomaly using TREGS. Right: final shape estimate for the diffusion anomaly using LM.

shape of the diffusion image in the optimization.

7. Conclusions and future work. We have analyzed why several popular non-linear least squares solvers perform poorly for problems with ill-conditioned Jacobians, in particular for problems arising in DOT, our problem of interest. Based on this analysis we propose a new method, TREGS, that combines a trust region approach with regularization for the local model (trust region) problem. In general, we argue, this leads to better optimization steps. Although this is hard to prove or analyze analytically, our extensive numerical experiments show that significant performance improvements are achieved. Compared with LM, the closest competitor, TREGS significantly reduces the number of function evaluations and generally also reduces the number of Jacobian evaluations (though not by a similar factor). A modified TSVD approach often beats LM for the linear forward problem, but it is not competitive with TREGS. For problems like the one we are interested in, a function evaluation corresponds to an expensive (large) dense matrix-vector product, an integral transform, or many solutions of a discretized, 3D PDE. Moreover, the cost of a Jacobian evaluation is about the same as the cost of a function evaluation.

Although further analysis of our proposed algorithm is needed, we show for general problems that the algorithm is guaranteed to converge to a first-order critical point if standard assumptions on the objective function hold.

Important future work remains. We need to do further theoretical analysis of our algorithm, and we would like to test and analyze the algorithm for other problems that suffer from ill-conditioned Jacobians. We expect the algorithm to be competitive for many other such problems, but at this point that is only conjecture.

REFERENCES

- [1] A. AGHASI, M. KILMER, AND E. L. MILLER, *Parametric level set methods for inverse problems*, SIAM J. Imaging Sci., 4 (2011), pp. 618–650.
- [2] M. AL-BAALI AND R. FLETCHER, *Variational methods for non-linear least squares*, J. Oper. Res. Soc., 36 (1985), pp. 405–421.
- [3] S. R. ARRIDGE, *Optical tomography in medical imaging*, Inverse Problems, 16 (1999), pp. R41–R93.
- [4] D. BOAS, D. BROOKS, E. MILLER, C. DIMARZIO, M. KILMER, R. GAUDETTE, AND Q. ZHANG, *Imaging the body with diffuse optical tomography*, IEEE Signal Process. Mag., 18 (2001), pp. 57–75.
- [5] M. BURGER, *Levenberg-Marquardt level set methods for inverse obstacle problems*, Inverse Problems, 20 (2004), pp. 259–282.
- [6] M. BURGER AND W. MÜHLHUBER, *Iterative regularization of parameter identification problems by sequential quadratic programming methods*, Inverse Problems, 18 (2002), pp. 943–969.
- [7] A. R. CONN, N. I. M. GOULD, AND PH. L. TOINT, *Trust-Region Methods*, SIAM, Philadelphia, 2000.
- [8] J. E. DENNIS, JR., AND R. B. SCHNABEL, *Numerical Methods for Unconstrained Optimization and Nonlinear Equations*, Classics Appl. Math. 16, SIAM, Philadelphia, 1996.
- [9] O. DORN AND D. LESSELIER, *Level set methods for inverse scattering*, Inverse Problems, 22 (2006), pp. R67–R131.
- [10] P. E. GILL AND W. MURRAY, *Algorithms for the solution of the nonlinear least-squares problem*, SIAM J. Numer. Anal., 15 (1978), pp. 977–992.
- [11] G. GOLUB AND C. VAN LOAN, *Matrix Computations*, 3rd ed., Johns Hopkins University Press, Baltimore, MD, 1996.
- [12] G. H. GOLUB, M. T. HEATH, AND G. WAHBA, *Generalized cross-validation as a method for choosing a good ridge parameter*, Technometrics, 21 (1979), pp. 215–223.
- [13] E. HABER, *Numerical Strategies for the Solution of Inverse Problems*, Ph.D. thesis, University of British Columbia, 1997.
- [14] E. HABER, U. M. ASCHER, AND D. OLDENBURG, *On optimization techniques for solving nonlinear inverse problems*, Inverse Problems, 16 (2000), pp. 1263–1280.
- [15] E. HABER AND D. OLDENBURG, *A GCV based method for nonlinear ill-posed problems*, Comput. Geosci., 4 (2000), pp. 41–63.
- [16] M. HANKE, *A regularizing Levenberg-Marquardt scheme, with applications to inverse groundwater filtration problems*, Inverse Problems, 13 (1997), pp. 79–95.
- [17] P. C. HANSEN, *Rank-Deficient and Discrete Ill-Posed Problems: Numerical Aspects of Linear Inversion*, SIAM, Philadelphia, 1998.
- [18] M. KILMER, E. MILLER, M. ENRIQUEZ, AND D. BOAS, *Cortical constraint method for diffuse optical brain imaging*, Proc. SPIE, 5559 (2004), pp. 381–391.
- [19] M. E. KILMER AND E. DE STURLER, *Recycling subspace information for diffuse optical tomography*, SIAM J. Sci. Comput., 27 (2006), pp. 2140–2166.
- [20] M. E. KILMER, E. MILLER, A. BARBARO, AND D. BOAS, *Three-dimensional shape-based imaging of absorption perturbation for diffuse optical tomography*, Appl. Optics, 42 (2003), pp. 3129–3144.
- [21] M. E. KILMER AND D. P. O’LEARY, *Choosing regularization parameters in iterative methods for ill-posed problems*, SIAM J. Matrix Anal. Appl., 22 (2001), pp. 1204–1221.
- [22] J. NOCEDAL AND S. J. WRIGHT, *Numerical Optimization*, Springer Series in Operations Research, Springer, Berlin, Heidelberg, New York, 1999.
- [23] M. ROJAS AND D. C. SORENSEN, *A trust-region approach to the regularization of large-scale discrete forms of ill-posed problems*, SIAM J. Sci. Comput., 23 (2002), pp. 1842–1860.
- [24] X.-C. TAI AND T. F. CHAN, *A survey on multiple level set methods with applications for identifying piecewise constant functions*, Internat. J. Numer. Anal. Model., 1 (2004), pp. 25–47.
- [25] K. VAN DEN DOEL AND U. ASCHER, *On level set regularization for highly ill-posed distributed parameter estimation problems*, J. Comput. Phys., 216 (2006), pp. 707–723.
- [26] K. VAN DEN DOEL AND U. ASCHER, *Dynamic level set regularization for large distributed parameter estimation problems*, Inverse Problems, 23 (2007), pp. 1271–1288.
- [27] K. VAN DEN DOEL, U. M. ASCHER, AND A. L. AO, *Multiple level sets for piecewise constant surface reconstruction in highly ill-posed problems*, J. Sci. Comput., 43 (2010), pp. 44–66.
- [28] C. R. VOGEL, *Computational Methods for Inverse Problems*, SIAM, Philadelphia, 2002.