SI152 Final Report

Jinxi Xiao[*]
ShanghaiTech University
`xiaojx@shanghaitech.edu.cn`

Abstract

This report is focused on the paper *An Adaptive Multi-step Levenberg–Marquardt Method* (1), which proposes a novel approch on improving the original algorithm. In this report, I will give some of my own understanding of the improved levenberg-marquardt(LM) method and implement the code to further explore some of the applications.[2]

1 Basic Terminologies

1.1 Levenberg-Marquardt Algorithm

We aim to solve

$$F(x) = 0 \tag{1}$$

where $F(x) : \mathbb{R}^n \to \mathbb{R}^m$. Here we make two assumptions that are used in the whole report:

Assumption 1 $F(x) = 0$ has a non-empty solution set X^*.
Assumption 2 F is continuously differentiable.

We may also observe a nonlinear least square version:

$$\min_x \frac{1}{2}||F(x)||_2^2 \tag{2}$$

It would be quite obvious that Equation 1 has a solution if and only if the minimum of 2 is zero. And for least square problems, we can design an iterative algorithm, which is the levenberg-marquardt(LM) algorithm to handle them:

1. Initialize the starting point $x_0,\ \mu_0$ and the threshold ϵ.
2. For the k-th iteration, find

   $$\Delta x_k = \arg\min_{\Delta x_k} ||F(x_k + \Delta x_k)||_2^2$$

   subject to $||\Delta x_k||_2 \le \mu_k$.
3. If $\Delta x_k < \epsilon$, terminates the program.
4. Otherwise, define $x_{k+1} = x_k + \Delta x_k$, and update μ_k. Return to step 2.

In LM method, the general opdate step of Δx is given by

$$d_k = -(J_k^T J_k + \lambda_k I)^{-1} J_k^T F_k \tag{3}$$

where $J_k \in \mathbb{R}^{m\times n}$ to be the Jacobian matrix computed at x_k, and λ_k is a positive parameter.

One of the most interesting things of LM method is that it has a Q-quadratic (Q-Order 2) convergence rate under certain mild conditions.

[*]https://xiaojxkevin.github.io/
[2]Codes can be found in https://github.com/xiaojxkevin/SI152_final_project

1.2 Local Error Bound Condition

This concept is originally from (2), which is used to analyse the convergence rate of LM. And this condition is crucial in the following contents.

Definition 1 Let $\mathcal{N}$ be a subset of $\mathbb{R}^n$ such that $X^* \cap \mathcal{N} \neq \emptyset$. We say that $F(x)$ provides a *local error bound* on $\mathcal{N}$ for system 1 if there exists a positive constant c such that

$$c\,\mathrm{dist}(x, X^*) \leq \|F(x)\|_2 \quad \forall\, x \in \mathcal{N}. \tag{4}$$

Notice that X^* is a set, thus we use dist to denote the distance of a point to the set.

2 Adaptive Multi-Step LM Algorithm

2.1 Motivation and the algorithm

Notice that in each step, LM algorithm requires to compute the Jacobian, which will be time-consuming when the dimensions of data are large. As a result, authors try to improve the original method. To be more specific, they notice that under certain circumstances, there's no need to compute the Jacobaian at every iteration, instead, the Jacobian computed last step can be directly used, which is called as an *approximate LM step*. This idea is inherited from authors' former paper (3). However, we will not discuss it in this report. After this modification, the time performance of the adaptive LM method will be better while still preserves q-superlinear convergence rate theoretically.

We constraint the number of approximate LM steps to be at most $t-1$, where $t \geq 1$ is a given integer. Counting every LM step and every approximate LM step as a single iteration, we compute the step at each iteration d_k as:

$$\left(G_k^T G_k + \lambda_k I\right) d_k = -\, G_k^T F_k \tag{5}$$

where G_k is the Jacobian J_k or the Jacobian used at the last iteration. If d_k is satisfactory and less than $t-1$ approximate LM steps are computed, we regard G_k as a good approximation of the Jacobian at the current iteration, and use it to compute another approximate LM step at next iteration. Define the ratio of the actual reduction to the predicted reduction of the merit function $\|F(x)\|_2^2$ at the k-th iteration as

$$r_k = \frac{\mathrm{Ared}_k}{\mathrm{Pred}_k} = \frac{\|F_k\|_2^2 - \|F(x_k + d_k)\|_2^2}{\|F_k\|_2^2 - \|F_k + G_k d_k\|_2^2} \tag{6}$$

The ratio r_k exploits the most relevant information on the step's quality at the iterate. It plays an important role in deciding whether d_k is acceptable. If r_k is positive (i.e., the iteration is successful), we accept d_k. Usually, we set

$$x_{k+1} = \begin{cases} x_k + d_k, & \text{if} \quad r_k \geq p_0, \\ x_k, & \text{otherwise,} \end{cases} \tag{7}$$

where p_0 is a small positive constant. Suppose $s-1$ approximate LM steps that used the latest evaluated Jacobian have been computed. If the iteration is very successful and s is less than t, we regard the latest evaluated Jacobian as a good approximation at the iterate and keep it, otherwise, we evaluate the exact one. That is, we set

$$G_{k+1} = \begin{cases} G_k, & \text{if} \quad r_k \geq p_1 \text{ and } s < t, \\ J_{k+1}, & \text{otherwise} \end{cases} \tag{8}$$

where $0 < p_0 < p_1 < 1$. Meanwhile, we update the LM parameter as follows:

$$\lambda_{k+1} = \begin{cases} \lambda_k, & \text{if} \quad r_k \geq p_1 \text{ and } s < t, \\ \mu_{k+1}\|F_{k+1}\|_2^\delta, & \text{otherwise} \end{cases} \tag{9}$$

where

$$\mu_{k+1} = \begin{cases} c_1\mu_k, & \text{if} \quad r_k < p_2, \\ \mu_k, & \text{if} \quad p_2 \leq r_k \leq p_3, \\ \max\{c_2\mu_k, \mu_{\min}\}, & \text{if} \quad r_k > p_3 \end{cases} \tag{10}$$

Algorithm 1 Adaptive multi-step LM algorithm

```
Require: $c_1 > 1 > c_2 > 0,\ 0 < p_0 < p_2 < p_1 < p_3 < 1,\ 1 \le \delta \le 2,\ t \ge 1,$
         $x_1 \in \mathbb{R}^n,\ \mu_1 > \mu_{\min} > 0$ .
                    ▷ Subscripts for $x$ and $\mu$ indicate the index(starting from 1) of the iteration.

1: $G_1 = J_1,\ \lambda_1 = \mu_1 \|F_1\|_2^\delta,\ k := 1,\ s := 1,\ i := 1,\ k_i = 1.$   ▷
       • $k$: index of the iteration.
       • $s$: For a certain $G_l$, # of times we have used it. $s < t$ for sure.
       • $i$: # of times we compute a new Jacobian.
       • $k_i$: index of the iteration where we compute a new Jacobian.
2: while $\|G_{k_i}^T F_{k_i}\|_2 \neq 0$ do
3:     Compute $d_k$ by solving Equation 5.
4:     Compute $r_k = \mathrm{Ared}_k / \mathrm{Pred}_k$ by Equation 6, and set $x_{k+1}$ by Equation 7.
5:     Update $G_{k+1}$, $\lambda_{k+1}$ and $\mu_{k+1}$ by Equation 8, 9 and 10 respectively.
6:     Set $k := k + 1$. If $G_k$ is the Jacobian at $x_k$, set $s := 1,\ i := i + 1,\ k_i = k$, otherwise set
   $s := s + 1$
7: end while
```

Here, $0 < c_2 < 1 < c_1$, $0 < p_0 < p_2 < p_1 < p_3 < 1$, $1 \le \delta \le 2$ and $\mu_{\min} > 0$ are constants. In sum, the general algorithm is shown in Algorithm 1.

Notice that when $t = 1$, the algorithm degenerates to the classic LM method.

2.2 Convergence rate

Details of the proof of the q-superlinear convergence rate of Algorithm 1 can be found in paper (1) Section 2 and Section 3. And I will only briefly list some of the key points.

2.2.1 Global Convergence

Theorem: If both $F(x)$ and $J(x)$ are *Lipschitz continuous* on $\mathbb{R}^n$, then we have

$$\lim_{k\to\infty} \inf \|J_k^T F_k\|_2 = 0 \tag{11}$$

2.2.2 Local Convergence

We make the following more assumptions:

- The sequence $\{x_k\}$ generated by Algorithm 1 satisfies $\mathrm{dist}(x_k, X^*) \to 0$, and there exist $x^* \in X^*$ and $0 < r < 1$ such that $\|x_k - x^*\| \le r/2$ for all large k.
- $J(x)$ is Lipschitz continuous on $N(x^*, r) = \{x : \|x - x^*\| \le r\}$.
- $\|F(x)\|_2$ provides a local error bound on $N(x^*, r)$.

Define $s_i = k_{i+1} - k_i$ to be the gap between two iterations where we have computed a new Jacobian. Notice that $1 \le s_i \le t$.

Theorem: There exists a constant $l_3 > 0$ such that

$$\|d_{k_{i+1}}\|_2 \le l_3 \|d_{k_i}\|_2^{s_i+1} \tag{12}$$

3 Simulation Results

To further test the performance of the adaptive multi-step LM algorithm, I first try this on a simulated problem, which is the *Rosenbrock function*. To be more specific, it is defined as

$$F(x) = \sum_{i=1}^{M-1} \left[100\left(x_{i+1} - x_i^2\right)^2 + (1 - x_i)^2\right] \tag{13}$$

where $x \in \mathbb{R}^M$. Notice that Equation 13 has a global minimum at $x = [1, 1, ..., 1]^T$. However, it would be extremely difficult to reach the global minimum when the dimension M becomes large. During the process, we compute the Jacobian as the following:

$$\frac{\partial F}{\partial x_i} = \begin{cases} 400x_1\left(x_1^2 - x_2\right) + 2(x_1 - 1), & i = 1 \\ 200\left(x_i - x_{i-1}^2\right) + 400x_i\left(x_i^2 - x_{i+1}\right) + 2(x_i - 1), & i = 2, ..., M-1 \\ 200\left(x_M - x_{M-1}^2\right), & i = M \end{cases} \tag{14}$$

In the numerical test, I have set $c_1 = 4$, $c_2 = 0.25$, $p_0 = 10^{-4}$, $p_1 = 0.5$, $p_2 = 0.25$, $p_3 = 0.75$, $\mu_{\min} = 10^{-5}$, $\delta = 2$ and the threshold for convergence would be 10^{-5}. In addition, I initialize $x_1 \sim$ Gaussian$(0, I)$ and $\mu_1 = 0.2$. Later on, I test the Rosenbrock function on both LM and Adaptive Multi-step LM algorithms, and record the number of function F computed(NF), number of Jacobian computed(NJ) and convergence time(unit as second). Also notice that there's no maximum iteration number, i.e. the algorithm terminates when the threshold is reached. Some numerical results can be found in Table 1.

We can clearly see that when we set $t = 5$, the algorithm achieves a comparative best result, which implies that it isn't always good to use the Jacobian computed in the last iteration.

Dimension M	t	LM NF/NJ/TIME(s)	Adaptive-LM NF/NJ/TIME(s)
2	3	3363/3363/0.1191	1687/977/0.0625
	5		673/361/0.0312
	8		1398/694/0.0469
	10		2710/1290/0.0937
8	3	9384/9384/1.1695	5135/2833/0.5041
	5		3877/2025/0.3909
	8		4870/2390/0.4739
	10		8579/3999/0.8129
20	3	13144/13144/3.8105	7053/3904/1.6766
	5		5704/2978/1.3346
	8		6154/2977/1.4137
	10		11244/5301/2.5698

Table 1: Numerical results on Rosenbrock Function with different hyperparameters.

3.1 A Visual Explanation

To have a further understanding of the algorithm, I visualize the adaptive multi-step LM algorithm on a 2D case, where I have initialized the starting point to be $[-1.5, -1.5]^T$. Result is shown in Figure 1. The reason for a sudden jump is that the shape of Rosenbrock function is like a valley, which further explains the difficult of optimizing on such function.

4 Toy Application: Inverse Kinematics

Inverse Kinematics problems are widely seen in the field of robotics. Some of the basic introductions can be found in *link1* and *link2*. Also, I have learnt that Gaussian-Newton method can be combined with neural networks to solve the inverse kinematics on 3D human poses (4). In this section, I will show that the adaptive multi-step LM algorithm can be used to solve the problem of IK with a 2-link case(shown in Figure 2).

We define the observed coordinate of the end-effector as

$$F\left(\begin{bmatrix}\theta_1 \\ \theta_2\end{bmatrix}\right) = \begin{bmatrix} l_2\cos(\theta_1 + \theta_2) + l_1\cos(\theta_1) \\ l_2\sin(\theta_1 + \theta_2) + l_1\sin(\theta_1) \end{bmatrix} \tag{15}$$

where l_1, l_2 are the length of two arms respectively, and $0 < \theta_1 < \frac{\pi}{2}$, $-\frac{\pi}{2} < \theta_2 < \frac{\pi}{2}$ to be the orientation of two arms.

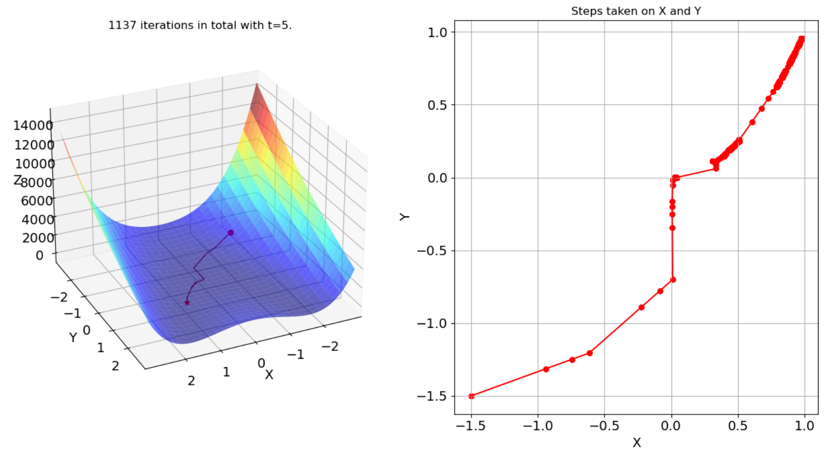

Figure 1: Visualize Result for a finding a solution for a 2D Rosenbrock function using Adaptive Multi-step LM algorithm.

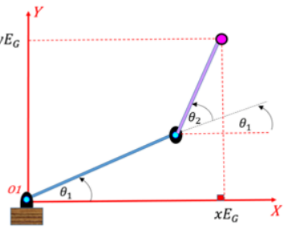

Figure 2: Two-Link Robot Arm

In each step, we compute the Jacobian matrix as

$$\nabla F = \begin{bmatrix} -(l_2\sin(\theta_1+\theta_2)+l_1\sin(\theta_1)) & -l_1\sin(\theta_1+\theta_2) \\ l_2\cos(\theta_1+\theta_2)+l_1\cos(\theta_1) & l_1\cos(\theta_1+\theta_2) \end{bmatrix} \tag{16}$$

In test case, I have set gt to be $\theta = [\frac{\pi}{3}, -\frac{\pi}{4}]^T$ and other parameters for the algorithm stay consistent to the previous section. The final result is interesting for there are two distinct solution if I initialize the starting point to be different. That is, I have obtained another valid solution: $[\frac{\pi}{6}, \frac{\pi}{4}]^T$. This phenomenon is trivial for there is indeed no guarantee for the uniqueness of the solution in IK problems.

References

[1] J. Fan, J. Huang, and J. Pan, “An adaptive multi-step levenberg–marquardt method,” *Journal of Scientific Computing*, vol. 78, pp. 531–548, 2019.

[2] N. Yamashita and M. Fukushima, “On the rate of convergence of the levenberg-marquardt method,” in *Topics in Numerical Analysis: With Special Emphasis on Nonlinear Problems*, pp. 239–249, Springer, 2001.

[3] J. Fan, “A shamanskii-like levenberg-marquardt method for nonlinear equations,” *Computational Optimization and Applications*, vol. 56, no. 1, pp. 63–80, 2013.

[4] J. Zhang, Y. Shi, Y. Ma, L. Xu, J. Yu, and J. Wang, “Ikol: Inverse kinematics optimization layer for 3d human pose and shape estimation via gauss-newton differentiation,” 2023.

A Math Proof for Equation 3

To construct the Lagrange dual form of the minimization problem(I will ignore subscripts for simplicity):

$$\mathcal{L}(\Delta x, \lambda) = \frac{1}{2}\|F(x) + J\Delta x\|_2^2 + \frac{1}{2}(\|\Delta x\|_2^2 - \mu) \tag{17}$$

where λ must be a non-negative parameter. Compute the derivative of Δx and set it to zero, then we will arrive at the solution.