RISE: An Incremental Trust-Region Method for Robust Online Sparse Least-Squares Estimation

David M. Rosen, *Student Member, IEEE*, Michael Kaess, *Member, IEEE*, and John J. Leonard, *Fellow, IEEE*

***Abstract*—Many point estimation problems in robotics, computer vision, and machine learning can be formulated as instances of the general problem of minimizing a sparse nonlinear sum-of-squares objective function. For inference problems of this type, each input datum gives rise to a summand in the objective function, and therefore performing *online* inference corresponds to solving a *sequence* of sparse nonlinear least-squares minimization problems in which additional summands are added to the objective function over time. In this paper, we present *Robust Incremental least-Squares Estimation (RISE)*, an incrementalized version of the Powell's Dog-Leg numerical optimization method suitable for use in online sequential sparse least-squares minimization. As a trust-region method, RISE is naturally robust to objective function nonlinearity and numerical ill-conditioning and is provably globally convergent for a broad class of inferential cost functions (twice-continuously differentiable functions with bounded sublevel sets). Consequently, RISE maintains the speed of current state-of-the-art online sparse least-squares methods while providing superior reliability.**

***Index Terms*—Computer vision, machine learning, online estimation, simultaneous localization and mapping (SLAM), sparse least-squares minimization.**

I. Introduction

MANY point estimation problems in robotics, computer vision, and machine learning can be formulated as instances of the general problem of minimizing a sparse nonlinear sum-of-squares objective function; for example, the archetypal problems of full (smoothing) simultaneous localization and mapping (SLAM) [1] (in robotics), bundle adjustment (BA) [2], [3] (in computer vision), and sparse (kernel) regularized least-squares classification and regression [4], [5] (in machine learning) all belong to this class. For inference problems of this type, each input datum gives rise to a summand in the objective function, and therefore performing *online* inference (in which the data are collected sequentially and the estimate updated after the incorporation of each new datum) corresponds to solving a *sequence* of sparse least-squares minimization problems in which additional summands are added to the objective function over time.

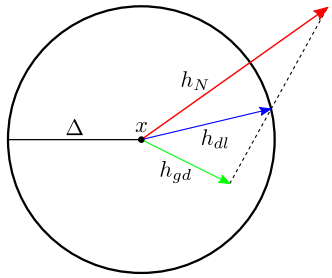


Fig. 1. The Powell's Dog-Leg update step h_{dl} is obtained by interpolating the (possibly approximate) Newton step h_N and the gradient descent step h_{gd} using a trust region of radius Δ centered on the current iterate x. By adapting Δ online in response to the observed performance of the Newton steps near x, the algorithm is able to combine the rapid end-stage convergence speed of Newton-type methods with the reliability of gradient descent.

In practice, these online inference problems are often solved by computing each estimate in the sequence as the solution of an independent minimization problem using standard sparse least-squares techniques (most commonly Levenberg–Marquardt [6]–[8]). While this approach is general and produces good results, it is computationally expensive and does not exploit the sequential structure of the underlying inference problem; this limits its utility in real-time online applications, where speed is crucial.

More sophisticated solutions achieve faster computation by directly exploiting the sequentiality of the online inference problem. The canonical example is online gradient descent, which is attractive for its robustness, simplicity, and low memory and per-iteration computational costs, but its first-order rate can lead to painfully slow convergence [8]. Alternatively, Kaess *et al.* have developed *incremental smoothing and mapping (iSAM)* [9], [10], which exploits recent algorithmic advances in sparse numerical linear algebra to implement an efficient incrementalized version of the Gauss–Newton method [8] for use in online sparse least-squares minimization. This incremental approach, together with the Gauss–Newton method's superlinear convergence rate, enables iSAM to achieve computational speeds unmatched by iterated batch techniques. However, the Gauss–Newton method can exhibit poor (even globally divergent) behavior when applied to objective functions with significant nonlinearity [11], which restricts the class of problems to which iSAM can be reliably applied. To date, the development of a fully incremental online sparse least-squares solver that combines the robustness of gradient descent with the superlinear convergence rate of Newton-type methods has remained an outstanding problem.

Manuscript received August 5, 2013; revised January 9, 2014; accepted May 1, 2014. This paper was recommended for publication by Associate Editor J. A. Castellanos and Editor D. Fox upon evaluation of the reviewers' comments. This work was supported in part by the Office of Naval Research under Grant N00014-12-1-0093, Grant N00014-11-1-0688, Grant N00014-06-1-0043, and Grant N00014-10-1-0936, and by the Air Force Research Laboratory under Contract FA8650-11-C-7137.

D. M. Rosen and J. J. Leonard are with the Computer Science and Artificial Intelligence Laboratory, Massachusetts Institute of Technology, Cambridge, MA 02139 USA (e-mail: dmrosen@mit.edu; jleonard@mit.edu).

M. Kaess is with the Robotics Institute, Carnegie Mellon University, Pittsburgh, PA 15213 USA (e-mail: kaess@cmu.edu).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TRO.2014.2321852

To that end, in this paper, we present *Robust Incremental least-Squares Estimation (RISE)*, an incrementalized version of the Powell's Dog-Leg numerical optimization algorithm [8], [12] suitable for use in online sequential sparse least-squares minimization. As a trust-region method (Fig. 1), Powell's Dog-Leg is naturally robust to objective function nonlinearity and numerical ill-conditioning and enjoys excellent global convergence properties [13]–[15]; furthermore, it is known to perform significantly faster than Levenberg–Marquardt in batch sparse least-squares minimization while obtaining solutions of comparable quality [16]. By exploiting iSAM's preexisting functionality to incrementalize the computation of the dog-leg step, RISE maintains the speed of current state-of-the-art online sparse least-squares solvers while providing superior robustness to objective function nonlinearity and numerical ill-conditioning.

The rest of this paper is organized as follows. In the next section, we formulate the sequential sparse least-squares minimization problem and discuss its connections to online inference. In Section III, we review the class of Newton-type optimization methods, focusing in particular on the Gauss–Newton method and its incrementalization to produce iSAM. Section IV introduces the general class of trust-region methods, paying particular attention to Powell's Dog-Leg. Here, we derive the *indefinite Gauss–Newton–Powell's Dog-Leg (IGN-PDL)* algorithm (an extension of Powell's Dog-Leg with Gauss–Newton steps to the case of indefinite Jacobians), analyze its robustness with respect to objective function nonlinearity and numerical ill-conditioning, and establish sufficient conditions for its global convergence (Theorem 3). We then derive the RISE and RISE2 algorithms in Section V by incrementalizing IGN-PDL with the aid of iSAM. We contextualize RISE with a discussion of related work in Section VI and evaluate its performance in Section VII on standard six-degree-of-freedom (6-DOF) pose-graph SLAM benchmarks and on a real-world visual mapping task using a calibrated monocular camera. Finally, Section VIII concludes with a summary of this paper's contributions and a discussion of future research directions.

II. Problem Formulation

We are interested in the general problem of obtaining a point estimate $x^* \in \mathbb{R}^n$ of some quantity of interest X as the solution of a sparse nonlinear least-squares problem

$$\min_{x \in \mathbb{R}^n} S(x)$$
$$S(x) = \sum_{i=1}^{m} r_i(x)^2 = \|r(x)\|^2 \tag{1}$$

for $r : \mathbb{R}^n \to \mathbb{R}^m$ with $m \geq n$. Problems of this form frequently arise in probabilistic inference in the context of maximum likelihood (ML) or maximum *a posteriori* (MAP) parameter estimation; indeed, performing ML or MAP estimation over *any* probability distribution $p : \mathbb{R}^n \to \mathbb{R}^+$ whose factor graph representation $\mathcal{G} = (\mathcal{F}, \mathcal{X}, \mathcal{E})$ [17] is sparse and whose factors are positive and bounded is equivalent to solving a sparse least-squares problem of the form (1) in which each summand r_i corresponds to a factor p_i of p [18]. Given the ubiquity of these models, robust and computationally efficient methods for solving (1) are thus of significant practical import.

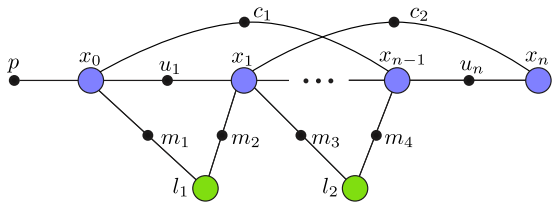


Fig. 2. Factor graph formulation of the full (smoothing) SLAM problem. Here, variable nodes are shown as large circles and factor nodes as small solid circles. The variables consist of robot poses x and landmark positions l, and the factors are odometry measurements u, a prior p, loop closing constraints c, and landmark measurements m. Each of the factors corresponds to a summand r_i in (1). In the online case, as the robot explores previously unseen areas, new variable nodes (i.e., robot positions and landmarks) and factor nodes (measurements) are added to this graph over time; the corresponding online inference problem is then given by (2).

In the case of online inference, the input data are collected sequentially, and we wish to obtain a revised estimate for X after the incorporation of each datum. Since each input datum gives rise to a summand in (1), online inference corresponds to solving the *sequence* of sparse least-squares problems

$$\min_{x_t \in \mathbb{R}^{n_t}} S^{(t)}(x_t)$$
$$S^{(t)}(x_t) = \sum_{i=1}^{m_t} r_i\left(x_i\right)^2 = \left\|r^{(t)}(x_t)\right\|^2 \tag{2}$$

for $r^{(t)} : \mathbb{R}^{n_t} \to \mathbb{R}^{m_t}$ and $t = 1, 2, \ldots$, where:

1) $m_t, n_t \in \mathbb{N}^+$ are monotonically nondecreasing in t.
2) $m_t \geq n_t$ for all t.
3) $x_i \in \mathbb{R}^{n_i}$ for all i and $x_i \subseteq x_j$ for all $i \leq j$.

Condition 1 above expresses the fact that the summation in (2) evolves over time only through the addition of new terms. Condition 2 is necessary in order for the minimization problem in (2) to have a unique solution. Condition 3 formalizes the idea that we also allow the vector of states X that we wish to estimate to be *augmented* online by the addition of new quantities of interest (for example, as in the case of robotic mapping when exploring previously unseen areas; cf., Fig. 2).

Our goal in this paper is to develop a fully incremental algorithm capable of robustly solving online sparse least-squares minimization problems of the form (2) in real time.

III. Review of Newton-Type Optimization Methods and Incremental Smoothing and Mapping

The RISE algorithm that we develop in Section V exploits iSAM's incremental computation of the Gauss–Newton step in order to solve the sequential sparse least-squares problem (2) efficiently in the online case. In this section, we review the general class of Newton-type optimization methods, their specialization to the Gauss–Newton method, and Gauss–Newton's incremental implementation in iSAM.

A. Newton's Method and Its Approximations

Newton's method [8], [11] is an iterative numerical method for estimating a solution x^* of the general nonlinear minimization problem

$$\min_{x\in\mathbb{R}^n} f(x), \quad f \in C^2(\mathbb{R}^n). \tag{3}$$

Given an initial estimate $x^{(i)}$ for x^*, the function f is locally approximated at $x^{(i)}$ by its second-order Taylor expansion $q^{(i)}$:

$$q^{(i)}(x^{(i)}+h) = f(x^{(i)}) + \nabla f(x^{(i)})^T h + \frac{1}{2}h^T \frac{\partial^2 f}{\partial x^2}(x^{(i)})h \tag{4}$$

and a revised estimate

$$x^{(i+1)} = x^{(i)} + h_N^{(i)} \tag{5}$$

is computed by choosing the *Newton step* $h_N^{(i)}$ to be any increment to $x^{(i)}$ that minimizes the value of the local approximation (4):

$$h_N^{(i)} \in \operatorname*{argmin}_{h\in\mathbb{R}^n} q^{(i)}\left(x^{(i)}+h\right). \tag{6}$$

Provided that $\frac{\partial^2 f}{\partial x^2}(x^{(i)}) > 0$, there is a unique minimizer $h_N^{(i)}$ in (6), which is determined as the solution of

$$\frac{\partial^2 f}{\partial x^2}\left(x^{(i)}\right) h_N^{(i)} = -\nabla f(x^{(i)}). \tag{7}$$

Assuming that each of the steps $h_N^{(i)}$ in (6) exists, Newton's method consists of iteratively applying equations (4), (6), and (5), in that order, to generate a sequence of estimates $x^{(0)}, x^{(1)}, \ldots$, for x^* until some stopping criterion is satisfied.

Newton's method has several attractive theoretical properties, in particular a fast (quadratic) convergence rate when initialized with an estimate $x^{(0)}$ that is close to a minimizer x^* of a sufficiently regular function f [8]. However, in application, it may not always be practical or computationally feasible to evaluate the gradient $\nabla f(x^{(i)})$ or Hessian $\frac{\partial^2 f}{\partial x^2}(x^{(i)})$ in the quadratic model (4) at every iteration (depending upon the dimension n and analytical complexity of the function f). In these cases, the local model (4) is often relaxed to

$$q^{(i)}(x^{(i)}+h) = f(x^{(i)}) + \left(g^{(i)}\right)^T h + \frac{1}{2}h^T B^{(i)} h \tag{8}$$

where $g^{(i)} \in \mathbb{R}^n$ and $B^{(i)} \in \mathbb{R}^{n\times n}$ symmetric are chosen such that

$$g^{(i)} \approx \nabla f(x^{(i)}), \quad B^{(i)} \approx \frac{\partial^2 f}{\partial x^2}(x^{(i)}) \tag{9}$$

and the corresponding update step $h_N^{(i)}$ is computed as a solution of

$$B^{(i)} h_N^{(i)} = -g^{(i)}. \tag{10}$$

The selection of different methods for performing the approximations (9) gives rise to a broad class of optimization algorithms collectively referred to as *Newton-type* or *approximate Newton* methods. With a careful choice of approximation scheme in (9), it is possible to preserve many of the desirable properties of Newton's method (most importantly a superlinear end-stage convergence rate), while dramatically reducing the computational burden of computing the update steps $h_N^{(i)}$.

B. Gauss–Newton Method

The *Gauss–Newton method* [8], [11] is an approximate Newton method for solving the minimization problem (3) in the special case (1) in which the objective function is a sum of squared nonlinear terms. In this case, we have

$$\frac{\partial S}{\partial x_j} = 2\sum_{i=1}^m r_i \frac{\partial r_i}{\partial x_j} \tag{11a}$$

$$\frac{\partial^2 S}{\partial x_j \partial x_k} = 2\sum_{i=1}^m \frac{\partial r_i}{\partial x_j}\frac{\partial r_i}{\partial x_k} + r_i \frac{\partial^2 r_i}{\partial x_j \partial x_k} \tag{11b}$$

and the Gauss–Newton method is obtained as an approximate Newton method by ignoring the effects of the second-order partial derivatives of r when forming the approximate Hessian $B^{(i)}$ in (9):

$$\frac{\partial^2 S}{\partial x_j \partial x_k} \approx 2\sum_{i=1}^m \frac{\partial r_i}{\partial x_j}\frac{\partial r_i}{\partial x_k} \tag{12}$$

(the exact gradient $\nabla S(x^{(i)})$ corresponding to (11a) is used for $g^{(i)}$). Using the function $r : \mathbb{R}^n \to \mathbb{R}^m$, we can write this approximation more conveniently in matrix notation as

$$g^{(i)} = 2J(x^{(i)})^T r(x^{(i)}) \tag{13a}$$

$$B^{(i)} = 2J(x^{(i)})^T J(x^{(i)}) \tag{13b}$$

where $J(x^{(i)})$ denotes the Jacobian of r evaluated at $x^{(i)}$:

$$J(x^{(i)}) = \left.\frac{\partial r}{\partial x}\right|_{x=x^{(i)}} \in \mathbb{R}^{m\times n}. \tag{14}$$

We observe that the approximation (12) is equivalent to the assumption that r is locally linear. Indeed, substitution of (13) into (8) produces

$$\begin{aligned} q^{(i)}(x^{(i)}+h) &= r(x^{(i)})^T r(x^{(i)}) + 2r(x^{(i)})^T J(x^{(i)})h \\ &\quad + h^T J(x^{(i)})^T J(x^{(i)})h \\ &= \left\| L^{(i)}(x^{(i)}+h) \right\|^2 \end{aligned} \tag{15}$$

where

$$L^{(i)}(x^{(i)}+h) = r(x^{(i)}) + J(x^{(i)})h \tag{16}$$

is the first-order Taylor expansion (i.e., linearization) of r about $x^{(i)}$. Consequently, by virtue of (11b), (12), and (15), we expect the Gauss–Newton method to produce the best results when applied to functions r that have relatively modest nonlinearities (as quantified by the magnitudes of their second partial derivatives) and small magnitudes $\|r\|$.

To compute the Gauss–Newton step $h_{\mathrm{GN}}^{(i)}$, we observe that if $J(x^{(i)})$ is full-rank, then $B^{(i)}$ as defined in (13b) is positive definite, and thus, $h_{\mathrm{GN}}^{(i)}$ is uniquely determined by

$$2J(x^{(i)})^T J(x^{(i)}) h_{\mathrm{GN}}^{(i)} = -2J(x^{(i)})^T r(x^{(i)}) \tag{17}$$

following (10). Letting

$$Q^{(i)}\begin{pmatrix} R^{(i)} \\ 0 \end{pmatrix} = J(x^{(i)}) \tag{18}$$

be a QR decomposition [19] of the Jacobian $J(x^{(i)})$ and

$$\begin{pmatrix} d^{(i)} \\ e^{(i)} \end{pmatrix} = \left(Q^{(i)}\right)^T r(x^{(i)}) \tag{19}$$

for $d^{(i)} \in \mathbb{R}^n$ and $e^{(i)} \in \mathbb{R}^{m-n}$, we can simplify (17) to

$$R^{(i)} h_{\mathrm{GN}}^{(i)} = -d^{(i)}. \tag{20}$$

Since $R^{(i)}$ is upper-triangular, (20) can be efficiently solved for $h_{\mathrm{GN}}^{(i)}$ by back-substitution.

C. Incremental Smoothing and Mapping: Incrementalizing Gauss–Newton

As shown in Section II, the arrival of new data corresponds to *augmenting* the function $r = r_{\mathrm{old}} : \mathbb{R}^n \to \mathbb{R}^m$ on the right-hand side of (1) to the function

$$\begin{aligned} \bar{r} &: \mathbb{R}^{n+n_{\mathrm{new}}} \to \mathbb{R}^{m+m_{\mathrm{new}}} \\ \bar{r}\left(x_{\mathrm{old}}, x_{\mathrm{new}}\right) &= \begin{pmatrix} r_{\mathrm{old}}\left(x_{\mathrm{old}}\right) \\ r_{\mathrm{new}}\left(x_{\mathrm{old}}, x_{\mathrm{new}}\right) \end{pmatrix} \end{aligned} \tag{21}$$

where $r_{\mathrm{new}} : \mathbb{R}^{n+n_{\mathrm{new}}} \to \mathbb{R}^{m_{\mathrm{new}}}$ is the set of new measurement functions, and $x_{\mathrm{new}} \in \mathbb{R}^{n_{\mathrm{new}}}$ is the set of new system variables introduced as a result of the new observations.

In the naïve application of the Gauss–Newton algorithm of Section III-B, the solution $x^* = (x^*_{\mathrm{old}}, x^*_{\mathrm{new}})$ for the augmented least-squares problem determined by (21) would be found by performing Gauss–Newton iterations until convergence. However, in the context of the sequential estimation problem (2), we already have a good estimate $\hat{x}_{\mathrm{old}}$ for the values of the previously introduced variables x_{old}, obtained by solving the least-squares minimization problem (1) prior to the introduction of the new measurement functions r_{new}. Thus, given any good initialization $\hat{x}_{\mathrm{new}}$ for the newly introduced system variables x_{new}, $\hat{x} = (\hat{x}_{\mathrm{old}}, \hat{x}_{\mathrm{new}})$ provides a good initialization for the augmented state $x = (x_{\mathrm{old}}, x_{\mathrm{new}})$ for the Gauss–Newton algorithm.

Furthermore, since we generally expect the initial estimate $\hat{x}$ to be close to the true minimizing value x^*, it is not usually necessary to iterate the Gauss–Newton algorithm until convergence after integration of *every* new observation; instead, a *single* Gauss–Newton step is computed and used to correct the initial estimate $\hat{x}$. The advantage to this approach is that it avoids having to recompute the Jacobian $\bar{J}(\hat{x})$ for $\bar{r}$ and its QR decomposition anew each time new observations arrive; instead, iSAM efficiently obtains $\bar{J}(\hat{x})$ together with its QR decomposition by *updating* the Jacobian $J(\hat{x}_{\mathrm{old}})$ and its QR decomposition, as we now describe.

Letting $x = (x_{\mathrm{old}}, x_{\mathrm{new}})$, the Jacobian $\bar{J}(x)$ for the augmented system (21) can be decomposed into block form as

$$\bar{J}(x) = \frac{\partial \bar{r}}{\partial x} = \begin{pmatrix} \dfrac{\partial r_{\mathrm{old}}}{\partial x_{\mathrm{old}}} & 0 \\ \dfrac{\partial r_{\mathrm{new}}}{\partial x_{\mathrm{old}}} & \dfrac{\partial r_{\mathrm{new}}}{\partial x_{\mathrm{new}}} \end{pmatrix} = \begin{pmatrix} J\left(x_{\mathrm{old}}\right) & 0 \\ J_{\mathrm{new}}(x) & \end{pmatrix} \tag{22}$$

where

$$J\left(x_{\mathrm{old}}\right) = \frac{\partial r_{\mathrm{old}}}{\partial x_{\mathrm{old}}} \in \mathbb{R}^{m \times n} \tag{23}$$

is the Jacobian of the previous function r_{old}, and

$$J_{\mathrm{new}}(x) = \frac{\partial r_{\mathrm{new}}}{\partial x} \in \mathbb{R}^{m_{\mathrm{new}} \times (n+n_{\mathrm{new}})} \tag{24}$$

is the Jacobian of the new measurement function r_{new}. Letting

$$J\left(\hat{x}_{\mathrm{old}}\right) = \begin{pmatrix} Q_1 & Q_2 \end{pmatrix}\begin{pmatrix} R \\ 0 \end{pmatrix} \tag{25}$$

be a QR decomposition for $J(\hat{x}_{\mathrm{old}})$, where $Q_1 \in \mathbb{R}^{m \times n}$ and $Q_2 \in \mathbb{R}^{m \times (m-n)}$, we have

$$\begin{pmatrix} Q_1 & 0 & Q_2 \\ 0 & I & 0 \end{pmatrix}\begin{pmatrix} R & 0 \\ J_{\mathrm{new}}(\hat{x}) & \\ 0 & \end{pmatrix} = \begin{pmatrix} Q_1 R & 0 \\ J_{\mathrm{new}}(\hat{x}) & \end{pmatrix} = \bar{J}(\hat{x}) \tag{26}$$

by (22) and (25), which gives a partial QR decomposition of $\bar{J}(\hat{x})$. This partial decomposition can be completed by using Givens rotations to zero out the remaining nonzero elements below the main diagonal. Let $G \in \mathbb{R}^{(n+m_{\mathrm{new}}) \times (n+m_{\mathrm{new}})}$ denote a matrix of Givens rotations such that

$$G\begin{pmatrix} R & 0 \\ J_{\mathrm{new}}(\hat{x}) & \end{pmatrix} = \begin{pmatrix} \bar{R} \\ 0 \end{pmatrix} \tag{27}$$

where $\bar{R} \in \mathbb{R}^{(n+n_{\mathrm{new}}) \times (n+n_{\mathrm{new}})}$ is upper-triangular. Defining

$$\bar{G} = \begin{pmatrix} G & 0 \\ 0 & I \end{pmatrix}, \qquad \bar{Q} = \begin{pmatrix} Q_1 & 0 & Q_2 \\ 0 & I & 0 \end{pmatrix}\bar{G}^T \tag{28}$$

(so that $\bar{G}, \bar{Q} \in \mathbb{R}^{(m+m_{\mathrm{new}}) \times (m+m_{\mathrm{new}})}$ are also orthogonal), (26)–(28) show that

$$\bar{J}(\hat{x}) = \bar{Q}\begin{pmatrix} \bar{R} \\ 0 \end{pmatrix} \tag{29}$$

is a QR decomposition for the augmented Jacobian $\bar{J}(\hat{x})$. Now, we can use (19) and (20) to compute the Gauss–Newton step $\bar{h}_{\mathrm{GN}}$ for the augmented system:

$$\begin{aligned} \bar{Q}^T \bar{r}(\hat{x}) &= \begin{pmatrix} G & 0 \\ 0 & I \end{pmatrix}\begin{pmatrix} Q_1^T & 0 \\ 0 & I \\ Q_2^T & 0 \end{pmatrix}\begin{pmatrix} r_{\mathrm{old}}(\hat{x}_{\mathrm{old}}) \\ r_{\mathrm{new}}(\hat{x}) \end{pmatrix} \\ &= \begin{pmatrix} G & 0 \\ 0 & I \end{pmatrix}\begin{pmatrix} d_{\mathrm{old}} \\ r_{\mathrm{new}}(\hat{x}) \\ e_{\mathrm{old}} \end{pmatrix} \end{aligned}$$

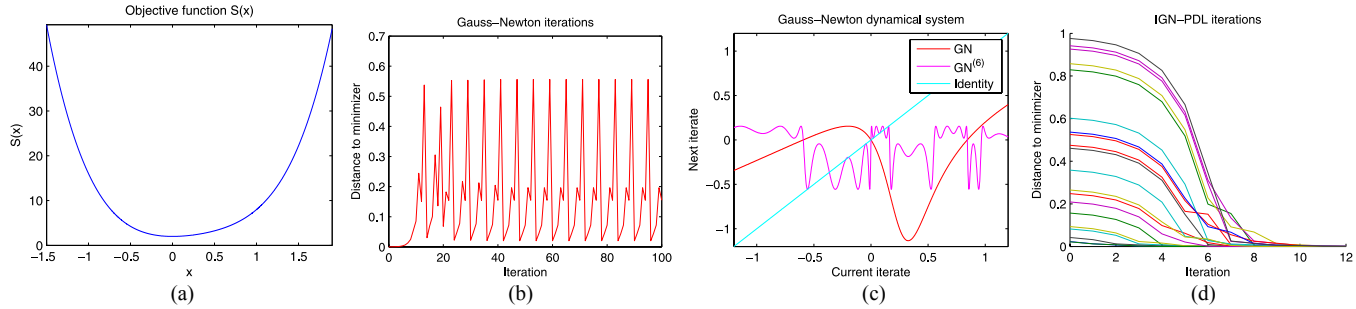


Fig. 3. Failure of the Gauss–Newton method. (a) Graph of the sum-of-squares objective function $S(x)$ determined by (33) for $\lambda = -2$. This function is C^∞ and strictly convex and has a unique global minimizer at $x^* = 0$ with $\frac{\partial^2 S}{\partial x^2}(x^*) > 0$. (b) Distance from the ith Gauss–Newton iterate $x^{(i)}$ to the global minimizer x^* for the first 100 iterations of a sequence initialized with $x^{(0)} = 10^{-4}$. Note that the Gauss–Newton method initially drives the estimate $x^{(i)}$ *away* from the global minimum x^* before converging to a six-periodic orbit. (c) Behavior of the Gauss–Newton update rule $GN : x^{(i)} \mapsto x^{(i+1)}$ and its sixfold composition $GN^{(6)}$ considered as dynamical systems. The graph of GN intersects the graph of the identity mapping at the minimizer $x^* = 0$, showing that x^* is a fixed point of GN (as we would hope); however, $\nabla GN(x^*) = -2$ so that $|\nabla GN(x^*) - 1| = 3 > 1$, and therefore, this fixed point is *repelling* [20, Ch. 10]. The intersections of the graph of $GN^{(6)}$ with the graph of the identity give the six-periodic points of GN, including the members of the six-periodic orbit to which the Gauss–Newton iterates converge in (b). This example illustrates the Gauss–Newton method's sensitivity to nonlinearity even when applied to well-behaved functions with good initialization. (d) In contrast, the IGN-PDL algorithm of Section IV-C is globally convergent when applied to this function (cf., Theorem 3). This plot shows the distance from the ith IGN-PDL iterate $x^{(i)}$ to the minimizer x^* for 20 sequences with initializations $x^{(0)}$ sampled uniformly randomly from $[-1, 1]$ and parameters $\Delta^{(0)} = 0.01$, $\eta_1 = 0.25$, $\eta_2 = 0.75$, $\gamma_1 = 0.5$, $\gamma_2 = 2$. The 12th iterates $x^{(12)}$ of these sequences are all within 3×10^{-4} of the minimizer x^*. Strong global convergence properties and robustness to highly nonlinear and ill-conditioned systems are general features of this algorithm (cf., Section IV-D).

$$= \begin{pmatrix} \bar{d} \\ e_{\text{new}} \\ e_{\text{old}} \end{pmatrix} = \begin{pmatrix} \bar{d} \\ \bar{e} \end{pmatrix} \tag{30}$$

where

$$\begin{pmatrix} \bar{d} \\ e_{\text{new}} \end{pmatrix} = G \begin{pmatrix} d_{\text{old}} \\ r_{\text{new}}(\hat{x}) \end{pmatrix} \tag{31}$$

for $\bar{d} \in \mathbb{R}^{n+n_{\text{new}}}$. The Gauss–Newton step $\bar{h}_{\text{GN}}$ used to correct the estimate $\hat{x}$ for the augmented system is then computed as the solution of

$$\bar{R}\bar{h}_{\text{GN}} = -\bar{d}. \tag{32}$$

Equations (24), (27), and (31) show how to obtain the $\bar{R}$ factor of the QR decomposition of $\bar{J}(\hat{x})$ and the corresponding linear system (32) by *updating* the R factor and linear system for the previous Jacobian $J(\hat{x}_{\text{old}})$ using Givens rotations. Since the updated factor $\bar{R}$ and the new right-hand side vector $\bar{d}$ are obtained by applying G directly to the augmented factor R in (27) and the augmented right-hand side vector d in (31), it is not necessary to explicitly form the orthogonal matrix $\bar{Q}$ in (28), nor is it necessary to form the matrix G explicitly either; instead, the appropriate individual Givens rotations can be applied sequentially directly to the matrix in (27) and the right-hand side vector in (31). Under the assumption of sparsity, only a few elements of the Jacobian $J_{\text{new}}(\hat{x})$ will be nonzero, and therefore, only a small number of Givens rotations will be needed to perform the update in (27). Obtaining the linear system (32) by this method is, thus, a computationally efficient operation.

Finally, we observe that while relinearization is not needed after *every* new observation, the system should be periodically relinearized about its corrected estimate in order to perform a full Gauss–Newton iteration and obtain a better estimate of the local minimum (this is particularly true after observations that are likely to significantly alter the estimates of system variables). When relinearizing the system about the corrected estimate, the incremental updating method outlined above is no longer applicable; instead, the QR factorization of the Jacobian needs to be recomputed anew. While this is an expensive batch operation, the factorization step can be combined with a variable reordering step [21] in order to reduce the fill-in in the resulting factor R, thereby maintaining sparsity and speeding up subsequent incremental computations.

IV. From Gauss–Newton to Powell's Dog-Leg

The incrementalized Gauss–Newton method outlined in Section III-C is computationally efficient, straightforward to implement, and enjoys rapid (up to quadratic [11, p. 113]) convergence near the minimum. However, the assumption of the local linearity of r [which underlies the approximations (13b) and (15)] means that the Gauss–Newton method can exhibit poor behavior when either r itself or its second partial derivatives $\frac{\partial^2 r}{\partial x_j \partial x_k}$ have large magnitude. Indeed, convergence of the Gauss–Newton method is not guaranteed, *not even locally*; this is in marked contrast with the behavior of Newton's method, which (although not globally convergent) can at least guarantee fast *local* convergence for a class of functions that is not excessively restrictive in practice (cf., [8, Th. 3.5]). Furthermore, it is not difficult to construct simple (even quadratic) examples of r where the sequence of Gauss–Newton iterates $\{x^{(i)}\}_{i=1}^{\infty}$ is in fact *globally divergent* (i.e., fails to converge when initialized with *any* point $x^{(0)} \neq x^*$). For example, consider the

minimization (1) determined by

$$r : \mathbb{R} \to \mathbb{R}^2$$
$$r(x) = \begin{pmatrix} x+1 \\ \lambda x^2 + x - 1 \end{pmatrix}. \tag{33}$$

For $\lambda = -2$, the function $S(x)$ is C^∞ and strictly convex and has a single global minimum at $x^* = 0$ with $\frac{\partial^2 S}{\partial x^2}(x^*) > 0$; hence, it satisfies all of the standard regularity conditions customarily assumed in numerical optimization (cf., e.g., [8] and [11]). Nevertheless, the Gauss–Newton algorithm is in fact *globally divergent* in this case [11, p. 113]. This lack of robustness even under ideal conditions (a well-behaved objective function with strong regularity properties and good initialization; cf., Fig. 3) is a serious shortcoming of the Gauss–Newton method.

To address this shortcoming, in this paper, we adopt the *Powell's Dog-Leg* algorithm [8], [12] as the method of choice for performing the sparse least-squares minimization (1). This algorithm combines the superlinear end-stage convergence speed of the Newton-type methods with the excellent global convergence properties [13]–[15] of gradient descent approaches. Indeed, when applied to sparse least-squares minimization problems, Powell's Dog-Leg performs significantly faster than Levenberg–Marquardt (the current method of choice in the robotics and computer vision communities) while maintaining comparable levels of accuracy [16].

A. Trust–Region Approach

As shown in Section III-A, in each iteration, Newton's method constructs a local model $q^{(i)}$ for the objective function f on a neighborhood $U^{(i)}$ of the current estimate $x^{(i)}$ and then determines an update step $h_N^{(i)}$ that minimizes $q^{(i)}$ in place of f. However, the model $q^{(i)}$ in (4) is constructed using information about f's first- and second-order derivatives at $x^{(i)}$, which depend only upon the *local* behavior of f near $x^{(i)}$. Consequently, while $q^{(i)}$ is a good approximation of f near $x^{(i)}$ (in a sense made precise by Taylor's Theorem), its global behavior may be quite different from f's. This can become problematic if the update step $h_N^{(i)}$ computed using $q^{(i)}$ leaves the region $U^{(i)}$ in which f is well approximated by $q^{(i)}$.

Trust-region methods [8] address this hazard by maintaining an explicit *trust-region*, an open ball of radius $\Delta^{(i)}$ centered on the current estimate $x^{(i)}$ within which f is considered well approximated by the local model $q^{(i)}$. The trust-region update step $h_{\mathrm{tr}}^{(i)}$ is then obtained as an increment to $x^{(i)}$ that minimizes the value of $q^{(i)}$, *subject to the condition that the update step does not leave the trust-region*:

$$h_{\mathrm{tr}}^{(i)} \in \operatorname*{argmin}_{\|h\| \le \Delta^{(i)}} q^{(i)}(x^{(i)} + h). \tag{34}$$

The following theorem gives necessary and sufficient optimality conditions characterizing the trust-region step $h_{\mathrm{tr}}^{(i)}$ in (34) when $q^{(i)}$ is a quadratic model (as in approximate Newton methods). Its proof follows from a straightforward application of the Karush–Kuhn–Tucker first-order optimality conditions together with the enforcement of the necessary second-order conditions for a minimizer (cf., [8, Chs. 4 and 12]).

Algorithm 1 Updating the Trust-Region Radius Δ

```
procedure UPDATE_DELTA(ρ, Δ, η1, η2, γ1, γ2)
    if ρ ≥ η2 then
        Δ ← γ2 Δ
    else if ρ < η1 then
        Δ ← γ1 Δ
    end if
    return Δ
end procedure
```

Algorithm 2 The Trust-Region Minimization Method

```
procedure TRUST-REGION(f, x0, Δ0, η1, η2, γ1, γ2)
    Initialize x ← x0, Δ ← Δ0.
    repeat
        Construct local model q for f about x.
        Compute trust-region step h_tr by solving (34).
        Set x_proposed ← (x + h_tr).
        Compute ρ using (38).
        if ρ ≥ η1 then
            Set x ← x_proposed.
        end if
        Δ ← UPDATE_DELTA(ρ, Δ, η1, η2, γ1, γ2).
    until (stopping criteria)
    return x
end procedure
```

Theorem 1 (Optimality conditions for the trust-region step): Let $f, g \in \mathbb{R}^n$, let $B \in \mathbb{R}^{n \times n}$ be symmetric, and define

$$q : \mathbb{R}^n \to \mathbb{R}$$
$$q(h) = f + g^T h + \frac{1}{2} h^T B h. \tag{35}$$

Then, $h^* \in \mathbb{R}^n$ is a solution of the constrained minimization

$$\min_{h \in \mathbb{R}^n} q(h) \quad \mathbf{s.t.}\ \|h\| \le \Delta \tag{36}$$

for $\Delta > 0$ if and only if $\|h^*\| \le \Delta$ and there exists $\lambda^* \ge 0$ such that the following conditions are satisfied:

$$(B + \lambda^* I) h^* = -g \tag{37a}$$
$$\lambda^* (\Delta - \|h^*\|) = 0 \tag{37b}$$
$$(B + \lambda^* I) \ge 0. \tag{37c}$$

As the trust-region method proceeds, the radius of the trust region $\Delta^{(i)}$ is varied adaptively according to the *gain ratio*:

$$\rho^{(i)} = \frac{\mathrm{ared}^{(i)}(h_{\mathrm{tr}}^{(i)})}{\mathrm{pred}^{(i)}(h_{\mathrm{tr}}^{(i)})} \tag{38}$$

where

$$\mathrm{ared}^{(i)}(h_{\mathrm{tr}}^{(i)}) = f(x^{(i)}) - f(x^{(i)} + h_{\mathrm{tr}}^{(i)})$$
$$\mathrm{pred}^{(i)}(h_{\mathrm{tr}}^{(i)}) = q^{(i)}(x^{(i)}) - q^{(i)}(x^{(i)} + h_{\mathrm{tr}}^{(i)}). \tag{39}$$

This compares the *actual* reduction in the objective function's value obtained by taking the proposed trust-region step $h_{\mathrm{tr}}^{(i)}$ with the *predicted* reduction in the function value using the local model $q^{(i)}$. Values of ρ close to 1 indicate that the local model $q^{(i)}$ is performing well near the current iterate $x^{(i)}$, so the trust-region radius $\Delta^{(i)}$ can be increased in the next iteration (to allow longer steps and hence faster convergence), while values close to 0 indicate that the local model is performing poorly, and $\Delta^{(i)}$ should be reduced accordingly (Algorithm 1). The entire trust-region method is summarized as Algorithm 2; here, $0 < \eta_1 < \eta_2 < 1$ and $0 < \gamma_1 < 1 < \gamma_2$ are user-supplied parameters specifying the gain-ratio thresholds and scaling factors used in the update of the trust-region radius.

B. Powell's Dog-Leg

While the canonical trust-region method (Algorithm 2) is intellectually pleasing and enjoys excellent convergence and robustness properties (cf., Section IV-D), it is not commonly used in practice because of the computational cost of finding the exact trust-region step h_{tr} in (34). (Briefly, this entails solving a root-finding problem for the Lagrange multiplier λ^* in (37a) that generally requires multiple factorizations of matrices of the form $B + \lambda I$ [22], a prohibitively expensive operation when appearing as a single step within another iterative algorithm.) Instead, most practical trust-region implementations solve (34) approximately using a computationally cheap approximation that is nevertheless accurate enough to preserve the desirable properties of the canonical algorithm. In this section, we describe *Powell's Dog-Leg* [8], [12], which is one such approximate method that can be used whenever the matrix B in the local quadratic model q in (34) is positive definite.

The dog-leg approximation is derived by regarding the trust-region step computed in (34) as a function $h_{\mathrm{tr}}(\Delta)$ of the trust-region radius and considering the effect of varying Δ. For $B \in \mathbb{R}^{n\times n}$ symmetric and positive definite, the local model q has a unique unconstrained minimizer: the Newton step

$$h_N = -B^{-1}g. \tag{40}$$

If the trust-region radius Δ is greater than the length of this step, then the constraint in (34) is inactive, and therefore

$$h_{\mathrm{tr}}(\Delta) = h_N, \quad \|h_N\| \leq \Delta. \tag{41}$$

Conversely, when the trust-region radius Δ is small, the (second-order) quadratic term in the local model (8) is dominated by the linear term, and therefore, the trust-region step computed in (34) will be well approximated by the maximum step length allowable in the direction of steepest descent:

$$h_{\mathrm{tr}}(\Delta) \approx -\frac{\Delta}{\|g\|}g, \quad \Delta \text{ small}. \tag{42}$$

As Δ increases through the range $[0, \infty)$, the exact trust-region step $h_{\mathrm{tr}}(\Delta)$ traces a smooth path interpolating these cases; Powell's Dog-Leg approximates this path using a piecewise linear path with two segments (Fig. 4). The first segment extends from x to the gradient descent step h_{gd}, which we define as the unconstrained minimizer of the local model q along the steepest descent direction:

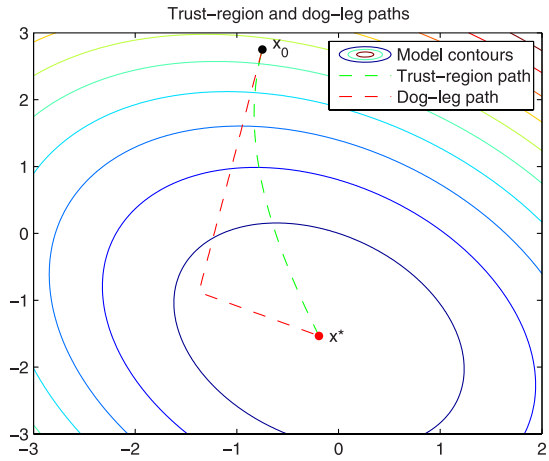


Fig. 4. Trust-region and Powell's Dog-Leg paths for a quadratic model q constructed at x_0 with minimizer x^*. As the trust-region radius Δ increases, the trust-region step $h_{\mathrm{tr}}(\Delta)$ traces a smooth path interpolating the constrained gradient descent step and the Newton step h_N. Powell's Dog-Leg approximates this path using a piecewise linear path with two segments: the first from x_0 to the full gradient descent step h_{gd} and the second from h_{gd} to the Newton step h_N.

$$h_{\mathrm{gd}} = -\alpha g, \qquad \alpha = \operatorname*{argmin}_{a\in\mathbb{R}^+} q(x - ag). \tag{43}$$

Using the definition of the local model (8) and the fact that $B > 0$ by hypothesis, the gradient descent step defined in (43) can be written in closed form as

$$h_{\mathrm{gd}} = -\alpha g, \qquad \alpha = \frac{g^T g}{g^T B g}. \tag{44}$$

The second segment linearly interpolates the gradient descent step h_{gd} and the Newton step h_N. Parameterizing the dog-leg path as $p_{\mathrm{dl}} : [0,1] \to \mathbb{R}^n$, we have

$$p_{\mathrm{dl}}(t) = \begin{cases} 2th_{\mathrm{gd}}, & 0 \leq t \leq \frac{1}{2} \\ h_{\mathrm{gd}} + (2t-1)(h_N - h_{\mathrm{gd}}), & \frac{1}{2} \leq t \leq 1. \end{cases} \tag{45}$$

A direct computation using the definitions (40), (43), and (45) proves the following result (cf., e.g., [8, Sec. 4.1]).

Lemma 1. The dog-leg path p_{dl} defined in (45) satisfies the following properties:

1) $\|p_{\mathrm{dl}}(t)\|$ is monotonically increasing for all $t \in [0,1]$.
2) $q(p_{\mathrm{dl}}(t))$ is monotonically decreasing for all $t \in [0,1]$.

By virtue of Lemma 1, the *Powell's Dog-Leg step* h_{dl} is defined to be the (unique) farthest point along the dog-leg path p_{dl} lying inside the trust-region boundary (cf., Fig. 1):

$$\begin{aligned} h_{\mathrm{dl}} &= p_{\mathrm{dl}}(\tau) \\ \tau &= \max\left\{t \in [0,1] \mid \|p_{\mathrm{dl}}(t)\| \leq \Delta\right\}. \end{aligned} \tag{46}$$

The algorithmic computation of the dog-leg step h_{dl} corresponding to definition (46) is given in Algorithm 3. The scalar β appearing in line 7 is chosen to satisfy

$$\|h_{\mathrm{gd}} + \beta(h_N - h_{\mathrm{gd}})\|^2 = \Delta^2 \tag{47}$$

Algorithm 3 Computing the Dog-Leg Step h_{dl}

procedure COMPUTE_DOG-LEG(h_N, h_{gd}, Δ)
  if $\|h_N\| \leq \Delta$ **then**
    $h_{\mathrm{dl}} \leftarrow h_N$
  else if $\|h_{\mathrm{gd}}\| \geq \Delta$ **then**
    $h_{\mathrm{dl}} \leftarrow \left(\frac{\Delta}{\|h_{\mathrm{gd}}\|}\right) h_{\mathrm{gd}}$
  else
    $h_{\mathrm{dl}} \leftarrow h_{\mathrm{gd}} + \beta\,(h_N - h_{\mathrm{gd}})$, where $\beta \in (0,1)$ is chosen such that $\|h_{\mathrm{dl}}\| = \Delta$ [cf. equation (48)].
  end if
  return h_{dl}
end procedure

which is quadratic in β. By Lemma 1, (47) has exactly one solution in $(0,1)$, which can be written in closed form as

$$v = h_N - h_{\mathrm{gd}}$$
$$\beta = \frac{-h_{\mathrm{gd}}^T v + \sqrt{(h_{\mathrm{gd}}^T v)^2 + (\Delta^2 - \|h_{\mathrm{gd}}\|^2)\|v\|^2}}{\|v\|^2}. \quad (48)$$

The complete Powell's Dog-Leg algorithm is obtained from Algorithm 2 by replacing the computation of the trust-region step h_{tr} in line 5 with the computation of the dog-leg step h_{dl} defined by (40), (44), and Algorithm 3.

C. Indefinite Gauss–Newton–Powell's Dog-Leg

In this section, we derive an approximate trust-region algorithm based on Powell's Dog-Leg with Gauss–Newton steps for solving minimization problems of the form (1).

To begin, we observe that by virtue of the Hessian approximations (13b) used in the Gauss–Newton local model q, $B \geq 0$ always. If $J(x)$ is full-rank, then $B > 0$, and the dog-leg step h_{dl} defined in Section IV-B exists and can be computed in terms of the Gauss–Newton step h_{GN} and the gradient descent step h_{gd}. Equations (18)–(20) already provide a closed-form solution for computing the Gauss–Newton step; therefore, it suffices to provide a closed-form solution for the gradient descent step h_{gd} in (44). Substituting the expressions for B and g from (13) into (44), we find

$$g = 2J(x)^T r(x), \qquad \alpha = \frac{\|g\|^2}{2\|J(x)g\|^2}. \quad (49)$$

Equations (18)–(20) and (49), as well as Algorithm 3, thus enable the computation of the dog-leg step h_{dl} when $B > 0$.

In the case that $J(x)$ is not full-rank, B is not positive definite and the dog-leg step is not defined. However, the *Cauchy step* h_C (the constrained minimizer of the local model q along the steepest descent direction):

$$h_C = -\kappa g$$
$$\kappa = \underset{0 \leq a \leq \frac{\Delta}{\|g\|}}{\operatorname{argmin}} q(x - ag) \quad (50)$$

always exists and is unique for $g \neq 0$ and arbitrary symmetric $B \in \mathbb{R}^{n \times n}$ in the local model (8). Indeed, direct computation

Algorithm 4 The Indefinite Gauss–Newton–Powell's Dog-Leg Algorithm

procedure IGN-PDL(r, x_0, Δ_0, η_1, η_2, γ_1, γ_2)
  Initialize $x \leftarrow x_0$, $\Delta \leftarrow \Delta_0$.
  repeat
    Set $g \leftarrow J(x)^T r(x)$.
    Compute R factor and right-hand side vector d as in equations (18) and (19).
    if R is nonsingular **then**
      Compute Gauss-Newton step h_{GN} using (20).
      Set $\alpha \leftarrow \|g\|^2 / \|J(x)g\|^2$.
      Set $h_{\mathrm{gd}} \leftarrow -\alpha g$.
      Set $h \leftarrow$COMPUTE_DOG-LEG(h_{GN}, h_{gd}, Δ).
    else
      Compute κ using equation (53c).
      Set $h \leftarrow -\kappa g$.
    end if
    Set $x_{\mathrm{proposed}} \leftarrow (x + h)$.
    Compute ρ using (38).
    if $\rho \geq \eta_1$ **then**
      Set $x \leftarrow x_{\mathrm{proposed}}$.
    end if
    $\Delta \leftarrow$ UPDATE_DELTA(ρ, Δ, η_1, η_2, γ_1, γ_2).
  until (stopping criteria)
  return x
end procedure

in (50) shows that the step length κ can be written in a simple closed form as

$$\kappa = \begin{cases} \min\left\{\frac{\Delta}{\|g\|}, \frac{\|g\|^2}{g^T B g}\right\}, & g^T B g > 0 \\ \frac{\Delta}{\|g\|}, & g^T B g \leq 0. \end{cases} \quad (51)$$

Substituting the approximate Hessian (13b) from the Gauss–Newton local model into (51) produces

$$\kappa = \begin{cases} \min\left\{\frac{\Delta}{\|g\|}, \frac{\|g\|^2}{2\|J(x)g\|^2}\right\}, & \|J(x)g\| > 0 \\ \frac{\Delta}{\|g\|}, & \|J(x)g\| = 0. \end{cases} \quad (52)$$

Our proposed algorithm, which we refer to as the *IGN-PDL* algorithm, is obtained from Algorithm 2 by using Gauss–Newton–Powell's Dog-Leg steps in line 5 when $J(x)$ is full-rank, and constrained Cauchy steps otherwise.

Finally, for the sake of notational simplicity and to avoid unnecessary work, we simplify the computations in (49) and (52) by canceling the common factors of 2 in the gradient vector g and the step sizes α and κ. This produces

$$g = J(x)^T r(x) \quad (53a)$$
$$\alpha = \frac{\|g\|^2}{\|J(x)g\|^2} \quad (53b)$$
$$\kappa = \begin{cases} \min\left\{\frac{\Delta}{\|g\|}, \frac{\|g\|^2}{\|J(x)g\|^2}\right\}, & \|J(x)g\| > 0 \\ \frac{\Delta}{\|g\|}, & \|J(x)g\| = 0. \end{cases} \quad (53c)$$

The IGN-PDL algorithm is summarized as Algorithm 4.

D. Theoretical Analysis

While the trust-region approach of Section IV-A may appear to be only a slight variation on the Newton-type methods, in fact trust-region algorithms possess markedly superior robustness properties. In this section, we examine two such properties (global convergence and robustness to numerical ill-conditioning) and prove a strong global convergence result for the IGN-PDL algorithm of Section IV-C.

1) Global Convergence: One of the most attractive and powerful features of trust-region algorithms is that they are globally convergent under very general conditions, even when using (possibly very loose) gradient and Hessian approximations in the local models $q^{(i)}$. By way of illustration, the following remarkable theorem is due to Carter [13].

Theorem 2 (Global convergence of trust-region methods): Let $f : \mathbb{R}^n \to \mathbb{R}$, $x^{(0)} \in \mathbb{R}^n$, and let $\Omega \subseteq \mathbb{R}^n$ be any convex open set containing the sublevel set $\mathcal{L}_f(x^{(0)})$ of f at $x^{(0)}$:

$$\mathcal{L}_f(x^{(0)}) = \left\{ x \in \mathbb{R}^n \mid f(x) \leq f(x^{(0)}) \right\}. \tag{54}$$

Assume further that f is lower-bounded on Ω, $f \in C^1(\Omega)$, and that ∇f is Lipschitz continuous on Ω. Fix constants $0 < \eta_1 < \eta_2 < 1$ and $0 < \gamma_1 < 1 < \gamma_2$, $\zeta < 1 - \eta_2$, $\beta, \sigma \in (0, \infty)$, and $c \in (0, 1]$, and let $\{x^{(i)}\}_{i=0}^{\infty}$ be the sequence of iterates obtained in Algorithm 2 using local models $q^{(i)}$ and approximate trust-region update steps $h_{\mathrm{tr}}^{(i)}$ in line 5 that satisfy the following criteria:

1) *Feasibility:* $\|h_{\mathrm{tr}}^{(i)}\| \leq \Delta^{(i)}$ for all $i \geq 0$.

2) *Gradient approximation error:* The approximate gradients $g^{(i)}$ used in each of the local models $q^{(i)}$ satisfy the bounded relative error criterion:

$$\frac{\|g^{(i)} - \nabla f(x^{(i)})\|}{\|g^{(i)}\|} \leq \zeta. \tag{55}$$

3) *Uniform boundedness of approximate Hessians:* Each $B^{(i)} \in \mathbb{R}^{n \times n}$ is symmetric and satisfies $\|B^{(i)}\| \leq \beta$.

4) *Asymptotic step direction:* Assuming that $\|B^{(i)}\| \leq \beta$ for all $i \geq 0$ and that

$$\liminf_{i \to \infty} \|g^{(i)}\| > 0 \text{ and } \lim_{i \to \infty} \Delta^{(i)} = 0 \tag{56}$$

the step direction $\Theta^{(i)}$ satisfies

$$\lim_{i \to \infty} \cos \Theta^{(i)} = 1 \tag{57}$$

where

$$\cos \Theta^{(i)} = -\frac{\left(g^{(i)}\right)^T h_{\mathrm{tr}}^{(i)}}{\|g^{(i)}\| \|h_{\mathrm{tr}}^{(i)}\|}. \tag{58}$$

5) *Uniform predicted decrease:* Each of the proposed update steps $h_{\mathrm{tr}}^{(i)}$ satisfies

$$\mathrm{pred}^{(i)}(h_{\mathrm{tr}}^{(i)}) \geq \frac{1}{2} c \|g^{(i)}\| \min \left\{ \Delta^{(i)}, \frac{\|g^{(i)}\|}{\sigma} \right\}. \tag{59}$$

Then, either $\nabla f(x^{(i)}) = 0$ for some iterate $x^{(i)}$ or the infinite subsequence $\{x^{(i_k)}\}_{k=0}^{\infty} \subseteq \{x^{(i)}\}_{i=0}^{\infty}$ of iterates that are accepted in line 9 of Algorithm 2 satisfies

$$\lim_{k \to \infty} \|\nabla f(x^{(i_k)})\| = 0. \tag{60}$$

With the aid of Theorem 2, we prove the following convergence result for the IGN-PDL algorithm (Algorithm 4).

Theorem 3 (Global convergence of the IGN-PDL algorithm): Let $r : \mathbb{R}^n \to \mathbb{R}^m$ be C^2, define

$$S(x) = \|r(x)\|^2, \tag{61}$$

and fix $0 < \eta_1 < \eta_2 < 1$ and $0 < \gamma_1 < 1 < \gamma_2$. Given any $x^{(0)} \in \mathbb{R}^n$, if the sublevel set $\mathcal{L}_S(x^{(0)})$ is bounded, then the sequence of iterates $\{x^{(i)}\}$ accepted in line 18 of Algorithm 4 either terminates at some $x^{(k)}$ with $\nabla S(x^{(k)}) = 0$ or satisfies

$$\lim_{i \to \infty} \|\nabla S(x^{(i)})\| \to 0. \tag{62}$$

This theorem is proved in the Appendix.

In practice, the hypotheses of Theorem 3 are quite general; intuitively, the bounded sublevel set condition simply prohibits the cost function S from assigning the same quality to arbitrarily large regions of the state space. Any reasonable inferential cost function arising in practice will satisfy this condition.

2) Robustness to Numerical Ill-Conditioning: In addition to their strong convergence properties, trust-region methods are also naturally robust against numerical ill-conditioning in the linear system (10) used to solve for the Newton step.

Recall that for a matrix $A \in \mathbb{R}^{m \times n}$ with $m \geq n$, the *condition number* $\kappa_2(A)$ (cf., [19, p. 230]) is

$$\kappa_2(A) = \begin{cases} \frac{\sigma_{\max}(A)}{\sigma_{\min}(A)}, & \mathrm{rank}(A) = n \\ \infty, & \mathrm{rank}(A) < n \end{cases} \tag{63}$$

where $\sigma_{\max}(A)$ and $\sigma_{\min}(A)$ give the maximum and minimum singular values of A, respectively. Matrices A with $\kappa_2(A) \gg 1$ are called *ill-conditioned*; these matrices tend to have poor numerical properties when used as the coefficient matrix for inverse problems of the form

$$\min_{x \in \mathbb{R}^n} \|Ax - b\|^2 \tag{64}$$

for $b \in \mathbb{R}^m$ (of which (10) and (15) are special cases). To see why this is so, observe that if $\mathrm{rank}(A) = n$, then (64) has the unique minimizer

$$x_{\mathrm{LS}} = \operatorname*{argmin}_{x \in \mathbb{R}^n} \|Ax - b\|^2 = \sum_{i=1}^{n} \frac{u_i^T b}{\sigma_i} v_i \tag{65}$$

where $\sigma_1 \geq \cdots \geq \sigma_n > 0$ are the singular values of A, and $u_i \in \mathbb{R}^{m \times 1}$ and $v_i \in \mathbb{R}^{n \times 1}$ are the left- and right-singular vectors (respectively) corresponding to σ_i for all $1 \leq i \leq n$ (cf., [19, Sec. 5.5.3]). If $\kappa_2(A) \gg 1$, then since $\{u_i\}_{i=1}^n$ and $\{v_i\}_{i=1}^n$ are orthonormal sets, (65) shows that x_{LS} will tend to be dominated by the effects of those (generally few) components of b lying in subspaces spanned by left-singular vectors u_i corresponding to small singular values σ_i. In practice, this commonly manifests as the least-squares solution x_{LS} "exploding" (as measured by $\|\cdot\|$) through subspaces of $\mathbb{R}^n$ spanned by right-singular vectors v_i corresponding to small singular values σ_i whenever A is ill-conditioned.

One standard approach for addressing poor numerical conditioning in (64) is *Tikhonov regularization* [23]. In this approach, the original problem (64) is replaced by

$$\min_{x\in\mathbb{R}^n} \|Ax-b\|^2 + \lambda\|\Gamma x\|^2 \tag{66}$$

where $\Gamma \in \mathbb{R}^{n\times n}$ is a conditioning matrix designed to control some property of interest of the Tikhonov-regularized solution, and $\lambda \geq 0$ is a parameter that controls the applied degree of regularization. The minimizer x_λ of (66) can be formally computed in closed form as

$$\begin{aligned} x_\lambda &= \operatorname*{argmin}_{x\in\mathbb{R}^n} \|Ax-b\|^2 + \lambda\|\Gamma x\|^2 \\ &= (A^TA + \lambda\Gamma^T\Gamma)^{-1}A^Tb. \end{aligned} \tag{67}$$

The standard form of Tikhonov regularization has $\Gamma = I$, which simply controls the norm of x_λ. More interestingly, this choice also has the effect of improving the numerical conditioning of (66) versus (64). To see this, observe that (66) with $\Gamma = I$ can be expressed as an instance of (64):

$$\min_{x\in\mathbb{R}^n} \|Ax-b\|^2 + \lambda\|x\|^2 = \min_{x\in\mathbb{R}^n} \left\|\bar{A}x-\bar{b}\right\|^2 \tag{68}$$

where

$$\bar{A} = \begin{pmatrix} A \\ \sqrt{\lambda}I \end{pmatrix}, \qquad \bar{b} = \begin{pmatrix} b \\ 0 \end{pmatrix}. \tag{69}$$

The $n \times n$ block $\sqrt{\lambda}I$ of the augmented coefficient matrix $\bar{A}$ ensures that $\sigma_{\min}(\bar{A}) \geq \sqrt{\lambda}$, and the Pythagorean theorem implies that $\sigma_{\max}(\bar{A}) = \|\bar{A}\| \leq \sqrt{\lambda + \|A\|^2}$. These two inequalities together imply that

$$\kappa_2(\bar{A}) \leq \frac{\sqrt{\lambda+\|A\|^2}}{\sqrt{\lambda}} = \sqrt{1+\frac{\|A\|^2}{\lambda}}. \tag{70}$$

Equation (70) shows that the regularization parameter λ controls the effective conditioning of the Tikhonov-regularized system (66)–(69) when $\Gamma = I$.

We now highlight a remarkable connection between trust-region methods and Tikhonov regularization. Consider the optimality condition (37a) for the trust-region step in Theorem 1, and suppose that $B \geq 0$. Since B is symmetric, it can be diagonalized by an orthogonal matrix Q:

$$B = Q\mathrm{diag}(\lambda_1, \ldots, \lambda_n)Q^T \tag{71}$$

where $\lambda_1 \geq \cdots \geq \lambda_n \geq 0$ since $B \geq 0$; (71) is thus also a singular value decomposition for B, and $\kappa_2(B) = \lambda_1/\lambda_n$. The same Q appearing in (71) also diagonalizes the matrix $\bar{B} = B + \lambda^* I$ appearing in (37a):

$$\bar{B} = Q\ \mathrm{diag}(\lambda_1+\lambda^*, \ldots, \lambda_n+\lambda^*)Q^T \tag{72}$$

and therefore

$$\kappa_2(\bar{B}) = \frac{\lambda_1+\lambda^*}{\lambda_n+\lambda^*} \leq \frac{\lambda_1}{\lambda_n} = \kappa_2(B) \tag{73}$$

where the inequality in (73) is strict if $\lambda^* > 0$.

Theorem 1 and (71)–(73) show that the trust-region step h_{tr} determined by (34) can be interpreted as a kind of Tikhonov-regularized solution of the system (10) defining the (approximate) Newton step h_N; indeed, in the specific case of the Gauss–Newton system defined by (13), the corresponding first-order optimality condition (37a) is

$$\left(J(x)^TJ(x) + \lambda^* I\right) h_{\mathrm{tr}} = -J(x)^T r(x) \tag{74}$$

which is an instance of the Tikhonov-regularized system (67). In particular, (73) shows that h_{tr} is the solution of a system whose conditioning is always *at least* as good as that of the system (10) defining h_N.

This analysis shows that trust-region methods are *innately robust* to ill-conditioning, and we therefore expect them to be particularly effective (as compared to pure Newton-type methods) whenever the approximate Hessians $B^{(i)}$ used in the local quadratic models (8) are ill-conditioned.

V. Robust Incremental least-Squares Estimation: Incrementalizing Powell's Dog-Leg

In this section, we present *RISE*, an incrementalized version of the IGN-PDL algorithm (Algorithm 4). For pedagogical clarity, we begin by following the original derivation of RISE as given in [24], in which IGN-PDL is incrementalized with the aid of iSAM. We then derive RISE2, a new and improved version of the RISE algorithm obtained by using iSAM2 [10] in place of the original iSAM.

A. Derivation of Robust Incremental least-Squares Estimation

The IGN-PDL algorithm computes the approximate trust-region update step h from the Gauss–Newton step h_{GN} and the gradient descent step h_{gd} when R is nonsingular, and the Cauchy step h_C when R is singular (cf., lines 7–10 and 13 of Algorithm 4). As iSAM already implements an efficient incremental algorithm for computing h_{GN} in the nonsingular case, it remains only to develop efficient methods to compute h_{gd} and h_C. In turn, lines 9 and 13 of Algorithm 4 show that h_{gd} and h_C are computed in terms of the gradient direction vector g and the scale factors α and κ defined in (53). Thus, it suffices to determine efficient incrementalized versions of (53a)–(53c).

Letting $x = (x_{\mathrm{old}}, x_{\mathrm{new}})$ as before and substituting the block decompositions (21) and (22) into (53a) produces

$$\begin{aligned} \bar{g} &= \bar{J}(\hat{x})^T\bar{r}(\hat{x}) \\ &= \begin{pmatrix} J(\hat{x}_{\mathrm{old}})^T & \\ 0 & J_{\mathrm{new}}(\hat{x})^T \end{pmatrix} \begin{pmatrix} r_{\mathrm{old}}(\hat{x}_{\mathrm{old}}) \\ r_{\mathrm{new}}(\hat{x}) \end{pmatrix} \\ &= \begin{pmatrix} J(\hat{x}_{\mathrm{old}})^T r_{\mathrm{old}}(\hat{x}_{\mathrm{old}}) \\ 0 \end{pmatrix} + J_{\mathrm{new}}(\hat{x})^T r_{\mathrm{new}}(\hat{x}). \end{aligned} \tag{75}$$

Comparing the right-hand side of (75) with (53a), we recognize the product $J(\hat{x}_{\mathrm{old}})^T r_{\mathrm{old}}(\hat{x}_{\mathrm{old}})$ as nothing more than $g = g_{\mathrm{old}}$, the gradient direction vector of the original (i.e., unaugmented) system at the linearization point $\hat{x}_{\mathrm{old}}$. Thus, (75) can be reduced to

$$\bar{g} = \begin{pmatrix} g \\ 0 \end{pmatrix} + J_{\mathrm{new}}(\hat{x})^T r_{\mathrm{new}}(\hat{x}). \tag{76}$$

Since the matrix $J_{\mathrm{new}}(\hat{x})$ is sparse and its row dimension is equal to the (small) number of new measurements added when

the system is extended, (76) provides an efficient method for obtaining the gradient direction vector $\bar{g}$ for the augmented system by *incrementally updating* the previous gradient direction vector g, as desired.

Furthermore, in addition to obtaining $\bar{g}$ from g using (76) in incremental update steps, we can also exploit computations already performed by iSAM to more efficiently batch-compute $\bar{g}$ during relinearization steps, when incremental updates cannot be performed. Substituting (29) into (53a), we obtain

$$\bar{g} = \left(\bar{Q}\begin{pmatrix}\bar{R}\\0\end{pmatrix}\right)^T \bar{r}(\hat{x}) = \begin{pmatrix}\bar{R}^T & 0\end{pmatrix}\bar{Q}^T\bar{r}(\hat{x}). \tag{77}$$

Comparing (77) with (30) then shows that

$$\bar{g} = \begin{pmatrix}\bar{R}^T & 0\end{pmatrix}\begin{pmatrix}\bar{d}\\\bar{e}\end{pmatrix} = \bar{R}^T\bar{d}. \tag{78}$$

The advantage of (78) versus (53a) is that $\bar{R}$ is a sparse matrix of smaller dimension than $\bar{J}(\hat{x})$ so that the matrix-vector multiplication in (78) will be faster. Moreover, since iSAM already computes the factor $\bar{R}$ and the right-hand side vector $\bar{d}$, the factors on the right-hand side of (78) are available at no additional computational expense.

Having shown how to compute the vector $\bar{g}$, it remains only to determine the scaling factors α and κ as in (53b) and (53c). The magnitude of $\bar{g}$ can, of course, be computed efficiently directly from $\bar{g}$ itself, which leaves only the denominator $\|\bar{J}(\hat{x})\bar{g}\|^2$. To compute this quantity, we again exploit the fact that iSAM already maintains the $\bar{R}$ factor of the QR decomposition for $\bar{J}(\hat{x})$; for since $\bar{Q}$ is orthogonal, then

$$\left\|\bar{J}(\hat{x})\bar{g}\right\| = \left\|\left(\bar{Q}\begin{pmatrix}\bar{R}\\0\end{pmatrix}\right)\bar{g}\right\| = \left\|\begin{pmatrix}\bar{R}\\0\end{pmatrix}\bar{g}\right\| = \left\|\bar{R}\bar{g}\right\| \tag{79}$$

and (53b) and (53c) are therefore equivalent to

$$\alpha = \frac{\|\bar{g}\|^2}{\|\bar{R}\bar{g}\|^2} \tag{80a}$$

$$\kappa = \begin{cases}\min\left\{\frac{\Delta}{\|\bar{g}\|}, \frac{\|\bar{g}\|^2}{\|\bar{R}\bar{g}\|^2}\right\}, & \|\bar{R}\bar{g}\| > 0\\ \frac{\Delta}{\|\bar{g}\|}, & \|\bar{R}\bar{g}\| = 0.\end{cases} \tag{80b}$$

Again, since $\bar{R}$ is sparse, the matrix-vector multiplication appearing in (80) is efficient.

Equations (76), (78), and (80) enable the implementation of RISE, a fully incrementalized version of the IGN-PDL algorithm that integrates directly into the existing iSAM framework (Algorithm 5).

Algorithm 5 The RISE Algorithm

1: **procedure** RISE
2: Initialization: $\hat{x}_{\text{old}}, \hat{x}_{\text{estimate}} \leftarrow x_0$, $\Delta \leftarrow \Delta_0$.
3: **while** ($\exists$ new data $(\hat{x}_{\text{new}}, r_{\text{new}})$) **do**
4: **if** (relinearize) **then**
5: Update linearization point: $\hat{x}_{\text{old}} \leftarrow \hat{x}_{\text{estimate}}$.
6: Construct Jacobian $\bar{J}(\hat{x}_{\text{old}}, \hat{x}_{\text{new}})$.
7: Perform complete QR decomposition on $\bar{J}(\hat{x})$, cache $\bar{R}$ factor and right-hand side vector $\bar{d}$ as in equations (18) and (19).
8: Set $\bar{g} \leftarrow \bar{R}^T\bar{d}$.
9: **else**
10: Compute the partial Jacobian $J_{\text{new}}(\hat{x})$ in (24).
11: Obtain and cache the new $\bar{R}$ factor and new right-hand side vector $\bar{d}$ by means of Givens rotations as in equations (27) and (31).
12: Set
$$\bar{g} \leftarrow \begin{pmatrix}g\\0\end{pmatrix} + J_{\text{new}}(\hat{x})^T r_{\text{new}}(\hat{x}).$$
13: **end if**
14: **if** $\bar{R}$ is nonsingular **then**
15: Compute Gauss-Newton step h_{GN} using (32).
16: Set $\alpha \leftarrow \|\bar{g}\|^2/\|\bar{R}\bar{g}\|^2$.
17: Set $h_{\text{gd}} \leftarrow -\alpha\bar{g}$.
18: Set $h \leftarrow$COMPUTE_DOG-LEG(h_{GN}, h_{gd}, Δ).
19: **else**
20: Compute κ using (80b).
21: Set $h \leftarrow -\kappa\bar{g}$.
22: **end if**
23: Set $\hat{x}_{\text{proposed}} \leftarrow (\hat{x} + h)$.
24: Compute ρ using (38).
25: **if** $\rho \geq \eta_1$ **then**
26: Update estimate: $\hat{x}_{\text{estimate}} \leftarrow \hat{x}_{\text{proposed}}$.
27: **else**
28: Retain current estimate: $\hat{x}_{\text{estimate}} \leftarrow \hat{x}$.
29: **end if**
30: Set $\Delta \leftarrow$UPDATE_DELTA(ρ, Δ, η_1, η_2, γ_1, γ_2).
31: Update cached variables: $\hat{x}_{\text{old}} \leftarrow \hat{x}$, $r \leftarrow \bar{r}$, $g \leftarrow \bar{g}$, $R \leftarrow \bar{R}$, $d \leftarrow \bar{d}$.
32: **end while**
33: **return** $\hat{x}_{\text{estimate}}$
34: **end procedure**

B. RISE2: Enabling Fluid Relinearization

In this section, we present RISE2, an improved version of RISE obtained by replacing iSAM with iSAM2 [10] in the derivation given in Section V-A. iSAM2 improves upon iSAM by eliminating the need to periodically reevaluate and refactor the entire Jacobian (two very expensive batch operations) in order to relinearize the system about its corrected estimate (cf., Section III-C); instead, iSAM2 efficiently relinearizes the system at *every* iteration by applying direct updates only to those (few) rows of R and d that are modified when relinearization occurs, a process known as *fluid relinearization*. Similarly, RISE2 eliminates expensive periodic batch factorizations (cf., lines 4–8 in Algorithm 5) by applying iSAM2's fluid relinearization in order to efficiently update R, d, and the gradient direction vector g to their values at the updated estimate $\hat{x}_{\text{estimate}}$ at *every* iteration.

Internally, iSAM2 achieves efficient computation by means of the Bayes Tree [25], which encodes a symbolic factorization of the R factor and right-hand side vector d of the linear system

(20) obtained from the QR decomposition in (18) and (19). Each node in the tree stores pairs of the form $[R_i, d_i]$, where $R_i \in \mathbb{R}^{1\times n}$ is the ith row of the matrix R, and $d_i \in \mathbb{R}$ is the ith element of the right-hand side vector d:

$$R = \begin{pmatrix} R_1 \\ \vdots \\ R_n \end{pmatrix} \in \mathbb{R}^{n\times n}, \qquad d = \begin{pmatrix} d_1 \\ \vdots \\ d_n \end{pmatrix} \in \mathbb{R}^{n\times 1} \quad (81)$$

(cf., [10, Fig. 3(c)]). Updating R and d by adding new measurements to the system or relinearizing the system about its updated estimate $\hat{x}_{\mathrm{estimate}}$ can be implemented as simple computationally efficient editing operations on the tree itself (cf., [10, Algs. 4 and 5], respectively); similarly, solving the updated system for the Gauss–Newton step is achieved by means of a simple single-pass algorithm flowing from the root toward the leaves ([10, Alg. 7]). As in the case of the original RISE algorithm, we obtain RISE2 by exploiting these computations to produce an efficient incremental update rule for the gradient direction vector g.

According to (78), g can be computed from R and d as

$$g = R^T d. \quad (82)$$

Substituting the block decomposition (81) into (82) produces

$$g = R^T d = \begin{pmatrix} R_1^T & \cdots & R_n^T \end{pmatrix} \begin{pmatrix} d_1 \\ \vdots \\ d_n \end{pmatrix} = \sum_{i=1}^{n} d_i R_i^T \quad (83)$$

which expresses g as a linear combination of the rows of R.

When fluid relinearization and updating are applied, some of the values $[R_i, d_i]$ in the nodes of the Bayes Tree may be modified. By means of (83), we can recover the updated value $\bar{g}$ of the gradient direction vector corresponding to the new linearization point $\hat{x}_{\mathrm{estimate}}$ using *only* those quantities that the Bayes Tree already computes during this update. Specifically, we initialize $\bar{g}$ according to

$$\bar{g} \leftarrow \begin{pmatrix} g \\ 0 \end{pmatrix} \quad (84)$$

and for each pair $[R_i, d_i]$ that was modified during the update of the Bayes Tree, we likewise update $\bar{g}$ according to

$$\bar{g} \leftarrow \bar{g} - \begin{pmatrix} g_i \\ 0 \end{pmatrix} + \bar{g}_i \quad (85)$$

where

$$g_i = d_i R_i^T, \qquad \bar{g}_i = \bar{d}_i \bar{R}_i^T. \quad (86)$$

Here, $[R_i, d_i]$ gives the values in row i *prior* to their update, and $[\bar{R}_i, \bar{d}_i]$ gives these values *after* their update. Equations (85) and (86) indicate that the update to $\bar{g}$ due to row i's update simply consists of subtracting off the contribution g_i to the gradient direction vector coming from row i *prior* to its update and replacing it with the *updated* contribution $\bar{g}_i$.

Replacing the computation of g in the original RISE algorithm (lines 4–13 of Algorithm 5) with the new incremental update procedure (84)–(86) produces RISE2 (Algorithm 6).

Algorithm 6 The RISE2 algorithm

1: **procedure** RISE2
2: Initialization: $\hat{x}_{\mathrm{estimate}} \leftarrow x_0$, $\Delta \leftarrow \Delta_0$.
3: **while** ($\exists$ new data $(\hat{x}_{\mathrm{new}}, r_{\mathrm{new}})$) **do**
4: Add new nodes for r_{new} to Bayes Tree and set initial augmented estimate: $\hat{x} = (\hat{x}_{\mathrm{estimate}}, \hat{x}_{\mathrm{new}})$.
5: Apply Algorithm 6 of [10] to update the Bayes Tree, R and d, and the linearization point $\hat{x}_{\mathrm{lin}}$.
6: Initialize gradient $\bar{g}$ for augmented system:
$$\bar{g} \leftarrow \begin{pmatrix} g \\ 0 \end{pmatrix}.$$
7: **for all** (pairs $[R_i, d_i]$ modified in Step 5) **do**
8: Compute the updated contribution $\bar{g}_i$ to the gradient direction vector $\bar{g}$ coming from row i:
$$\bar{g}_i \leftarrow \bar{d}_i \bar{R}_i^T.$$
9: Update gradient direction vector:
$$\bar{g} \leftarrow \bar{g} - \begin{pmatrix} g_i \\ 0 \end{pmatrix} + \bar{g}_i.$$
10: Cache updated value of g_i: $g_i \leftarrow \bar{g}_i$.
11: **end for**
12: Cache updated value of gradient: $g \leftarrow \bar{g}$.
13: **if** R is nonsingular **then**
14: Compute the Gauss-Newton step h_{GN} using Algorithm 7 of [10].
15: Set $\alpha \leftarrow \|g\|^2 / \|Rg\|^2$.
16: Set $h_{\mathrm{gd}} \leftarrow -\alpha g$.
17: Set $h \leftarrow$ COMPUTE_DOG-LEG(h_{GN}, h_{gd}, Δ).
18: **else**
19: Compute κ using (80b).
20: Set $h \leftarrow -\kappa g$.
21: **end if**
22: Set $\hat{x}_{\mathrm{proposed}} \leftarrow (\hat{x}_{\mathrm{lin}} + h)$.
23: Compute ρ using (38).
24: **if** $\rho \geq \eta_1$ **then**
25: Update estimate: $\hat{x}_{\mathrm{estimate}} \leftarrow \hat{x}_{\mathrm{proposed}}$.
26: **else**
27: Retain current estimate: $\hat{x}_{\mathrm{estimate}} \leftarrow \hat{x}_{\mathrm{lin}}$.
28: **end if**
29: $\Delta \leftarrow$ UPDATE_DELTA(ρ, Δ, η_1, η_2, γ_1, γ_2).
30: **end while**
31: **return** $\hat{x}_{\mathrm{estimate}}$
32: **end procedure**

We point out that in both cases, RISE(2)'s efficiency and incrementality are a direct result of exploiting iSAM(2)'s preexisting functionality for incrementally updating R and d. In addition to being a purely intellectually pleasing result, this also means that any other computations depending upon preexisting iSAM(2) functionality (for example, online covariance extraction for data association [26]) can proceed with RISE(2) *without modification*.

VI. Related Work

There is a vast body of prior work on least-squares problems (1) and their solution; indeed, entire books have been devoted to this subject alone [27]. However, since our own interest in this problem is motivated by visual mapping applications, we will restrict our attention to prior work in the SLAM and BA literature that attempts to solve (1) exactly.

In the SLAM literature, the first formulation of the *full* or *smoothing* problem as an instance of (1) is due to Lu and Milios [28], who proposed to solve it using the Gauss–Newton algorithm. This approach remains perhaps the most popular, with many well-known algorithms [10], [29]–[33] differing only in how they solve the linear system (10) corresponding to (13). Lu and Milios themselves originally proposed to solve (10) directly using dense matrix inversion, but the $O(n^3)$ computational cost of this technique is only tractable for fairly small problems. Subsequent work achieved improved computational speeds by exploiting symmetric positive definiteness and sparsity in the approximate Hessian (13b) using iterative methods such as sparse preconditioned conjugate gradient [29] or (multiresolution) Gauss–Seidel relaxation [30]. Thrun and Montemerlo [31] exploited the sparse block structure of the Hessian in a direct method by using the Schur complement to solve first the (dense but low-dimensional) system corresponding to the robot pose update and then the (large but sparse) system corresponding to the landmark update. Grisetti *et al.* [32] exploited sparsity by implementing a hierarchical version of Gauss–Newton on a multiresolution pyramid and using lazy evaluation when propagating updates downward.

To further exploit sparsity, Dellaert and Kaess [34] conducted a thorough investigation of direct linear-algebraic techniques for efficiently solving sparse linear systems of the form (10). One surprising result of this analysis was the primacy of variable ordering strategies as a factor in the computational cost of solving these systems; indeed, the use of fast sparse matrix factorization algorithms (such as sparse multifrontal QR [35] or Cholesky [36] decompositions) coupled with good variable ordering heuristics [21] enables the linear system (10) to be *solved* in a fraction of the time needed to simply *construct* it (i.e., evaluate the Jacobian or Hessian). This insight, in turn, eventually led to the development of the Bayes Tree [25] and iSAM2 [10], which directly *update* the reduced linear system (20) at each iteration rather than reevaluating and refactoring the Hessian in (10); this completely incremental approach is the current state of the art among Gauss–Newton-based optimization algorithms in robotics.

Alternatively, some SLAM approaches propose to solve (1) using first-order methods. These methods have excellent robustness properties, but are limited to a linear convergence rate since they do not take advantage of curvature information. Olson *et al.* [37] proposed to overcome this slow convergence by using a deterministic variation of Jacobi-preconditioned stochastic gradient descent with a clever parameterization that enables the algorithm to make more rapid progress through the state space. While originally conceived as a batch algorithm, Olson's method was later adapted to the online setting through the use of spatially varying learning rates and lazy evaluation [38]. Grisetti *et al.* [39] improved upon Olson's original parameterization by ordering variables using a spanning tree through the network of constraints rather than temporally. These approaches were later unified to produce TORO [40].

In BA, the optimization (1) is typically solved in the batch (offline) case, and therefore BA solutions generally emphasize good robustness and convergence properties rather than real-time computational speed. To that end, many popular software packages used for BA (e.g., g^{2}o [33], SBA [41], and sSBA [42]) implement sparse batch versions of Levenberg–Marquardt to take advantage of its desirable robustness and numerical properties. One notable online approach is the *continuable LM* method of [43], which (like iSAM and RISE) applies a single update step per iteration and (also like RISE) maintains the value of the damping parameter λ across iterations; however, unlike RISE [which incrementally *updates* the linear system (10)], it completely reconstructs and refactors the modified normal equations (74) in each iteration, which are the most expensive parts of computing each update step.

In summary, while prior work has addressed subsets of the following criteria, we believe that the RISE algorithm is the first to satisfy all three simultaneously.

1) *Speed:* We implement a numerical optimization method with a superlinear convergence rate and incrementally update the linear system (32) across iterations rather than recomputing it, thus achieving computational speeds comparable to state-of-the-art online incremental sparse least-squares solvers.
2) *Robustness:* Unlike purely Gauss–Newton-based methods, RISE is provably robust to highly nonlinear systems and numerically ill-conditioned Jacobians and is globally convergent for a broad class of objective functions.
3) *Generality:* We place no restrictions on the admissible parameterizations of the state variables, nor on the number of arguments to each of the functions r_i.

VII. Experimental Results

In this section, we illustrate the performance of the Powell's Dog-Leg, Gauss–Newton, and Levenberg–Marquardt batch methods and the iSAM(2) and RISE(2) incremental methods on a few representative optimization problems arising in the context of robotic mapping. We use the implementations of these algorithms (including preliminary research implementations of RISE and RISE2) available in the iSAM[1] and GTSAM[2] software libraries. Within each experiment, all algorithms are implemented atop a common set of data structures; therefore, any differences in their performance are due solely to differences among the algorithms themselves.

Since our aim in these experiments is to characterize the performance of optimization algorithms, our primary experimental metrics will be the objective function value(s) obtained by each

[1]The iSAM Library (version 1.6), available at http://people.csail.mit.edu/kaess/isam/isam_v1_6.tgz.

[2]The GTSAM Library (version 2.1.0), available at https://research.cc.gatech.edu/borg/sites/edu.borg/files/downloads/gtsam-2.1.0.tgz.

method and their total elapsed computation times. To that end, we have selected test problem instances for which the objective function is correctly specified (i.e., for which the data association is known and correct, either by construction in simulation or through the use of visual fiducials in real-world experiments) in order to experimentally control for the effects of front-end feature extraction and data association.

Finally, one of our principal motivations in developing an incremental optimization method that is more resilient to nonlinearity is to enable the use of robust cost functions in SLAM and visual mapping applications in order to attenuate the ill effects of occasional gross measurement errors. To that end, in the following, we formulate M-estimation using a robust cost function $C(\delta) \geq 0$ as an instance of least-squares minimization by defining, for each raw residual $r_i(x)$, the corresponding robust least-squares residual $r'_i(x) = \pm\sqrt{C(r_i(x))}$; the M-estimate is then obtained as the minimizer of the least-squares problem (1) in the residuals $r'_i(x)$.

A. Simulated Data: sphere2500

Here, we consider the performance of the Powell's Dog-Leg, Gauss–Newton, and Levenberg–Marquardt batch methods and the iSAM and RISE incremental methods on 1000 randomly generated instances[3] of the sphere2500 dataset [10], a standard 6-DOF pose-graph SLAM benchmark. In these experiments, we use the implementations of these algorithms available in the iSAM v1.6 library with their default settings, and apply the pseudo-Huber robust cost function (cf., [2, Sec. A6.8]) with parameter $b = 0.5$.

1) Batch Methods: In this experiment, we compare the performance of the three batch methods to validate our adoption of Powell's Dog-Leg in Section IV as the sparse least-squares optimization method of choice. All algorithms use the same stopping criteria and (in the interest of time) are limited to a maximum of 500 iterations. The initial estimate of the robot path is obtained by integrating the simulated raw odometry measurements. Results from the experiment are summarized in Table I, and the solutions computed by each algorithm for a single representative problem instance are shown in Fig. 5.

As expected, Powell's Dog-Leg and Levenberg–Marquardt obtain solutions of comparable quality, significantly outperforming Gauss–Newton. The superior performance of these algorithms can be understood in terms of their robust convergence properties and high tolerance for nonlinearity; it is clear from Table I that Gauss–Newton makes only marginal progress toward the minima in these examples.

Furthermore, in addition to its favorable accuracy, Powell's Dog-Leg is also the fastest algorithm of the three by an order of magnitude, both in terms of the number of iterations necessary to converge to a local minimum *and* the total elapsed computation time. In this case, the superior speed of Powell's Dog-Leg versus Gauss–Newton is a consequence of its improved convergence; its superior speed versus Levenberg–Marquardt has been studied previously [16] and is due in part to the fact that Powell's Dog-Leg need only solve (10) for the Newton step once at each linearization point, whereas Levenberg–Marquardt must solve the modified normal equations (74) (an expensive operation) whenever the linearization point is updated *or* the damping parameter λ is changed.

2) Incremental Methods: Next, we compare the original iSAM algorithm with RISE (Algorithm 5); results are summarized in Table II (note that the statistics given for each method in the first and second rows of Table II are computed using only the set of problem instances for which that method ran to completion, as explained below).

As expected, RISE significantly outperformed iSAM in terms of final solution quality. In over half of the problem instances, the solution computed by iSAM diverged so far from the true minimum that the numerically computed Jacobian became rank-deficient, forcing the algorithm to abort the computation (solving equation (20) for the Gauss–Newton step requires that the Jacobian be full-rank). Even for those problem instances in which iSAM ran to completion (which are necessarily the instances that are the "easiest" to solve), Table II shows that the solutions computed using the incremental Gauss–Newton approach have significantly greater costs than those computed using the incremental Powell's Dog-Leg method. Indeed, RISE's performance on *all* of the problem instances was, on the average, significantly better than iSAM's performance on only the *easiest* instances.

RISE's enhanced robustness versus iSAM does come at a slightly greater computational cost: Each iteration of the RISE algorithm must compute the Gauss–Newton step (the *output* of iSAM) as an intermediate result in the computation of the dog-leg step. As shown in Algorithms 3 and 5, the cost of computing the dog-leg step given the Gauss–Newton step is dominated by the costs of the matrix-vector multiplications needed to compute the gradient descent step; since these have the same asymptotic time-complexity as the backsubstitution that iSAM already performs to compute the Gauss–Newton step, we expect that RISE will suffer at most a small constant-factor slowdown in speed versus iSAM. The results in Table II show that, in practice, this constant-factor slowdown has only a modest effect on RISE's overall execution speed (an increase of about 20% versus iSAM) when the computational costs of manipulating the underlying data structures are also included: both iSAM and RISE are fast enough to run comfortably in real time.

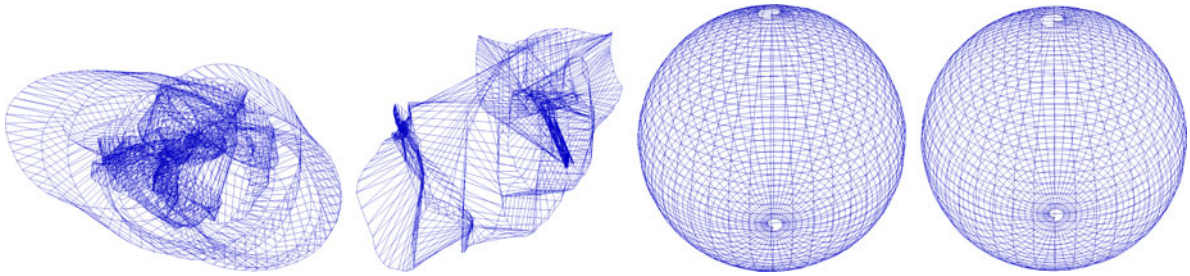
B. Visual Mapping With a Calibrated Monocular Camera

In this experiment, we consider a significantly more challenging test scenario: visual mapping with a calibrated monocular camera via incremental BA. BA is known to suffer from a litany of numerical challenges, including strong nonlinearities in the objective function (due to the nonlinear camera projection mappings, the rotational degrees of freedom in the camera pose estimates, and the use of robust cost functions) and poor numerical conditioning (which can be caused by unfavorable camera configurations or large variations across the uncertainties of the

[3]Generated using the `generateSpheresICRA2012.cpp` executable in the iSAM v1.6 library.

TABLE I
SUMMARY OF RESULTS FOR SPHERE2500 BATCH EXPERIMENTS

	Powell's Dog-Leg			Gauss–Newton			Levenberg–Marquardt		
	Mean	Median	Std. Dev.	Mean	Median	Std. Dev.	Mean	Median	Std. Dev.
Objective function value	8.285 E3	**8.282 E3**	71.40	4.544 E6	3.508 E6	4.443 E6	9.383 E3	8.326 E3	2.650 E3
Computation time (sec)	16.06	**15.73**	1.960	226.2	226.0	2.028	126.7	127.0	43.51
# Iterations	34.48	**34**	4.171	499.9	500	2.500	338.2	328	138.9
# Iteration limit reached		**0**			998			311	



(a) (b) (c) (d)

Fig. 5. Representative instance of the sphere2500 6-DOF SLAM problem from the batch experiments. (a) Initial estimate for the solution (objective function value 1.221 E8). (b) Solution obtained by Gauss–Newton (3.494 E6). (c) Solution obtained by Levenberg–Marquardt (8.306 E3). (d) Solution obtained by Powell's Dog-Leg (8.309 E3). Note that the objective function value for each of these solutions is within $\pm 0.5\%$ of the median value for the corresponding method given in Table I.

TABLE II
SUMMARY OF RESULTS FOR SPHERE2500 INCREMENTAL EXPERIMENTS

	RISE			iSAM		
	Mean	Median	Std. Dev.	Mean	Median	Std. Dev.
Objective function value	9.292 E3	**9.180 E3**	5.840 E2	6.904 E11	1.811 E4	1.242 E13
Computation time (sec)	50.21	50.18	0.13	42.97	**42.95**	0.13
# Rank-deficient Jacobians		**0 (0.0%)**			586 (58.6%)	

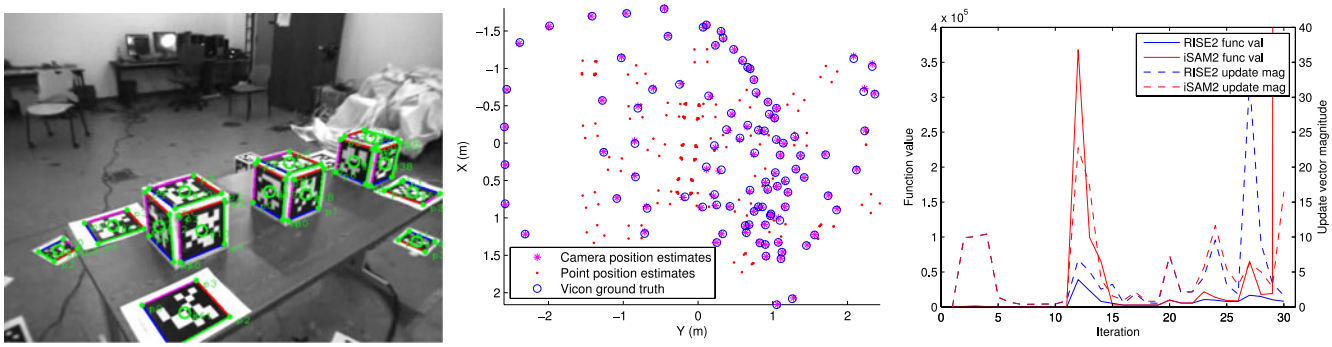
reconstructed camera poses and point positions) [3]. Successful BA optimization methods must be able to robustly address these numerical difficulties.

The input for this experiment consists of a short (141 second) 640×480 monochrome video sequence recorded by the left camera of a Bumblebee2 stereocamera as it was hand-scanned over a room-sized static scene containing 46 AprilTags [44]. A set of keyframes and point observations was extracted from this video by sampling at 1 Hz (to provide a reasonable baseline between successive keyframes) and then selecting as keyframes all images in which at least three AprilTags were detected; this yielded 104 keyframes containing 3876 observations of the 184 AprilTag corner points. These observations were used to estimate the 3-D camera pose of each keyframe and the 3D position of each corner point via BA (i.e., by minimizing the sum of reprojection errors under the Huber robust cost function (cf., [2, Sec. A6.8]) with parameter $b = 1$); the camera's internal calibration was estimated using Zhang's method [45] immediately prior to recording the video and treated as a fixed constant. No prior information about the AprilTags' geometry was included in the adjustment; the tags were used only to solve the data association problem, which is beyond the scope of this paper. Ground truth was also acquired by tracking the position of the camera using a Vicon system.[4]

Incremental BA was performed using the iSAM2 and RISE2 implementations available in the GTSAM v2.1.0 library (here, one iteration comprised the incorporation of all observations from a single keyframe). Camera poses were initialized using the EPnP algorithm [46] (whenever at least four previously initialized points were in view) or the two-view homogeneous DLT method of [2, Sec. 9.6]; points were initialized using either the homogeneous DLT triangulation method of [2, Sec. 12.2] or the two-view homogeneous method of [2, Sec. 9.6]. Results from the experiment are shown in Fig. 6 and Table III.

The RISE2 algorithm successfully processed all 104 frames, converging to a final solution with an objective function value of 558.8 and a raw RMS reprojection error of 0.3797 pixels. In contrast, the iSAM2 algorithm did not even run to completion on this example; in the 31st iteration, it reported that the Jacobian $J(x^{(31)})$ was numerically rank-deficient and aborted the com-

[4]The complete dataset for this experiment (including ground truth) is available at http://groups.csail.mit.edu/marine/apriltags_groundtruth_BA/.



(a) (b) (c)

Fig. 6. Visual mapping with a calibrated monocular camera. (a) Graphical overlay highlighting 14 detected AprilTags and the estimated positions of their corner points in keyframe 96, one of 104 keyframes extracted from the input video. (b) Overhead view of the final camera and point position estimates obtained via incremental BA using RISE2, registered against the Vicon system ground truth. (c) Plot of the objective function values and update vector magnitudes computed by iSAM2 and RISE2 for the first 30 iterations of the visual mapping task (the iSAM2 algorithm aborted the computation in iteration 31 with a numerically rank-deficient Jacobian).

TABLE III
SUMMARY OF RESULTS FOR INCREMENTAL BA

	RISE2 (final)			RISE2 (iteration 30)			iSAM2 (iteration 30)		
	Mean	Median	Std. Dev.	Mean	Median	Std. Dev.	Mean	Median	Std. Dev.
Objective function value		5.588 E2			**8.157 E3**			1.574 E7	
Computation time (sec)		1.040 E2			8.618 E-1			**7.746 E-1**	
Camera position errors (m)	1.574 E-2	1.384 E-2	8.209 E-3	1.844 E-1	6.170 E-2	5.855 E-1	3.164 E-1	4.117 E-2	6.967 E-1

putation. A closer investigation reveals that the Gauss–Newton update step applied by iSAM2 in iteration 30 overshot the true objective function minimum by moving five points *behind* several of the cameras from which they were visible near the newly initialized camera for keyframe 30 (cf., Fig. 6(c) and Table III); this led to a poor initialization of the camera pose for keyframe 31 and the subsequent failure of the algorithm.

Our experience has been that this failure mode (overshooting followed by the rank-deficiency of the Jacobian computed in the next iteration) is generic when iSAM(2) is applied to highly nonlinear or ill-conditioned systems. In contrast, RISE(2)'s use of the trust-region approach enables it to reliably overcome these numerical challenges.

VIII. CONCLUSION

In this paper, we have presented RISE, an incremental trust-region method for robust online sparse least-squares estimation. RISE improves upon current state-of-the-art sequential sparse least-squares solvers by providing superior robustness to objective function nonlinearity and numerical ill-conditioning while simultaneously exploiting prior advances in incremental optimization to achieve fast online computation. In addition to deriving the RISE algorithm itself, we also provided a thorough theoretical analysis of its desirable numerical properties and proved its global convergence for a broad class of inferential cost functions (twice-continuously differentiable functions with bounded sublevel sets). Finally, we evaluated the algorithm empirically in simulation using standard 6-DOF pose-graph SLAM benchmark datasets and demonstrated its superior performance on a more challenging real-world example arising in the context of online visual mapping.

In addition to its utility in SLAM and visual mapping tasks, we believe that the RISE algorithm can also be advantageously applied to other numerically challenging online inference problems sharing the common mathematical structure (2); we are particularly interested in its use as a superlinear optimization method for online machine learning with kernels [5]. Recent work in the robotics [47] and computer vision [48] literature has shown promising results in this direction, but utilizes Gauss–Newton-based approaches that (as demonstrated herein) limit the class of objective functions that can be reliably employed. It would be interesting to consider applications of RISE (and related techniques) in the context of online machine learning in future work.

APPENDIX

Proof of Theorem 3: We prove Theorem 3 by means of Theorem 2. Since $\mathcal{L}_S(x^{(0)})$ is bounded, there exists $D > 0$ such that $\|x - x^{(0)}\| < D$ for all $x \in \mathcal{L}_S(x^{(0)})$. Define the convex open set

$$\Omega = \{x \in \mathbb{R}^n \mid \|x - x^{(0)}\| < D\}. \tag{87}$$

Then, $\overline{\Omega}$ is a closed and bounded (hence compact) subset of $\mathbb{R}^n$ containing the sublevel set $\mathcal{L}_S(x^{(0)})$. Equation (11b) gives the elements of the Hessian matrix $\frac{\partial^2 S}{\partial x^2}$, which are continuous since $r \in C^2$, and therefore $\|\frac{\partial^2 S}{\partial x^2}\|$ attains a maximum value L on the

compact set $\overline{\Omega}$; L is thus a Lipschitz constant for ∇S on Ω. The same argument applied to the approximate Hessians B whose elements are defined by (12) shows that $\|B\|$ likewise attains a maximum β on $\overline{\Omega}$. Finally, S (as a sum-of-squares function) is clearly lower-bounded by 0 everywhere. This establishes the initial hypotheses of Theorem 2.

We now check each of the enumerated criteria 1–5 in turn. Condition 1 is immediate from the definitions of the dog-leg and constrained Cauchy steps in and (46) and (50). Condition 2 is satisfied with $\zeta = 0$ since the gradients used in Algorithm 4 are exact [cf., (13a)]. Condition 3 is satisfied using the value of β obtained in the preceding paragraph. Condition 4 is trivially satisfied in the case of the Cauchy step h_C, since this step is by definition always antiparallel to the gradient [cf., (50)]. In the case of the dog-leg step, we observe using (44) that

$$\|h_{\mathrm{gd}}\| = \left\| -\frac{\|g\|^2}{g^T B g} g \right\| = \frac{\|g\|^3}{g^T B g} \geq \frac{\|g\|^3}{\beta \|g\|^2} = \frac{\|g\|}{\beta} \tag{88}$$

so that (by virtue of the computation in Algorithm 3) h_{dl} is also antiparallel to the gradient whenever

$$\Delta^{(i)} < \frac{\|g^{(i)}\|}{\beta} \leq \|h_{\mathrm{gd}}^{(i)}\|. \tag{89}$$

Equation (89) will hold for all $i > N$ for some sufficiently large N by virtue of the additional hypotheses (56) in effect for the purposes of this condition.

Finally, for condition 5, consider the predicted decrease for the Cauchy step $h_C = -\kappa g$ defined in (50) and (51)

$$\mathrm{pred}(h) = \kappa \|g\|^2 - \frac{1}{2} \kappa^2 g^T B g. \tag{90}$$

The full (i.e., unconstrained) Cauchy step has $\kappa = \frac{\|g\|^2}{g^T B g}$, in which case (90) becomes

$$\begin{aligned} \mathrm{pred}(h) &= \left(\frac{\|g\|^2}{g^T B g} \right) \|g\|^2 - \frac{1}{2} \left(\frac{\|g\|^2}{g^T B g} \right)^2 g^T B g \\ &= \frac{1}{2} \cdot \frac{\|g\|^4}{g^T B g} \geq \frac{\|g\|^2}{2\beta} \end{aligned} \tag{91}$$

since $\|B\| \leq \beta$. In the constrained case with $g^T B g > 0$, $\kappa = \Delta / \|g\|$ only when $\Delta / \|g\| < \|g\|^2 / g^T B g$ (implying $\Delta g^T B g < \|g\|^3$), in which case (90) simplifies as

$$\begin{aligned} \mathrm{pred}(h) &= \left(\frac{\Delta}{\|g\|} \right) \|g\|^2 - \frac{1}{2} \left(\frac{\Delta}{\|g\|} \right)^2 g^T B g \\ &= \Delta \|g\| - \frac{1}{2} \frac{\Delta}{\|g\|^2} \left(\Delta g^T B g \right) \\ &\geq \frac{1}{2} \Delta \|g\|. \end{aligned} \tag{92}$$

Finally, for the constrained case with $g^T B g \leq 0$, we observe that in fact $g^T B g = 0$ since the Gauss–Newton Hessian approximation (13b) is always positive semidefinite; thus, $\kappa = \frac{\Delta}{\|g\|}$ and (90) simplifies as

$$\mathrm{pred}(h) = \left(\frac{\Delta}{\|g\|} \right) \|g\|^2 = \Delta \|g\|. \tag{93}$$

Equations (91)–(93) show that the Cauchy step defined in (50) satisfies the uniform predicted decrease condition (59) with $c = 1$ and $\sigma = \beta$. For the dog-leg step, we simply observe that by virtue of (45), (46), and Lemma 1, $\mathrm{pred}(h_{\mathrm{dl}}) \geq \mathrm{pred}(h_C)$ whenever h_{dl} exists. This proves condition 5 and completes the proof.

ACKNOWLEDGEMENTS

The authors would like to thank F. Dellaert and R. Roberts for the RISE2 implementation in the GTSAM library.

REFERENCES

[1] S. Thrun, W. Burgard, and D. Fox, *Probabilistic Robotics*. Cambridge, MA, USA: MIT Press, 2008.

[2] R. Hartley and A. Zisserman, *Multiple View Geometry in Computer Vision*, 2nd ed. Cambridge, UK: Cambridge Univ. Press, 2004.

[3] B. Triggs, P. McLauchlan, R. Hartley, and A. Fitzgibbon, "Bundle adjustment—A modern synthesis," in *Vision Algorithms: Theory and Practice*, ser. Lecture Notes in Computer Science, vol. 1883, W. Triggs, A. Zisserman, and R. Szeliski, Eds. New York NY, USA: Springer-Verlag, 2000, pp. 298–372.

[4] C. Bishop, *Pattern Recognition and Machine Learning*. New York, NY, USA: Springer Science + Business Media, 2006.

[5] B. Schölkopf and A. Smola, *Learning with Kernels*. Cambridge, MA, USA: MIT Press, 2002.

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

[7] D. Marquardt, "An algorithm for least-squares estimation of nonlinear parameters," *J. Soc. Ind. Appl. Math.*, vol. 11, no. 2, pp. 431–441, Jun. 1963.

[8] J. Nocedal and S. Wright, *Numerical Optimization*, 2nd ed. New York, NY, USA: Springer Science + Business Media, 2006.

[9] M. Kaess, A. Ranganathan, and F. Dellaert, "iSAM: Incremental smoothing and mapping," *IEEE Trans. Robot.*, vol. 24, no. 6, pp. 1365–1378, Dec. 2008.

[10] M. Kaess, H. Johannsson, R. Roberts, V. Ila, J. Leonard, and F. Dellaert, "iSAM2: Incremental smoothing and mapping using the Bayes tree," *Int. J. Robot. Res.*, vol. 31, no. 2, pp. 216–235, Feb. 2012.

[11] R. Fletcher, *Practical Methods of Optimization*, 2nd ed. New York, NY, USA: Wiley, 1987.

[12] M. Powell, "A new algorithm for unconstrained optimization," in *Nonlinear Programming*, J. Rosen, O. Mangasarian, and K. Ritter, Eds. London, U.K.: Academic, 1970, pp. 31–65.

[13] R. Carter, "On the global convergence of trust region algorithms using inexact gradient information," *SIAM J. Numer. Anal.*, vol. 28, no. 1, pp. 251–265, Feb. 1991.

[14] M. Powell, "On the global convergence of trust region algorithms for unconstrained minimization," *Math. Program.*, vol. 29, no. 3, pp. 297–303, 1984.

[15] G. Shultz, R. Schnabel, and R. Byrd, "A family of trust-region-based algorithms for unconstrained minimization with strong global convergence properties," *SIAM J. Numer. Anal.*, vol. 22, no. 1, pp. 47–67, Feb. 1985.

[16] M. Lourakis and A. Argyros, "Is Levenberg-Marquardt the most efficient optimization algorithm for implementing bundle adjustment," *Int. Conf. Comput. Vis.*, vol. 2, pp. 1526–1531, 2005.

[17] F. Kschischang, B. Frey, and H.-A. Loeliger, "Factor graphs and the sum-product algorithm," *IEEE Trans. Inf. Theory*, vol. 47, no. 2, pp. 498–519, Feb. 2001.

[18] D. Rosen, M. Kaess and J. Leonard, "Robust incremental online inference over sparse factor graphs: Beyond the Gaussian case," in *Proc. IEEE Int. Conf. Robot. Autom.*, Karlsruhe, Germany, May 2013, pp. 1017–1024.

[19] G. Golub and C. V. Loan, *Matrix Computations*, 3rd ed. Baltimore, MD, USA: Johns Hopkins Univ. Press, 1996.

[20] S. Strogatz, *Nonlinear Dynamics and Chaos*. Cambridge, MA, USA: Perseus, 2001.

[21] T. Davis, J. Gilbert, S. Larimore, and E. Ng, "A column approximate minimum degree ordering algorithm," *ACM Trans. Math. Softw.*, vol. 30, no. 3, pp. 353–376, Sep. 2004.

[22] J. Moré and D. Sorensen, "Computing a trust region step," *SIAM J. Sci. Statist. Comput.*, vol. 4, no. 3, pp. 55–572, Sep. 1983.

[23] A. Tikhonov and V. Arsenin, *Solutions of Ill-Posed Problems*. Washington, DC, USA: V. H. Winston & Sons, 1977.
[24] D. Rosen, M. Kaess and J. Leonard, "An incremental trust-region method for robust online sparse least-squares estimation," in *Proc. IEEE Int. Conf. Robot. Autom.*, St. Paul, MN, USA, May 2012, pp. 1262–1269.
[25] M. Kaess, V. Ila, R. Roberts, and F. Dellaert, "The Bayes tree: An algorithmic foundation for probabilistic robot mapping," presented at the Int. Workshop Algorithm. Found. Robot., Singapore, Dec. 2010.
[26] M. Kaess and F. Dellaert, "Covariance recovery from a square root information matrix for data association," *J. Robot. Auton. Syst.*, vol. 57, no. 12, pp. 1198–1210, Dec. 2009.
[27] A. Björck, *Numerical Methods for Least Squares Problems*. Philadelphia, PA, USA: SIAM, 1996.
[28] F. Lu and E. Milios, "Globally consistent range scan alignment for environmental mapping," *Auton. Robots*, vol. 4, pp. 333–349, Apr. 1997.
[29] K. Konolige, "Large-scale map-making," in *Proc. Nat. Conf. Artif. Intell.*, 2004, pp. 457–463.
[30] U. Frese, P. Larsson, and T. Duckett, "A multilevel relaxation algorithm for simultaneous localization and mapping," *IEEE Trans. Robot.*, vol. 21, no. 2, pp. 196–207, Apr. 2005.
[31] S. Thrun and M. Montemerlo, "The GraphSLAM algorithm with applications to large-scale mapping of urban structures," *Int. J. Robot. Res.*, vol. 25, no. 5–6, pp. 403–429, 2006.
[32] G. Grisetti, R. Kümmerle, C. Stachniss, U. Frese, and C. Hertzberg, "Hierarchical optimization on manifolds for online 2D and 3D mapping," in *Proc. IEEE Int. Conf. Robot. Autom.*, Anchorage, AK, USA, May 2010, pp. 273–278.
[33] R. Kümmerle, G. Grisetti, H. Strasdat, K. Konolige, and W. Burgard, "g2o: A general framework for graph optimization," in *Proc. IEEE Int. Conf. Robot. Autom.*, Shanghai, China, May 2011, pp. 3607–3613.
[34] F. Dellaert and M. Kaess, "Square Root SAM: Simultaneous localization and mapping via square root information smoothing," *Int. J. Robot. Res.*, vol. 25, no. 12, pp. 1181–1203, Dec. 2006.
[35] P. Matstoms, "Sparse QR factorization in MATLAB," *ACM Trans. Math. Softw.*, vol. 20, no. 1, pp. 136–159, Mar. 1994.
[36] Y. Chen, T. Davis, W. Hager, and S. Rajamanickam, "Algorithm 887: CHOLMOD, supernodal sparse Cholesky factorization and update/downdate," *ACM Trans. Math. Softw.*, vol. 35, no. 3, pp. 22:1–22:14, Oct. 2008.
[37] E. Olson, J. Leonard, and S. Teller, "Fast iterative alignment of pose graphs with poor initial estimates," in *Proc. IEEE Int. Conf. Robot. Autom.*, Orlando, FL, USA, May 2006, pp. 2262–2269.
[38] E. Olson, O. Brock, and C. Stachniss, "Spatially-adaptive learning rates for online incremental SLAM," in *Proc Robot., Sci. Syst.*, Jun. 2007, pp. 73–80.
[39] G. Grisetti, C. Stachniss, S. Grzonka, and W. Burgard, "A tree parameterization for efficiently computing maximum likelihood maps using gradient descent," in *Proc. Robot., Sci. Syst.*, 2007, pp. 65–72.
[40] G. Grisetti, D. Rizzini, C. Stachniss, E. Olson, and W. Burgard, "Online constraint network optimization for efficient maximum likelihood map learning," in *Proc. IEEE Int. Conf. Robot. Autom.*, Pasadena, CA, USA, May 2008, pp. 1880–1885.
[41] M. Lourakis and A. Argyros, "SBA: A software package for generic sparse bundle adjustment," *ACM Trans. Math. Softw.*, vol. 36, no. 1, pp. 1–30, Mar. 2009.
[42] K. Konolige, "Sparse bundle adjustment," in *Proc. Brit. Mach. Vis. Conf.*, 2010, pp. 1–11.
[43] K. Konolige, G. Grisetti, R. Kümmerle, W. Burgard, B. Limketkai, and R. Vincent, "Efficient sparse pose adjustment for 2D mapping," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, Taipei, Taiwan, Oct. 2010, pp. 22–29.
[44] E. Olson, "AprilTag: A robust and flexible visual fiducial system," in *Proc. IEEE Intl. Conf. Robot. Autom.*, Shanghai, China, May. 2011, pp. 3400–3407.
[45] Z. Zhang, "A flexible new technique for camera calibration," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 22, no. 11, pp. 1330–1334, Nov. 2000.
[46] V. Lepetit, F. Moreno-Noguer, and P. Fua, "EPnP: An accurate $O(n)$ solution to the PnP problem," *Int. J. Comput. Vis.*, vol. 81, pp. 155–166, 2009.
[47] C. Tong, P. Furgale, and T. Barfoot, "Gaussian process Gauss-Newton for non-parametric simultaneous localization and mapping," *Int. J. Robot. Res.*, vol. 32, no. 5, pp. 507–525, May 2013.
[48] A. Ranganathan, M.-H. Yang, and J. Ho, "Online sparse Gaussian process regression and its applications," *IEEE Trans. Image Process.*, vol. 20, no. 2, pp. 391–404, Feb. 2011.

David M. Rosen (S'14) received the B.S. degree in mathematics (with a minor in control and dynamical systems) from the California Institute of Technology, Pasadena, CA, USA, in 2008 and the M.A. degree in mathematics from the University of Texas at Austin, Austin, TX, USA, in 2010. He is working toward the Ph.D. degree in computer science with the Computer Science and Artificial Intelligence Laboratory, Massachusetts Institute of Technology, Cambridge, MA, USA.

His research interests include the mathematical foundations of robotic perception, including optimization, probabilistic inference, and machine learning.

Michael Kaess (M'09) received the M.S. and Ph.D. degrees in computer science from the Georgia Institute of Technology, Atlanta, GA, USA, in 2002 and 2008, respectively.

He is currently an Assistant Research Professor with the Robotics Institute, Carnegie Mellon University, Pittsburgh, PA, USA. Previously, he was a Research Scientist and Postdoctoral Associate with the Computer Science and Artificial Intelligence Laboratory, Massachusetts Institute of Technology, Cambridge, MA, USA. His research interests include robot perception for navigation and autonomy.

John J. Leonard (F'14) received the B.S.E.E. degree in electrical engineering and science from the University of Pennsylvania, Philadelphia, PA, USA, in 1987 and the D.Phil. degree in engineering science from the University of Oxford, Oxford, U.K, in 1994.

He is currently a Professor of mechanical and ocean engineering with the Department of Mechanical Engineering and a Member of the Computer Science and Artificial Intelligence Laboratory, Massachusetts Institute of Technology (MIT), Cambridge, MA, USA. His research interests include the problems of navigation and mapping for autonomous mobile robots.

Dr. Leonard has served as an Associate Editor of the IEEE JOURNAL OF OCEANIC ENGINEERING and the IEEE TRANSACTIONS ON ROBOTICS AND AUTOMATION. Currently, he serves as the Area Head for Ocean Science and Engineering within the MIT Department of Mechanical Engineering and as Co-Director of the Ford-MIT Alliance. He received a National Science Foundation Career Award in 1998, an E.T.S. Walton Visitor Award from Science Foundation Ireland in 2004, and the King-Sun Fu Memorial Best Transactions on Robotics Paper Award in 2006.