Applied Mathematics Letters 112 (2021) 106739

Contents lists available at ScienceDirect

Applied Mathematics Letters

www.elsevier.com/locate/aml

A gradient descent method for solving a system of nonlinear equations

Wenrui Hao

Penn state university, University Park, PA 16802, United States of America

ARTICLE INFO

Article history:
Received 26 July 2020
Received in revised form 29 August 2020
Accepted 29 August 2020
Available online 5 September 2020

Keywords:
Gradient descent method
Systems of nonlinear equations
Convergence analysis

ABSTRACT

This paper develops a gradient descent (GD) method for solving a system of nonlinear equations with an explicit formulation. We theoretically prove that the GD method has linear convergence in general and, under certain conditions, is equivalent to Newton's method locally with quadratic convergence. A stochastic version of the gradient descent is also proposed for solving large-scale systems of nonlinear equations. Finally, several benchmark numerical examples are used to demonstrate the feasibility and efficiency compared to Newton's method.

1. Introduction

Systems of nonlinear equations are omnipresent and play crucial roles in scientific computing and mathematical modeling ranging from differential equations [1,2], integral equations [3,4], and optimization problems [5,6] to data-driven modeling and machine learning [7,8]. To date, Newton's, Newton-like methods (e.g. Gauss–Newton and Quasi-Newton methods [9,10]) have widely been used for solving systems of nonlinear equations; however, the computational cost of solving the linear system at each stage can be expensive for large-scale systems [11]. Recently, several methods have been developed to overcome this challenge: inexact Newton methods solve the Newton equations only approximately and in some unspecified manner [12]; Newton–Krylov methods use Krylov subspace methods to solve the linear system which mostly requires only products of the Jacobian matrix with vectors and can be implemented as "matrix-free" format [13]. But the expensive computational cost still limits the application of Newton's method to data-driven modeling problems [14] and optimization problems arising from machine learning [15]. This is one of the reasons that people do not use Newton's method for these problems although it has quadratic convergence locally.

On the other hand, systems of nonlinear equations are related to unconstrained optimizations and nonlinear least-squares. Therefore we may apply the algorithms in solving nonlinear least-squares problems

E-mail address: wxh64@psu.edu.

https://doi.org/10.1016/j.aml.2020.106739
0893-9659/© 2020 Elsevier Ltd. All rights reserved.

to solve systems of nonlinear equations by minimizing the sum of squares of the equations. Recently, the first-order iterative optimization algorithms, e.g., the GD method [16] and the Adam algorithm [17], have widely been used for solving large-scale optimization problems arising from machine learning. These algorithms do not need to solve a linear system each iteration which is the advantage compared to Newton-like methods. For instance, a modification of GD method has been developed to provide a continuous solution path in the homotopy setup [18]. However, updating the step size for each iteration relies on the trust-region and line-search algorithms and limits the direct application of the GD method to solving the system of nonlinear equations. In particular, these algorithms may be time-consuming especially for large-scale systems since they need the backtracking procedure, verify the Wolfe conditions repeatedly, or the information of previous solutions [19]. In this paper, we develop a GD method for solving the system of nonlinear equations by deriving the explicit formula of the stepsize. Moreover, we explore the theoretical convergence of the GD method and apply it to several benchmark systems of nonlinear equations. Therefore, the newly developed GD method does not require a linear solver like Newton's method and other line-search algorithms for solving systems of nonlinear equations. By introducing stochasticity, it also provides a new way to solve large-scale systems of nonlinear equations by using only a part of nonlinear equations for each iteration.

2. The problem setup and gradient descent method

We consider a general system of nonlinear equations below

$$\boldsymbol{F}(\boldsymbol{x}) = \begin{pmatrix} F_1(\boldsymbol{x}), F_2(\boldsymbol{x}), \dots, F_m(\boldsymbol{x}) \end{pmatrix}^T = \boldsymbol{0}, \tag{1}$$

where $\boldsymbol{x} = (x_1, \dots, x_n) \in \mathbb{R}^n$ is the variable and F_i is a nonlinear equation. The system is called a square system if $n = m$. Otherwise it is called an underdetermined system $(n > m)$ and an overdetermined system $(n < m)$. Then solving the system (1) is equivalent to solving the following least minimization problem

$$\min \frac{1}{2}\|\mathbf{F}(\mathbf{x})\|_2^2. \tag{2}$$

The GD method for solving the above optimization problem is written as

$$\boldsymbol{x}^{k+1} = \boldsymbol{x}^k - \eta \nabla \boldsymbol{F}(\boldsymbol{x}^k)^T \boldsymbol{F}(\boldsymbol{x}^k), \tag{3}$$

where $\nabla \boldsymbol{F}(\boldsymbol{x}^k)$ is the Jacobian matrix at $\boldsymbol{x}^k$ and η is the stepsize. There are a variety of line search algorithms [20] to compute the stepsize such as the BB Method [21], the Cauchy step-size [22], the alternate minimization [23], the random choice of stepsize [24], and etc. All the aforementioned line search algorithms are for the general optimization problem not for the system of nonlinear equations and therefore the convergence might be very slow for large-scale systems of nonlinear equations.

2.1. How to choose η?

In this paper, we propose a new explicit formula of the stepsize η for solving systems of nonlinear equations. First, we write the GD method as $\boldsymbol{x}^{k+1} = \boldsymbol{x}^k - \eta \boldsymbol{p}^k$, where $\boldsymbol{p}^k = \nabla \boldsymbol{F}(\boldsymbol{x}^k)^T \boldsymbol{F}(\boldsymbol{x}^k)$. By the Taylor expansion, we have $\boldsymbol{F}(\boldsymbol{x}^{k+1}) \approx \boldsymbol{F}(\boldsymbol{x}^k) - \eta \nabla \boldsymbol{F}(\boldsymbol{x}^k)\boldsymbol{p}^k \approx 0$, which implies that

$$\eta = \frac{\boldsymbol{v}^T \boldsymbol{F}(\boldsymbol{x}^k)}{\boldsymbol{v}^T \nabla \boldsymbol{F}(\boldsymbol{x}^k)\boldsymbol{p}^k} \tag{4}$$

for any given $\boldsymbol{v}$. Moreover, in order to guarantee the convergence, we choose $\boldsymbol{v} = \nabla \boldsymbol{F}(\boldsymbol{x}^k)\boldsymbol{p}^k$ in (4) such that $\boldsymbol{F}(\boldsymbol{x}^{k+1})^T \boldsymbol{F}(\boldsymbol{x}^{k+1}) \le \boldsymbol{F}(\boldsymbol{x}^k)^T \boldsymbol{F}(\boldsymbol{x}^k)$. In fact,

$$\begin{aligned} \boldsymbol{F}(\boldsymbol{x}^{k+1})^T \boldsymbol{F}(\boldsymbol{x}^{k+1}) &= \boldsymbol{F}(\boldsymbol{x}^k)^T \boldsymbol{F}(\boldsymbol{x}^k) + \eta^2 \big(\nabla \boldsymbol{F}(\boldsymbol{x}^k)\boldsymbol{p}^k\big)^T \nabla \boldsymbol{F}(\boldsymbol{x}^k)\boldsymbol{p}^k - 2\eta \big(\nabla \boldsymbol{F}(\boldsymbol{x}^k)\boldsymbol{p}^k\big)^T \boldsymbol{F}(\boldsymbol{x}^k) \\ &= \boldsymbol{F}(\boldsymbol{x}^k)^T \boldsymbol{F}(\boldsymbol{x}^k) - \frac{(\boldsymbol{v}^T \boldsymbol{F}(\boldsymbol{x}^k))^2}{\boldsymbol{v}^T \boldsymbol{v}} \le \boldsymbol{F}(\boldsymbol{x}^k)^T \boldsymbol{F}(\boldsymbol{x}^k), \end{aligned} \tag{5}$$

which implies the convergence of the GD method. Then we summarize the GD method for solving (1) as follows

$$\boldsymbol{x}^{k+1} = \boldsymbol{x}^k - \eta \nabla \boldsymbol{F}(\boldsymbol{x}^k)^T \boldsymbol{F}(\boldsymbol{x}^k),\ \eta = \frac{\boldsymbol{v}^T \boldsymbol{F}(\boldsymbol{x}^k)}{\boldsymbol{v}^T \boldsymbol{v}},\ \text{and } \boldsymbol{v} = \nabla \boldsymbol{F}(\boldsymbol{x}^k) \nabla \boldsymbol{F}(\boldsymbol{x}^k)^T \boldsymbol{F}(\boldsymbol{x}^k). \tag{6}$$

In particular, when solving a linear system, namely, $\boldsymbol{F}(\boldsymbol{x}) := A\boldsymbol{x} - b$, the GD method becomes

$$\boldsymbol{x}^{k+1} = \boldsymbol{x}^k - \eta A^T (A\boldsymbol{x}^k - b) \text{ and } \eta = \frac{(A\boldsymbol{x}^k - b)^T A A^T (A\boldsymbol{x}^k - b)}{(A\boldsymbol{x}^k - b)^T A A^T A A^T (A\boldsymbol{x}^k - b)}. \tag{7}$$

2.2. Stochastic gradient descent

For the large-scale system of nonlinear equations, we use the stochastic gradient descent (SGD) method to reduce the computational cost by solving a part of the original system, namely, randomly choose s equations from $\boldsymbol{F}(\boldsymbol{x})$ for each iteration. In order to introduce the stochasticity, we define a random variable ξ on a probability space $(\Omega, \mathcal{F}, \mathcal{P})$: $\xi : \Omega \to \Gamma$, where Γ is a set with all the combinations of s numbers out of $\{1, 2, \ldots, m\}$. Since $|\Gamma| = \binom{m}{s}$, we denote $\Gamma = \{\gamma_1, \gamma_2, \ldots, \gamma_{|\Gamma|}\}$, and assume the random variable ξ follow the uniform distribution, namely, $\mathcal{P}(\xi = \gamma_i) = \frac{1}{|\Gamma|}$ for $1 \le i \le |\Gamma|$. By defining $\boldsymbol{F}(\boldsymbol{x}, \gamma_i) := \big(F_{i_1}(\boldsymbol{x}), F_{i_2}(\boldsymbol{x}), \cdots, F_{i_s}(\boldsymbol{x}) \quad \big)^T$, where $\gamma_i = \{i_1, \ldots, i_s\} \subset \{1, \ldots, m\}$, we write the SGD method as

$$\boldsymbol{x}^{k+1} = \boldsymbol{x}^k - \eta \nabla \boldsymbol{F}(\boldsymbol{x}^k, \xi_k)^T \boldsymbol{F}(\boldsymbol{x}^k, \xi_k),\ \eta = \frac{\boldsymbol{v}^T \boldsymbol{F}(\boldsymbol{x}^k, \xi_k)}{\boldsymbol{v}^T \boldsymbol{v}},\ \text{and } \boldsymbol{v} = \nabla \boldsymbol{F}(\boldsymbol{x}^k, \xi_k) \nabla \boldsymbol{F}(\boldsymbol{x}^k, \xi_k)^T \boldsymbol{F}(\boldsymbol{x}^k, \xi_k).$$

Then by (5), we have

$$\begin{aligned} \boldsymbol{F}(\boldsymbol{x}^{k+1})^T \boldsymbol{F}(\boldsymbol{x}^{k+1}) &\triangleq \mathbb{E}\big(\boldsymbol{F}(\boldsymbol{x}^{k+1}, \xi_{k+1})^T \boldsymbol{F}(\boldsymbol{x}^{k+1}, \xi_{k+1})\big) = \mathbb{E}\big(\boldsymbol{F}(\boldsymbol{x}^k, \xi_k)^T \boldsymbol{F}(\boldsymbol{x}^k, \xi_k)\big) - \mathbb{E}\Big(\frac{(\boldsymbol{v}^T \boldsymbol{F}(\boldsymbol{x}^k, \xi_k))^2}{\boldsymbol{v}^T \boldsymbol{v}}\Big) \\ &\le \mathbb{E}\big(\boldsymbol{F}(\boldsymbol{x}^k, \xi_k)^T \boldsymbol{F}(\boldsymbol{x}^k, \xi_k)\big) = \boldsymbol{F}(\boldsymbol{x}^k)^T \boldsymbol{F}(\boldsymbol{x}^k), \end{aligned} \tag{8}$$

which implies the convergence of the SGD method.

3. Convergence analysis

In this section, we consider the convergence analysis of the GD method in a general case: the rank of the Jacobian $\nabla \boldsymbol{F}(\boldsymbol{x})$ is r which might not equal to m and n. When $r = m = n$, it is an isolated solution of a square system of nonlinear equations. If $r < m$ and $r < n$, we have positive dimensional solution sets for a system of nonlinear equations. Then we have the following two theorems: the first one is linear convergence as long as the solution is not singular; the second one is quadratic convergence if the Jacobian matrix at the solution has a special structure. We refer the convergence of the general SGD method in [25] and quadratic convergence of stochastic Newton's method in [26].

Theorem 1. *Let $\boldsymbol{F}$ be a smooth function and the Jacobian $\nabla \boldsymbol{F}(\boldsymbol{x})$ is a rank-r matrix in a neighborhood of a zero $\boldsymbol{x}^*$, then the gradient descent method converges linearly to $\boldsymbol{x}^*$ if the initial guess $\boldsymbol{x}^0$ is in the neighborhood of $\boldsymbol{x}^*$.*

Proof. We define the numerical error of the kth iteration as $\boldsymbol{E}^k = \boldsymbol{x}^k - \boldsymbol{x}^*$ and have

$$\boldsymbol{x}^{k+1} - \boldsymbol{x}^* = \boldsymbol{x}^k - \boldsymbol{x}^* - \eta \nabla F(\boldsymbol{x}^k)^T F(\boldsymbol{x}^k). \tag{9}$$

Since $\boldsymbol{F}$ is a smooth function in the neighborhood of $\boldsymbol{x}^*$, we have the following Taylor expansion:

$$\boldsymbol{F}(\boldsymbol{x}^k) = \boldsymbol{F}(\boldsymbol{x}^*) + \nabla\boldsymbol{F}(\boldsymbol{x}^*)\boldsymbol{E}^k + \sum_{i=1}^m (\boldsymbol{E}^k)^T H_i(\boldsymbol{x}^*)\boldsymbol{E}^k \boldsymbol{e}_i + O(\|\boldsymbol{E}^k\|^3), \tag{10}$$

$$\nabla\boldsymbol{F}(\boldsymbol{x}^k) = \nabla\boldsymbol{F}(\boldsymbol{x}^*) + \sum_{i=1}^m e_i \big(H_i(\boldsymbol{x}^*)\boldsymbol{E}^k\big)^T + O(\|\boldsymbol{E}^k\|^2), \tag{11}$$

where $H_i(\boldsymbol{x})$ is the Hessian matrix of $F_i(\boldsymbol{x})$ and e_i is the standard basis of $\mathbb{R}^m$. Therefore (9) becomes

$$\boldsymbol{E}^{k+1} = \boldsymbol{E}^k - \eta\nabla\boldsymbol{F}(\boldsymbol{x}^*)^T\nabla\boldsymbol{F}(\boldsymbol{x}^*)\boldsymbol{E}^k + \sum_i (\boldsymbol{E}^k)^T H_i(\boldsymbol{x}^*)\boldsymbol{E}^k \mathbf{e}_i + O(\|\boldsymbol{E}^k\|^3). \tag{12}$$

By denoting $A = \nabla F(\boldsymbol{x}^*)^T\nabla F(\boldsymbol{x}^*)$ which is a symmetric matrix, we have $\boldsymbol{E}^{k+1} = (I - \eta A)\boldsymbol{E}^k + O(\|\boldsymbol{E}^k\|^2)$. Since $\nabla F(\boldsymbol{x}^*)$ is a rank-r matrix, we have $\nabla F(\boldsymbol{x}^*) = U\Sigma_r V^T$ and $A = V\Sigma_r^2 V^T$, where $\Sigma_r = diag([\sigma_1, \dots, \sigma_r, 0, \dots, 0]^T)$ and V is the orthogonal eigenvector matrix of A with the eigenvalues $\lambda_i = \sigma_i^2$ $(i = 1, \dots, r)$. Then we split the identity matrix $I = I_r + I_{n-r}$ where $I_r = diag([\underbrace{1, \dots, 1}_{r}, 0, \dots, 0]^T)$ and $I_{n-r} = diag([\underbrace{0, \dots, 0}_{r}, 1, \dots, 1]^T)$. Then we have

$$I_{n-r}V^T\boldsymbol{E}^{k+1} \approx I_{n-r}V^T\boldsymbol{E}^k \approx \cdots \approx I_{n-r}V^T\boldsymbol{E}^0, \tag{13}$$

which implies that the GD method does not update the solution on the kernel of A. Moreover, since $\boldsymbol{F}(\boldsymbol{x}^k) = \nabla\boldsymbol{F}(\boldsymbol{x}^*)\boldsymbol{E}^k + O(\|\boldsymbol{E}^k\|^2)$, we have

$$I_{n-r}U^T\boldsymbol{F}(\boldsymbol{x}^{k+1}) \approx I_{n-r}U^T\boldsymbol{F}(\boldsymbol{x}^k) \approx \cdots \approx I_{n-r}U^T\boldsymbol{F}(\boldsymbol{x}^0) = 0. \tag{14}$$

Therefore, we consider the convergence on $I_rV^T\boldsymbol{E}^k$ and $I_rU^T\boldsymbol{F}(\boldsymbol{x}^k)$ only.

Then in (5), by the Taylor expansion, we have

$$\boldsymbol{v}^T\boldsymbol{F}(\boldsymbol{x}^k) = \boldsymbol{F}(\boldsymbol{x}^k)^T\nabla\boldsymbol{F}(\boldsymbol{x}^k)\nabla\boldsymbol{F}(\boldsymbol{x}^k)^T\boldsymbol{F}(\boldsymbol{x}^k) = (\boldsymbol{E}^k)^T A^2\boldsymbol{E}^k + O(\|\boldsymbol{E}^k\|^3), \tag{15}$$

where $A^2 = AA$ is a rank-r positive definite matrix. Therefore, $\boldsymbol{v}^T\boldsymbol{F}(\boldsymbol{x}^k) \geq 0$ and becomes zero if and only if $I_rV^T\boldsymbol{E}^k = 0$. Similarly, we denote $B = \nabla F(\boldsymbol{x}^*)\nabla F(\boldsymbol{x}^*)^T$ and have

$$\boldsymbol{v}^T\boldsymbol{v} = F(\boldsymbol{x}^k)^T\nabla\boldsymbol{F}(\boldsymbol{x}^k)\nabla\boldsymbol{F}(\boldsymbol{x}^k)^T\nabla\boldsymbol{F}(\boldsymbol{x}^k)\nabla\boldsymbol{F}(\boldsymbol{x}^k)^T\boldsymbol{F}(\boldsymbol{x}^k) = \boldsymbol{F}(\boldsymbol{x}^k)^T B^2\boldsymbol{F}(\boldsymbol{x}^k) + O(\|\boldsymbol{E}^k\|^3),$$

which implies that $\boldsymbol{v}^T\boldsymbol{v} \geq 0$ and vanishes if and only if $I_rU^T\boldsymbol{F}(\boldsymbol{x}^k) = 0$. Therefore

$$\boldsymbol{v}^T\boldsymbol{v} \leq \max_i \lambda_i^2\|I_rU^T\boldsymbol{F}(\boldsymbol{x}^k)\|_2^2 \text{ and } \boldsymbol{v}^T\boldsymbol{F}(\boldsymbol{x}^k) \geq \min_i \lambda_i^2\|I_rV^T\boldsymbol{E}^k\|_2^2. \tag{16}$$

Moreover $\boldsymbol{F}(\boldsymbol{x}^k) = \nabla\boldsymbol{F}(\boldsymbol{x}^*)\boldsymbol{E}^k + O(\|\boldsymbol{E}^k\|^2)$, we have

$$\|I_rU^T\boldsymbol{F}(\boldsymbol{x}^k)\|_2^2 \leq \max_i \lambda_i\|I_rV^T\boldsymbol{E}^k\|_2^2, \tag{17}$$

which implies that

$$\big(\boldsymbol{v}^T\boldsymbol{F}(\boldsymbol{x}^k)\big)^2 \geq \frac{\min_i \lambda_i^4}{\max_i \lambda_i^2}\|I_rU^T\boldsymbol{F}(\boldsymbol{x}^k)\|_2^4. \tag{18}$$

Then we have the following estimate by (5)

$$\|\boldsymbol{F}(\boldsymbol{x}^{k+1})\|_2^2 \leq \Big(1 - \frac{\min_i \lambda_i^4}{\max_i \lambda_i^4}\Big)\|\boldsymbol{F}(\boldsymbol{x}^k)\|_2^2, \tag{19}$$

which implies linear convergence. □

Based on linear convergence estimate, the convergence rate depends on the ratio of $\frac{\min_i \lambda_i}{\max_i \lambda_i}$. If $\min_i \lambda_i = \max_i \lambda_i$, then we have the GD method is equivalent to Newton's method with quadratic convergence.

Theorem 2. *Let $\boldsymbol{F}$ be a smooth function and the Jacobian $\nabla \boldsymbol{F}(\boldsymbol{x})$ is a rank-r matrix in a neighborhood of a zero $\boldsymbol{x}^*$, if the singular values of $\nabla \boldsymbol{F}(\boldsymbol{x}^*)$, $\sigma_i = \sigma$, $i = 1, \dots, r$, then the gradient descent method is equivalent to Newton's method in the neighborhood of $\boldsymbol{x}^*$ with quadratic convergence.*

Proof. Since the singular values of the Jacobian matrix keep the same, we have $\nabla \boldsymbol{F}(\boldsymbol{x}^*)^T \nabla \boldsymbol{F}(\boldsymbol{x}^*) = \sigma^2 I_r$ and $\nabla \boldsymbol{F}(\boldsymbol{x}^*)^\dagger = \frac{1}{\sigma^2} \nabla \boldsymbol{F}(\boldsymbol{x}^*)^T$. Moreover, due to the smoothness of $\boldsymbol{F}$, we have the Taylor expansion for η as follows:

$$\eta = \frac{(\boldsymbol{E}^k)^T \nabla \boldsymbol{F}(\boldsymbol{x}^*)^T \nabla \boldsymbol{F}(\boldsymbol{x}^*) \nabla \boldsymbol{F}(\boldsymbol{x}^*)^T \nabla \boldsymbol{F}(\boldsymbol{x}^*) \boldsymbol{E}^k}{(\boldsymbol{E}^k)^T \nabla F(\boldsymbol{x}^*)^T \nabla \boldsymbol{F}(\boldsymbol{x}^*) \nabla \boldsymbol{F}(\boldsymbol{x}^*)^T \nabla \boldsymbol{F}(\boldsymbol{x}^*) \nabla \boldsymbol{F}(\boldsymbol{x}^*)^T \nabla \boldsymbol{F}(\boldsymbol{x}^*) \boldsymbol{E}^k} = \frac{1}{\sigma^2}. \tag{20}$$

Therefore, the GD method is equivalent to Newton's scheme, namely, $\boldsymbol{x}^{k+1} = \boldsymbol{x}^k - \nabla \boldsymbol{F}(\boldsymbol{x}^k)^\dagger \boldsymbol{F}(\boldsymbol{x}^k)$.

Although there is a quadratic convergence analysis of Newton's method for the regular rank-r Jacobian matrix in [27], we provide a different proof of quadratic convergence in this paper. Due to the smoothness of $\boldsymbol{F}(\boldsymbol{x})$, we have

$$\nabla \boldsymbol{F}(\boldsymbol{x}^k)^\dagger = \nabla \boldsymbol{F}(\boldsymbol{x}^*)^\dagger + \sum_{i=1}^m e_i \big(C_i(\boldsymbol{x}^*) \boldsymbol{E}^k\big)^T + O(\|\boldsymbol{E}^k\|^2), \tag{21}$$

where $C_i(\boldsymbol{x}^*)$ is the second order Taylor expansion matrix function for $F_i(\boldsymbol{x})$. Therefore

$$\boldsymbol{E}^{k+1} = \boldsymbol{E}^k - \nabla \boldsymbol{F}(\boldsymbol{x}^*)^\dagger \nabla \boldsymbol{F}(\boldsymbol{x}^*) \boldsymbol{E}^k + \sum_{i=1}^m \big(\boldsymbol{E}^k\big)^T \nabla \boldsymbol{F}(\boldsymbol{x}^*) C_i(\boldsymbol{x}^*) \boldsymbol{E}^k e_i.$$

By denoting $H = \nabla \boldsymbol{F}(\boldsymbol{x}^*)^\dagger \nabla \boldsymbol{F}(\boldsymbol{x}^*) = [V_1, \dots, V_r][V_1, \dots, V_r]^T$, we have

$$H\boldsymbol{E}^{k+1} = H\boldsymbol{E}^k - H\nabla \boldsymbol{F}(\boldsymbol{x}^*)^\dagger \nabla \boldsymbol{F}(\boldsymbol{x}^*) \boldsymbol{E}^k + H \sum_{i=1}^m \big(\boldsymbol{E}^k\big)^T \nabla \boldsymbol{F}(\boldsymbol{x}^*) C_i(\boldsymbol{x}^*) \boldsymbol{E}^k e_i.$$

Since $H\nabla \boldsymbol{F}(\boldsymbol{x}^*) = \nabla F(\boldsymbol{r})^\dagger$, we obtain

$$H\boldsymbol{E}^{k+1} = H \sum_{i=1}^m \big(\boldsymbol{E}^k\big)^T \nabla \boldsymbol{F}(\boldsymbol{x}^*) C_i(\boldsymbol{x}^*) \boldsymbol{E}^k e_i, \tag{22}$$

which implies that $\|H\boldsymbol{E}^{k+1}\|_2 \le M \|H\boldsymbol{E}^k\|_2^2$. On the other hand, the projection on $\big(I - H\big)$ contributes the quadratic term only, in fact

$$\begin{aligned} \boldsymbol{F}(\boldsymbol{x}^* + (I-H)\boldsymbol{E}^k) &= \boldsymbol{F}(\boldsymbol{x}^*) + \nabla \boldsymbol{F}(\boldsymbol{x}^*)\big(I - H\big)\boldsymbol{E}^k + O(\|\boldsymbol{E}^k\|^2) \\ &= \nabla \boldsymbol{F}(\boldsymbol{x}^*)\boldsymbol{E}^k - \nabla F(\boldsymbol{x}^*) \nabla \boldsymbol{F}(\boldsymbol{x}^*)^\dagger \nabla \boldsymbol{F}(\boldsymbol{x}^*) \boldsymbol{E}^k + O(\|\boldsymbol{E}^k\|^2) = O(\|\boldsymbol{E}^k\|^2). \end{aligned}$$

Therefore, we have quadratic convergence for Newton's method. □

Remark.

- If the Jacobian matrix is orthogonal, namely, $\nabla \boldsymbol{F}(\boldsymbol{x}^k)^T \nabla \boldsymbol{F}(\boldsymbol{x}^k) = I$ for each $\boldsymbol{x}^k$, we have $\sigma = 1$ thus the GD method is equivalent to Newton's method [28]. It is very hard to achieve this condition since the Jacobian matrix is changing dynamically at each iteration.
- One special case of nonlinear equations with positive dimensional solution sets is the rank-one Jacobian matrix. If a positive dimensional solution set is defined by a manifold $\mathcal{M}(\boldsymbol{x}) = 0$, then a system of nonlinear equations containing this manifold is defined as $\boldsymbol{F}(\boldsymbol{x}) = \boldsymbol{G}(x)\mathcal{M}(\boldsymbol{x}) = 0$. For instance, a 3D

Table 1
The number of iterations (computing time is in the unit of a millisecond (ms)) between GD and Newton's methods in Example 4.1 with different initial guesses.

Initial guess	$(1,0,0)^T$	$(0,1,0)^T$	$(0,0,1)^T$	$(1,1,0)^T$	$(1,0,1)^T$	$(0,1,1)^T$	$(1,1,1)^T$
Newton	8 (2.1 ms)	7 (2 ms)	9 (2.5 ms)	7 (2.4 ms)	Diverged	Diverged	13 (1.3 ms)
GD	101 (3.9 ms)	23 (1.4 ms)	116 (7.4 ms)	22 (1.4 ms)	88 (3.4 ms)	102 (2.8 ms)	31 (1.1 ms)

example of $\boldsymbol{F}(x,y,z) = \begin{pmatrix} (x+1)(x^2+y^2+z^2-1) \\ (y+1)(x^2+y^2+z^2-1) \\ (z+1)(x^2+y^2+z^2-1) \end{pmatrix}$ can be viewed as $\mathcal{M}(x,y,z) = x^2+y^2+z^2-1$ and $\boldsymbol{G}(x,y,z) = \begin{pmatrix} x+1, y+1, z+1 \end{pmatrix}^T$. In this case, the Jacobian matrix $\nabla \boldsymbol{F}(\boldsymbol{x}) = \nabla \boldsymbol{G}(\boldsymbol{x})\mathcal{M}(\boldsymbol{x}) + \boldsymbol{G}(\boldsymbol{x})\nabla\mathcal{M}(\boldsymbol{x})^T$. For any point $\boldsymbol{x}^*$ on the manifold $\mathcal{M}(\boldsymbol{x}) = 0$, we have $\nabla \boldsymbol{F}(\boldsymbol{x}^*) = \boldsymbol{G}(\boldsymbol{x}^*)\nabla\mathcal{M}(\boldsymbol{x}^*)^T$ which is a rank-one matrix. Then by Theorem 2 with $r = 1$, the GD method is equivalent to Newton's method and has quadratic convergence.

4. Numerical examples

In this section, we compare GD and SGD methods with Newton's method on different systems of nonlinear equations with a stopping tolerance of 10^{-10}.

4.1. An example with the isolated solution

We consider a system of nonlinear equations

$$\boldsymbol{F}(x,y,z) = \begin{pmatrix} x^2+y^2+z^2-1, x+y+z, x-y^2 \end{pmatrix}^T = 0 \tag{23}$$

which has nonsingular isolated solutions. Table 1 shows a comparison between GD and Newton's methods with different initial guesses. Newton's method converges faster than the GD method in general but may diverge if the initial guess is not good. In terms of computing time, two methods are comparable although Newton's method has much fewer iterations.

4.2. A multi-dimensional example

We consider a large-scale system of nonlinear equations with the following formulation

$$\boldsymbol{F}_i(\boldsymbol{x}) = 2x_i - x_{i-1} - x_{i+1} + x_i^3 - 1, i = 1,\dots,n \text{ and } x_0 = x_{n+1} = 1 \tag{24}$$

which has an explicit solution, $(1,\dots,1)^T$. We choose the initial guess as $(2,\dots,2)^T$ and show the results of the GD, SGD ($s = n/2$ and $s = n/4$), and Newton's methods in Table 2. Newton's method converges with fewer iterations than the GD and SGD methods due to the quadratic convergence. In terms of computing time, the GD and SGD methods are comparable since Newton's method needs solving a linear system for each iteration. Since the SGD method with a batch size $s = n/2$ has a better performance, we use $s = n/2$ for the following numerical examples.

4.3. An example with quadratic convergence

We test quadratic convergence of the GD method by constructing the following example

$$\boldsymbol{F}(\boldsymbol{x}) = \frac{A}{2}\boldsymbol{x} + \frac{A}{4}\boldsymbol{x}^2 - \frac{3A}{4}\mathbf{1}, \tag{25}$$

Table 2
The number of iterations and computing time for three methods in (24) vs. n.

n	16	32	64	128	256	512	1024	2048
Newton	7 (5.3 ms)	7 (6.1 ms)	7 (12.3 ms)	7 (27.5 ms)	7 (0.12 s)	7(0.62 s)	7 (5.72 s)	7 (41.5 s)
GD	56 (5.5 ms)	54 (6.6 ms)	52 (8.9 ms)	51 (12.2 ms)	52 (0.03 s)	52 (0.12 s)	50 (0.41 s)	52 (2.01 s)
SGD (s = n/2)	118 (6.8 ms)	130 (7.9 ms)	140 (12.2 ms)	143 (0.02 s)	150 (0.02 s)	158 (0.07 s)	161 (0.56 s)	165 (2.11 s)
SGD (s = n/4)	233 (7.3 ms)	263 (10.1 ms)	275 (13.2 ms)	287 (0.03 s)	301 (0.03 s)	314 (0.08 s)	323 (0.50 s)	327 (2.24 s)

Table 3
The number of iterations and computing time for Newton, GD and SGD methods in Example 4.3 vs. n.

n	16	32	64	128	256	512	1024	2048
Newton	6 (2.9 ms)	6 (3.6 ms)	6 (7.6 ms)	6 (17.1 ms)	6 (65.4 ms)	6(0.41 s)	6 (3.85 s)	6 (31.6 s)
GD	6 (4.2 ms)	6 (1.6 ms)	6 (8.9 ms)	6 (4.4 ms)	6 (6.5 ms)	6 (14.9 ms)	6 (0.12 s)	6 (2.68 s)
SGD	20 (6.9 ms)	24 (4.3 ms)	22 (3.7 ms)	26 (17.5 s)	34 (63.8 ms)	33 (0.32 s)	37 (1.85 s)	35 (8.45 s)

Table 4
The number of iterations and computing time for three methods in Example 4.4 vs. n with different $f(\boldsymbol{x})$.

$f(\boldsymbol{x})$	n	16	32	64	128	256	512	1024	2048
$\boldsymbol{x}$	Newton	9 (7.3 ms)	10 (5.2 ms)	11 (10.6 ms)	12 (28.9 ms)	13 (0.15 s)	13(0.76 s)	14 (7.97 s)	15 (77.8 s)
	GD	9 (2.7 ms)	10 (1.4 ms)	11 (2.7 ms)	12 (2.7 ms)	13 (11.5 ms)	13 (69.2 ms)	14 (0.28 s)	15 (1.19 s)
	SGD	10 (2.3 ms)	11 (1.7 ms)	12 (1.4 ms)	13 (2.3 s)	14 (6.1 ms)	15 (52.2 ms)	16 (0.24 s)	17 (0.84 s)
$\boldsymbol{x}^3$	Newton	12 (3.2 ms)	14 (6.6 ms)	15 (14.4 ms)	17 (36.4 ms)	18 (0.22 s)	20(1.31 s)	21 (14.7 s)	23 (133.4 s)
	GD	12 (2.3 ms)	14 (2.1 ms)	15 (2.8 ms)	17 (5.2 ms)	18 (22.6 ms)	20 (0.12 s)	21 (0.46 s)	23 (1.91 s)
	SGD	16 (1.7 ms)	19 (1.9 ms)	22 (2.8 ms)	27 (6.7 s)	31 (23.3 ms)	34 (0.13 ms)	39 (0.51 s)	43 (2.22 s)
$\sin(\boldsymbol{x})$	Newton	9 (4.1 ms)	10 (3.8 ms)	11 (11.4 ms)	11 (29.2 ms)	12 (0.12 s)	13(0.87 s)	14 (8.34 s)	15 (79.4 s)
	GD	9 (2.1 ms)	10 (1.1 ms)	11 (3.1 ms)	11 (2.7 ms)	12 (9.3 ms)	13 (72.5 ms)	14 (0.31 s)	15 (1.28 s)
	SGD	9 (1.1 ms)	10 (1.1 ms)	11 (1.4 ms)	12 (2.5 s)	13 (6.7 ms)	14 (52.2 ms)	15 (0.18 s)	16 (0.76 s)

where $\mathbf{1}$ is the all-ones vector and the matrix $A = U\Sigma V^T \in \mathbb{R}^{n\times n}$. Here U and V are computed by the singular value decomposition of the tridiagonal matrix whose main diagonal element is 3 and upper/lower diagonal element is 1, and the diagonal elements of Σ are randomly chosen as σ and $-\sigma$. Obliviously, the analytical solution is $(1,\ldots,1)^T$. We choose the initial guess as $(2,\ldots,2)^T$ and show the results of the GD, SGD ($s = n/2$), and Newton's methods in Table 3. Both Newton's and GD methods converge to the solution in 6 iterations but the GD method is much faster than Newton's method in terms of computing time. Even the SGD method is faster than Newton's method although the number of iterations is larger due to stochasticity.

4.4. An example with positive dimensional solutions

We consider the following system of nonlinear equations: $\boldsymbol{F}(\boldsymbol{x}) = \quad f(\boldsymbol{x})(\boldsymbol{x}^T\boldsymbol{x} - 1) \quad = 0$, which has the positive dimensional solution set $\boldsymbol{x}^T\boldsymbol{x} = 1$. By choosing different $f(\boldsymbol{x})$ for different dimension, n, we show the results of three methods in Table 4. All the three methods converge to the solution with few iterations due to quadratic convergence. Moreover, the computing time of both the GD and SGD methods is much less than Newton's method.

4.5. An example of homotopy tracking

Since both the GD and SGD methods have the local convergence, we apply these methods to the homotopy tracking which can always guarantee the local neighborhood by controlling the homotopy

Table 5
Computing time (in the unit of seconds) of the homotopy tracking with three methods vs. n.

n	16	32	64	128	256	512	1024	2048
Newton	0.03	0.09	0.12	0.45	1.99	13.28	109.80	1090.61
GD	0.01	0.02	0.02	0.07	0.35	3.35	14.57	106.52
SGD (s = n/2)	0.11	0.14	0.23	0.78	4.12	40.96	220.34	1409.64

parameter. In particular, we consider the homotopy equation below:

$$\boldsymbol{F}(\boldsymbol{x},t)=(1-t)\begin{pmatrix} x_1^2+2x_1+3x_nx_2-1\\ \vdots\\ x_i^2+2x_i+3x_{i-1}x_{i+1}-1\\ \vdots\\ x_n^2+2x_n+3x_{n-1}x_1-1\end{pmatrix}+t\begin{pmatrix} x_1^2-1\\ \vdots\\ x_i^2-1\\ \vdots\\ x_n^2-1\end{pmatrix}=0, \tag{26}$$

where $i=2,\dots,n-1$ and t is the homotopy parameter. When $t=1$, we have an explicit solution $x_i=1$ $\forall i$. The target system $\boldsymbol{F}(\boldsymbol{x},0)$ is solved by tracking t from 1 to 0. We choose $\Delta t=-0.1$ and utilize three methods for solving the system of nonlinear equations for each t. All the comparisons among three methods are shown in Table 5: the GD method is faster than two other methods especially for large-scale systems while the SGD method is comparable with the Newton's method.

5. Conclusions

The GD method has been widely used in solving optimization problems. However, the stepsize is normally computed by the line-search algorithms and could be very slow when they are applied to large-scale systems of nonlinear equations. In this paper, we extend the GD method to solve systems of nonlinear equations and derive the explicit formula to compute the stepsize. With this new formulation, we prove the convergence of the GD method which has linear convergence in general. If the singular values of the Jacobian matrix are equal, the GD method is equivalent to Newton's method locally with the quadratic convergence. Several numerical examples are used to validate the convergence and to demonstrate the efficiency compared to Newton's method. Therefore, the GD method is very efficient especially in solving large-scale systems of nonlinear equations since it is an explicit scheme while Newton's method is an implicit scheme and needs to solve a linear system each iteration. Moreover, the SGD method, using only a part of the original system each iteration, is designed for large-scale problems whose evaluations are time-consuming. We treat the size of sub-systems as a hyperparameter depending upon the problem itself. In the future, we will further explore the SGD method to improve its efficiency.

Acknowledgments

This work is supported by the National Science Foundation (NSF) (Grant No. DMS-1818769).

References

[1] J. Dennis, R. Schnabel, Numerical Methods for Unconstrained Optimization and Nonlinear Equations, Vol. 16, SIAM, 1996.

[2] W. Hao, J. Hauenstein, B. Hu, A. Sommese, A bootstrapping approach for computing multiple solutions of differential equations, J. Comput. Appl. Math. 258 (2014) 181–190.

[3] K. Atkinson, A survey of numerical methods for solving nonlinear integral equations, J. Integral Equ. Appl. 4 (1) (1992) 15–46.

[4] W. Hao, J. Harlim, An equation-by-equation method for solving the multidimensional moment constrained maximum entropy problem, Commun. Appl. Math. Comput. Sci. 13 (2) (2018) 189–214.

[5] W. Hao, A homotopy method for parameter estimation of nonlinear differential equations with multiple optima, J. Sci. Comput. 74 (3) (2018) 1314–1324.

[6] C. Kelley, Iterative Methods for Optimization, SIAM, 1999.

[7] Å. Björck, Numerical Methods for Least Squares Problems, SIAM, 1996.

[8] Q. Chen, W. Hao, A homotopy training algorithm for fully connected neural networks, Proc. R. Soc. Lond. Ser. A Math. Phys. Eng. Sci. 475 (2231) (2019) 20190662.

[9] D. Bates, A. Sommese, J. Hauenstein, C. Wampler, Numerically Solving Polynomial Systems with Bertini, SIAM, 2013.

[10] B. Blaschke, A. Neubauer, O. Scherzer, On convergence rates for the iteratively regularized Gauss–Newton method, IMA J. Numer. Anal. 17 (3) (1997) 421–436.

[11] P. Deuflhard, Global inexact Newton methods for very large scale nonlinear problems, IMPACT Comput. Sci. Eng. 3 (4) (1991) 366–393.

[12] R. Dembo, S. Eisenstat, T. Steihaug, Inexact newton methods, SIAM J. Numer. Anal. 19 (2) (1982) 400–408.

[13] D. Knoll, D. Keyes, Jacobian-free Newton–Krylov methods: a survey of approaches and applications, J. Comput. Phys. 193 (2) (2004) 357–397.

[14] D. Coakley, P. Raftery, M. Keane, A review of methods to match building energy simulation models to measured data, Renew. Sustain. Energy Rev. 37 (2014) 123–141.

[15] S. Sra, S. Nowozin, S. Wright, Optimization for Machine Learning, MIT Press, 2012.

[16] L. Bottou, Large-scale machine learning with stochastic gradient descent, in: Proceedings of COMPSTAT'2010, Springer, 2010, pp. 177–186.

[17] D. Kingma, J. Ba, Adam: A method for stochastic optimization, 2014, arXiv preprint arXiv:1412.6980.

[18] Z. Griffin, J. Hauenstein, Real solutions to systems of polynomial equations and parameter continuation, Adv. Geom. 15 (2) (2015) 173–187.

[19] J. Nocedal, S. Wright, Numerical Optimization, Springer Science & Business Media, 2006.

[20] Y. Yuan, Step-sizes for the gradient method, AMS IP Stud. Adv. Math. 42 (2) (2008) 785.

[21] J. Barzilai, J. Borwein, Two-point step size gradient methods, IMA J. Numer. Anal. 8 (1) (1988) 141–148.

[22] H. Curry, The method of steepest descent for non-linear minimization problems, Quart. Appl. Math. 2 (3) (1944) 258–261.

[23] Y. Dai, Alternate step gradient method, Optimization 52 (4–5) (2003) 395–415.

[24] M. Raydan, B. Svaiter, Relaxed steepest descent and Cauchy–Barzilai–Borwein method, Comput. Optim. Appl. 21 (2) (2002) 155–167.

[25] L. Bottou, F. Curtis, J. Nocedal, Optimization methods for large-scale machine learning, SIAM Rev. 60 (2) (2018) 223–311.

[26] Q. Chen, W. Hao, A randomized Newton's method for solving differential equations based on the neural network discretization, 2019, arXiv:1912.03196.

[27] Z. Zeng, A Newton's iteration converges quadratically to nonisolated solutions too, 2019.

[28] H. Ninomiya, H. Asai, Orthogonalized steepest descent method for solving nonlinear equations, in: Proceedings of ISCAS'95-International Symposium on Circuits and Systems, Vol. 1, IEEE, 1995, pp. 740–743.