Powell's Dogleg Method for Nonlinear Least-Squares Minimization

Final Project : 16-811, Fall 2017

Sudharshan Suresh (`sudhars1`), Shubham Agrawal (`sagrawa1`)

November 3, 2018

Abstract

While solving non-linear optimization problems, we operate on linear approximations to the original equation $f(x)$. Least-square problems can be solved via the trust region method, where we consider a region around which the original function can be reasonably approximated using a model. They are robust and work well for ill-conditioned problems. While Levenberg-Marquardt is a popular trust region method, for certain applications Powell's Dogleg can achieve comparable results with significantly faster performance. This project compares the different optimization methods on a representative pose-graph problem, in terms of computation time and residual. This allows us to conclude on the advantages of the dogleg method.

1 Introduction

Non-linear least squares minimization is a fundamental building block of algorithms in robotics, computer vision and learning. In its essence, it approximates the problem with a linear model and iterates the method for refinement. It considers a given m observations and tries to fit an n parameter non-linear model (where $n < m$). Common examples are the Simultaneous Localization and Mapping (SLAM) problem in the robotics community, bundle adjustment (BA) in computer vision and regression in learning. The end-goal of these problems is to obtain states/parameters that maximally explain the measurements when corrupted with noise. For example, Fig. 1 shows us the optimized result of the `victoriapark` experimental dataset[1] (with 6969 poses, 151 landmarks, 3640 landmark measurements and 6968 odometry constraints).

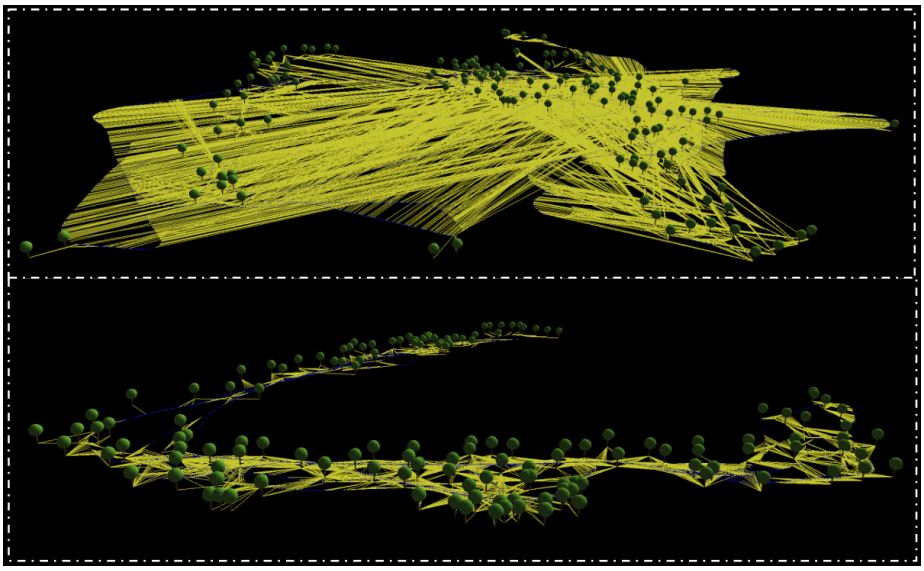


Figure 1: Result of pose-graph optimization on the `victoriapark` dataset

1.1 Prior Work and Tools

The online inference problem is a unique case, as it allows us to exploit sequentiality. A staple method is online gradient descent, which however leads to slow convergence. As an alternative, an incremental method, iSAM [2] (incremental smoothing and mapping) has been developed, particularly for sparse non-linear optimization on the SLAM problem. Other open-source libraries often used by the robotics community include Ceres [3], a C++ library for large optimization problems, and GTSAM [4], which was specifically built towards factor graph approaches. For all optimization results in this paper, we use the iSAM library.

1.2 The Non-linear Least Square Problem

We have a set of m known data points $(a_1, y_1), (a_2, y_2), \ldots, (a_m, y_m)$, and a function $y = f(a, x)$. Here, there are n parameters $x = (x_1, x_2 \ldots x_n)$. We want to find this set of parameters x that best describes this data in the least square sense -

$$S = \sum_{i=1}^{m} R_i^2$$

The function S is given in terms of the residual term $R_i = y_i - f(a_i, x)$. To minimize the function, we equate its derivative to zero -

$$\frac{\partial S}{\partial x_j} = 2 \sum_i R_i \frac{\partial R_i}{\partial x_j} = 0$$

The gradient $\frac{\partial R_i}{\partial x_j}$ does not have a closed solution, which calls upon an iterative technique. As you may already know, Newton's method is a popular such technique, which uses the iterative rule -

$$x^{(n+1)} = x^n - H^{-1}g$$

The gradient g and Hessian H are represented in terms of derivatives as -

$$g_j = 2\sum_{i=1}^{m}\left(R_i\frac{\partial R_i}{\partial x_j}\right)$$

$$H_{jk} = 2\sum_{i=1}^{m}\left(\frac{\partial R_i}{\partial x_j}\frac{\partial R_i}{\partial x_k} + R_i\frac{\partial^2 R_i}{\partial x_j \partial x_k}\right)$$

The second derivative term in the Hessian is required for Newton's method, and is a computationally costly term to compute. We shall see how online optimization methods circumvent this issue.

2 Gauss-Newton

Fundamentally, Gauss-Newton ignores the 2^{nd} derivative term in the Hessian computation above. This follows the assumption that -

1. The function r_i is small in magnitude, especially around the minima.
2. There are only small non-linearities, so that $\frac{\partial^2 r_i}{\partial x_j \partial x_k}$ is small in magnitude.

$$\left|R_i\frac{\partial^2 R_i}{\partial x_j \partial x_k}\right| \ll \left|\frac{\partial R_i}{\partial x_j}\frac{\partial R_i}{\partial x_k}\right|$$

This allows us to make these approximations from Newton's method -

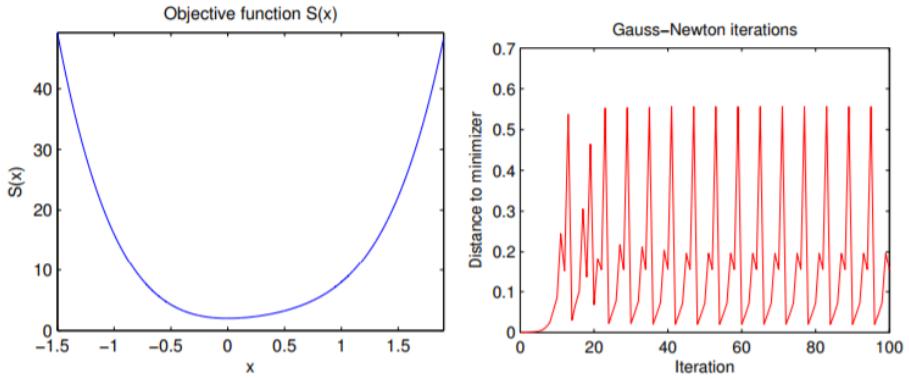
$$g = 2J_r^T r \qquad\qquad H \approx 2J_r^T J_r$$

where J_r comprises of the Jacobian entries $\frac{\partial(r_i)}{\partial x_j} = -J_{ij}$. The update rule for Gauss-Newton is thus -

$$x^{(n+1)} = x^{(n)} + \Delta$$
$$\Delta = -\left(J_r^\intercal J_r\right)^{-1} J_r^\intercal r$$

Gauss-Newton's convergence can approach quadratic, however it is not the preferred algorithm for pose-graph optimization. This is because convergence is not guaranteed globally. The assumptions made on its 2^{nd} partial derivative fails for functions that are not well behaved. It is also possible to construct a well-behaved function with a good initialization that shows global *divergence*. Shown in Fig. 2 is the combination of two residual functions r_1 and r_2 that show this behavior.

$$r_1(x) = x + 1$$
$$r_2(x) = -2x^2 + x - 1.$$



(a) The objective function $S(x)$ (b) Gauss-Newton iterates

Figure 2: Resultant plot, showing the minima at x $\approx$ 0, and Gauss-Newton failing to converge. In this case, even an initiation of $x_0 = 10^{-4}$ fails to assure convergence.

The example shows Gauss-Newton getting caught in a 6-period orbit. Large-scale optimization has many such situations, which is why we don't prefer using Gauss-Newton. This leads us into the discussion on trust-region methods that avoid this problem, which work on functions with significant non-linearities.

3 Trust Region Methods

Trust region methods are a class of iterative optimization algorithms that seek to replace the target optimization function with a simpler one - A linear or quadratic model of the function. Trust region methods have been studied during the last few decades and have given rise to numerical algorithms that are reliable and robust, possessing strong convergence properties and being applicable even to ill-conditioned problems. In a trust region framework, information regarding the objective function is gathered and used to construct a quadratic model function whose behavior in the neighborhood of the current point is similar to that of the Objective function. The model function is trusted to accurately represent it only for points within a hypersphere of radius Δ, centered on the current point, hence the name trust region.

Conceptually, the trust-region approach replaces a n-dimensional unconstrained optimization problem by a n-dimensional constrained one. The replacement pays off because :

1. The subproblem need not be solved to high accuracy, an approximate solution is enough.
2. The model function belongs to a class for which highly effective specialized algorithms have been developed.

3.1 The Trust Region subproblem

We seek to simplify the optimization problem by building a model $f(x_k + s)$ which could be either[6] :

- Linear Model :
$$m_k^L(s) = f_k + s^T g_k$$
- Quadratic model:
$$m_k^Q(s) = f_k + s^T g_k + \frac{1}{2} s^T B_k s$$
Where B_k is chosen to be symmetric

Challenges

- Models may not resemble $f(x_k + s)$ if s is large
- Quadratic models m^Q may be unbounded from below if B_k is indefinite, and possibly if B_k is only positive semi-definite.

Hence, we impose a "trust-region" constraint,

$$||s|| \leq \Delta_k$$

for some scalar radius $\Delta_k > 0$.

Therefore the Trust region subproblem is the constrained optimization problem -

$$\min_{s \in \Re^n} m_k(s) \;\; s.t. \; ||s|| \leq \Delta_k$$

3.2 Variants of Trust region methods

While the basic intuition remains the same, TRMs can vary over the following possible choice when designing the algorithm:

1. Different choices of trust region R_k. For example, using hyperspheres defined by other norms such as $||.\|_1$ or $||.\|_\infty$
2. Choosing the model function m_k. When a Quadratic 2^{nd} order Taylor expansion is chosen, there is a choice in how to approximate the Hessian.
3. Approximate calculation of :

$$x_{k+1} \approx arg \min_{x \in \Re_k} m_k(x)$$

4 State-of-the-art (Levenberg-Marquardt)

4.1 Theory

The Levenberg-Marquardt algorithm [5] is an iterative trust region method, very commonly used in problems involving non linear least squares minimization. It has become a standard technique for nonlinear least-squares problems, widely adopted in various disciplines for dealing with data-fitting applications. The method elegantly combines the Gauss-Newton and gradient descent methods into a single equation by means of a "damping" parameter λ as follows :

$$\left(J^T J + \lambda \, diag\left(J^T J\right)\right) \Delta = J^T b$$

When the current solution is far from a local minimum, the value of λ is increased, and hence the algorithm behaves like a steepest descent method: slow, but guaranteed to converge. When the current solution is close to a local minimum, λ become close to 0, and the method becomes a Gauss-Newton method and exhibits fast convergence.

Choice of Damping Parameter

Various (more or less heuristic) arguments have been put forward for the best choice for the damping parameter λ. Theoretical arguments exist showing why some of these choices guarantee local convergence of the algorithm; however, these choices can make the global convergence of the algorithm suffer from the undesirable properties of steepest descent, in particular very slow convergence close to the optimum.

The absolute values of any choice depends on how well-scaled the initial problem is. Marquardt recommended starting with a value λ_0 and a factor $k > 1$. Initially setting $\lambda = \lambda_0$ and computing the residual sum of squares $S(X)$ after one step from the starting point with the damping factor of $\lambda = \lambda_0$ and secondly with λ_0/k. If both of these are worse than the initial point, then the damping is increased by successive multiplication by k until a better point is found.

If use of the damping factor λ/k results in a reduction in squared residual, then this is taken as the new value of λ (and the new optimum location is taken as

that obtained with this damping factor) and the process continues; if using λ/k resulted in a worse residual, but using λ resulted in a better residual, then λ is left unchanged and the new optimum is taken as the value obtained with λ as damping factor.

4.2 Pseudocode

At each iteration, the augmented equation is solved to calculate a step Δ. The cost function is then evaluated at the new point $x_k+\Delta$ and if there is a reduction, the update is accepted, and the value of λ is decreased to make the method more like Gauss-Newton. However if the cost function S is found not to be decreasing, the update gets rejected, and λ is increased, as the method now reduces its "trust" of the approximation, and becomes more like gradient descent. The augmented equation is then re-solved with this new value of λ.

Algorithm 1 Levenberg-Marquardt algorithm

procedure LM(S, X_0) ▷ Quadratic cost function S
  $\lambda \leftarrow k$
  $t = 0$
  repeat
    J,b $\leftarrow$ linearize $S(X)$ at X^t
    $\Delta \leftarrow$ solve $J^T J + \lambda\, diag\left(J^T J\right) \Delta = J^T b$
    if $S(X^t + \Delta) < S(X^t)$ **then** ▷ Accept update
      $X^{t+1} = X^t + \Delta$
      $\lambda \leftarrow \lambda / k$
    else ▷ Reject update
      $X^{t+1} = X^t$
      $\lambda \leftarrow \lambda * k$
    $t \leftarrow t + 1$
  until convergence
  return X^t ▷ Return latest estimate

4.3 Drawbacks of the LM algorithm

1. When a LM step fails, the LM algorithm requires that the augmented equations resulting from an increased damping term are solved again. In other words, every update to the damping term calls for a new solution of the augmented equations, thus failed steps entail unproductive effort.

2. If J_k is ill-conditioned with condition number ρ then $J_k^T J_k$ is much more illconditioned with condition number ρ^2

3. When the LM iteration approaches convergence, λ tends to zero and the coefficient matrix in thus becomes potentially ill-conditioned. [7]

4. Also, if J_k is a sparse matrix,the matrix $J_k^T J_k + \lambda I$ will usually be dense thus precluding the use of sparse matrix methods for solving the augmented equation

5 Powell's Dogleg[8]

Similarly to the LM algorithm, the DL algorithm for unconstrained minimization tries combinations of the Gauss-Newton and steepest descent directions. In the case of DL, however, this is explicitly controlled via the use of a trust region. Trust region methods have been studied during the last few decades and have given rise to numerical algorithms that are reliable and robust, possessing strong convergence properties and being applicable even to ill-conditioned problem.

5.1 Theory

PDL seeks to calculate the GN and gradient descent steps seperately and then combine them, subject to the step being constrained to an explicit trust region, which is usually a hypersphere of radius Δ.

The Gauss Newton step may be calculated with the familiar expression :

$$\Delta x^{\text{Gauss-Newton}} = \arg\min_{\Delta x} \frac{1}{2}\|J(x)\Delta x + f(x)\|^2$$

While the Gradient descent step may be calculated as the steeped descent minimizer, the cauchy point, denoted as Δx^{Cauchy} :

$$\Delta x^{\text{Cauchy}} = -\frac{\|g(x)\|^2}{\|J(x)g(x)\|^2} g(x)$$

The Trust region is modulated by computing a "Gain ratio" at each iteration. This ratio indicates how accurately the approximation models the actual function, by calculating the ration between the actual change in cost to the predicted change in cost, i.e :

$$GainRatio = \frac{Actual\ Reduction\ in\ objective\ function}{Predicted\ change\ in\ Approximation\ model}$$

This can be calculated as :

$$\rho = \frac{S\left(X^T\right) - S\left(X^T + \Delta\right)}{L\left(0\right) - L\left(\Delta\right)}$$

Where S is the objective function, L is the approximation model and ρ is the Gain ratio. The Gain Ratio is used to change the radius of the hyper-sphere of the trust-region. The conditions are mostly heuristic, but a typical implementation would be:

$$\begin{cases} \rho < 0.25 & \Delta = \frac{\Delta}{2} \\ \rho > 0.75 & \Delta = 2\Delta \\ else & \Delta = \Delta \end{cases}$$

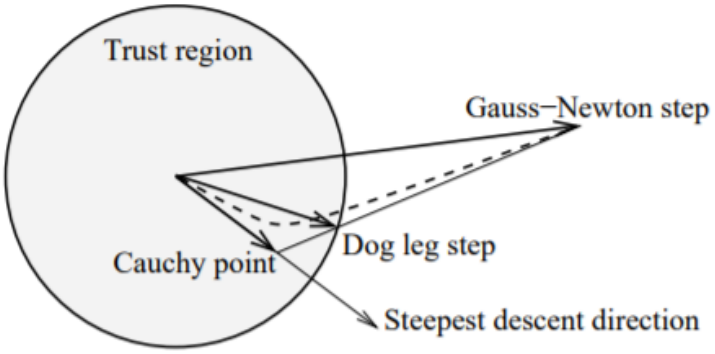


Figure 3: Illustration of the steps in dogleg method.

5.2 Pseudocode

With the seperate computation of the Gauss-Newton step h_N and Gradient descent step h_{gd}, PDL can then combine them, subject to the trust region constraint, to take a step of the optimal size. For the general case where the Cauchy point lies inside the trust region and the Gauss newton step lies outside it, PDL will first move to the cauchy point and then bend towards the GN point until it reaches the trust region boundary. This "bend" in the path taken leads to the "Dogleg" in the method's name.

Algorithm 2 Computing the dog-leg step h_{dl}

procedure COMPUTE_DOG_LEG(h_N, h_{gd}, Δ)
  if $||h_N|| \leq \Delta$ **then** ▷ GN step is within Trust region
    $h_{dl} \leftarrow h_N$
  else if $||h_{gd} \geq \Delta||$ **then** ▷ Updates outside trust region, truncate
    $h_{dl} \leftarrow (\Delta / ||h_{gd}||)\, h_{gd}$
  else ▷ Combine Cauchy and GN step
    $h_{dl} \leftarrow h_{gd} + \beta\,(h_N - h_{gd})$ ▷ β such that $||h_{dl}|| = \Delta$
  return h_{dl}

6 Experiment and Results

Performance is evaluated similar to [9], on a 6DOF pose-graph that we randomly generate - sphere2500. This is constructed by running `generateSpheresICRA2012.cpp` found in the iSAM library. The pose-graph consists of 2500 poses, 4949 constraints (2499 odometry and 2450 loop closings). The optimization is performed using the iSAM v 1.6 library, using a pseudo-Huber robust cost function and a maximum of 500 iterations. This cost function is one often used in SLAM, which is robust to outliers.

The initialization of the sphere is one corrupted with Gaussian noise, and the algorithm attempts to optimize for the original sphere. We test this for all three methods, while also varying the level of noise added. The results can be viewed in Fig. 4 and 5. The adjoining table 1 and 2 display the runtime details.

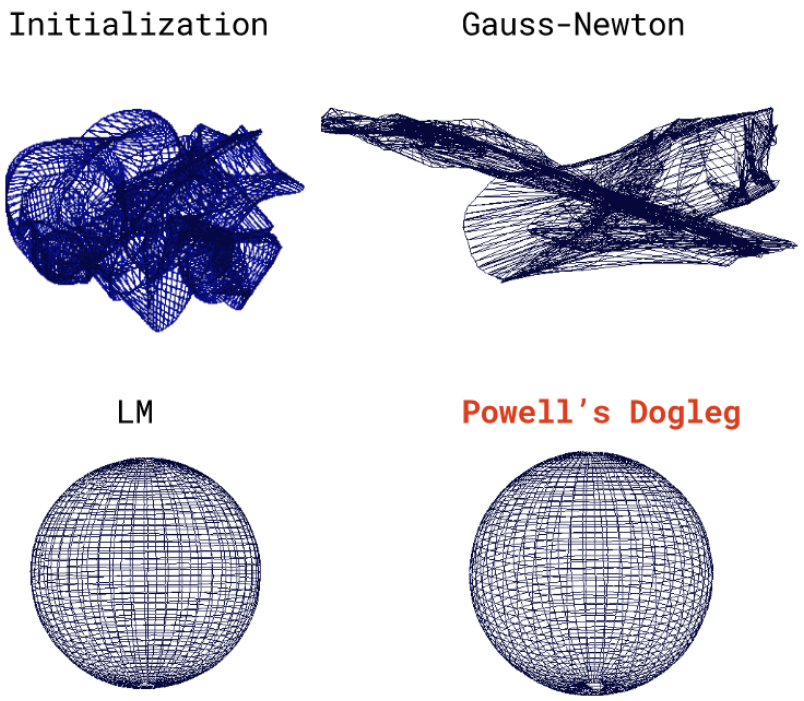


Figure 4: Pose-graph optimization on `sphere2500` with initialization $\sigma = 0.1$

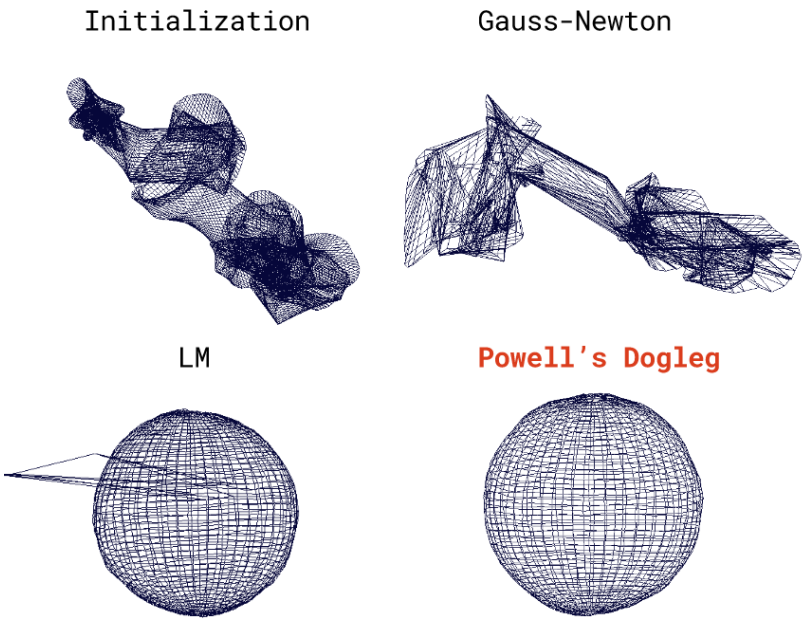


Figure 5: Pose-graph optimization on `sphere2500` with initialization $\sigma = 0.3$

Table 1: Results for $\sigma = 0.1$

Method	Residual	Time (s)	Iterations
Gauss-Newton	2.43659e+07	109.733	500 (max)
Levenberg-Marquardt	11976.5	37.9395	221
Powell's Dogleg	8282.07	18.5653	82

Table 2: Results for $\sigma = 0.3$

Method	Residual	Time (s)	Iterations
Gauss-Newton	7.10762e+06	110.263	500 (max)
Levenberg-Marquardt	8290.98	81.1136	500 (max)
Powell's Dogleg	8290.9	22.8722	112

6.1 Inference

We find that Gauss-Newton fails completely in either case. This is because the method doesn't converge due to its non-linearity. Powell's Dogleg and Levenberg-Marquadt reconstruct the sphere well, with a low sum of square difference. However, Dogleg is significantly faster than LM, by a factor of 2 and 4. This is due to the large number of steps rejected by LM, where the information matrix has to be re-computed. LM and Dogleg also outperform Gauss-Newton in terms of robustness in the presence of noise.

7 Conclusion

In summary, the purpose of this project was to (i) study the different optimization methods (ii) compare their advantages in pose-graph problems (iii) test them out on a reference dataset, to confirm our comparisons. All of the above has been successfully shown in the sections above. We thus reason that PDL is an algorithm that is well suited for large scale optimization problems. This notion is backed up by related work such as [10].

References

[1] http://www-personal.acfr.usyd.edu.au/nebot/victoria_park.html

[2] Kaess, Michael, Ananth Ranganathan, and Frank Dellaert. "iSAM: Incremental smoothing and mapping." IEEE Transactions on Robotics 24.6 (2008): 1365-1378.

[3] Agarwal, Sameer, and Keir Mierle. "Ceres solver." (2012): 132.

[4] Dellaert, Frank. Factor graphs and GTSAM: A hands-on introduction. Georgia Institute of Technology, 2012.

[5] Moré, Jorge J. "The Levenberg-Marquardt algorithm: implementation and theory." Numerical analysis. Springer, Berlin, Heidelberg, 1978. 105-116.

[6] https://people.maths.ox.ac.uk/hauser/hauser_lecture3.pdf

[7] Westerberg, Arthur W., and Stephen W. Director. "A modified least squares algorithm for solving sparse nxn sets of nonlinear equations." Computers and Chemical Engineering 2.2-3 (1978): 77-81.

[8] Powell, M. J. D. "Convergence properties of a class of minimization algorithms." Nonlinear programming 2.0 (1975): 1-27.

[9] Rosen, David M., Michael Kaess, and John J. Leonard. "RISE: An incremental trust-region method for robust online sparse least-squares estimation." IEEE Transactions on Robotics 30.5 (2014): 1091-1108.

[10] Lourakis, M. L. A., and Antonis A. Argyros. "Is Levenberg-Marquardt the most efficient optimization algorithm for implementing bundle adjustment?." Computer Vision, 2005. ICCV 2005. Tenth IEEE International Conference on. Vol. 2. IEEE, 2005.