MATH6651 Research Project

"A conjugate gradient algorithm for large-scale unconstrained optimization problems and nonlinear equations"[1]
Gonglin Yuan and Wujie Hu, Journal of Inequalities and Applications, 2018

Sam Luik
Ioana Nedelcu
Colin Pierce

December 1st, 2020

1 Introduction

There are many mathematical models that rely on unconstrained optimization of smooth functions or systems of nonlinear equations and consequently there are many methods and algorithms to numerically solve these models. (In this case, the system is defined by a function with each component being a nonlinear equation). For the most part, these types of algorithms start with an initial point in the domain of the function, x_0 to initialize a sequence, x_k. Ideally, this sequence of values converges to the desired minimum/maximum/zero of the function. The sequence of values is determined by applying a line search technique to obtain the step size, α_k in a chosen search direction, d_k: $x_k = x_{k-1} + \alpha_k d_k$.

The report will largely be split into two sections, each dealing with either the problem of optimization or nonlinear equations and the application of a three-term search direction.

1.1 Optimization

The most common optimization methods are Newton and quasi-Newton (using the first and second derivative of the function as a search direction [2]), steepest descent and gradient (derivative-based search direction along with an optimal step size), trust region (optimizing within a region), and derivative-free methods [3]. However, even if these methods are suited for small- and medium-scale optimization models, they are not efficient for larger dimensions, ie a large number of values.

A more efficient conjugate gradient was then developed for solving linear systems of equations, $Ax = b$ or equivalently optimizing quadratic functions $\frac{1}{2}x^T Ax - x^T b$, (matrix A positive, semi-definite.) The method has increased efficiency; in particular, the convergence rate is proportional to the square root of the condition number of A (ratio of maximum to minimum eigenvalues), whereas the gradient descent convergence is proportional to the condition number itself. It uses an A-conjugate direction set, d_k along with an exact step size α_k that minimizes $f(x_k + \alpha_k d_k)$. It has been shown that the conjugate gradient method converges to the minimum value in n steps or yields an approximate solution in $\sqrt(n)$ steps.

Conjugate gradient methods can also be applied to nonlinear functions by instead using an inexact line search for the step size and a similar derivative-based direction set. One such large-scale algorithm uses a weak Wolfe-Powell (WWP) line search technique for the optimal step length and a Polak-Ribiere-Poliere (PRP) technique for the optimal direction [4].

The WWP line search algorithm for finding α_k is given by:

$$f(x_k + \alpha_k d_k) \leq f(x_k) + \phi \alpha_k g(x_k) d_k$$
$$(g(x_k + \alpha_k d_k), d_k) \geq \rho g(x_k) d_k)$$

with $\phi \in (0, 0.5), \rho \in (\phi, 1), g = \nabla f$ [5]. The first inequality is known as the Armijo rule for sufficient function descent.

The direction d_k is found by adding to the gradient an additional $\beta_k d_{k-1}$ term:

$$d_k = -g(x_k) + \beta_k d_{k-1}$$

where the PRP algorithm gives β_k by:

$$\beta_k = \frac{(g(x_k), (g(x_k) - g(x_{k-1}))}{\|g(x_{k-1})^2\|}$$

However, the PRP search direction does not have global convergence under the WWP line search algorithm. The Yuan-Wei-Li line search algorithm solves this by including an additional term to yield global convergence [6]. YWL line search will then be used with a modified three-term conjugate gradient search directions that guarantee a sufficient descent condition [7].

1.2 Nonlinear Equations

For large-scale nonlinear systems of equations, the same logic of a line search and similar three-term gradient search direction is applied to derive a sequence of values. However, an Armijo rule line search and three-term direction are used to generate a sequence of interim values w_k based on previous x_{k-1}. These then define a hyperplane $\{x|(h(w_k),(x–w_k))=0\}$ that separates the true roots of $h(x)$ from the current x_k value (details in section 3.2). The next actual x_{k+1} iteration value is found by projecting the current value x_k onto this hyperplane. This way the new point will be closer to the root by virtue of the projection. It has been shown that this method has global convergence for monotone, nonlinear functions and does not require the use of the function's gradient [8]. This new projected point is then used for the next iteration. Note that finding the root of a system of equations can be converted into an optimization problem of minimizing the Euclidean norm of the function defining the system.

In terms of the paper, the authors explored the use of both the YWL line search and projection methods and a modified three-term conjugate gradient search direction. This conjugate gradient algorithm has sufficient descent, trust region trait, global convergence and performs as well or better than similar existing algorithms for both optimization and systems of nonlinear equations.

2 Problem Description

Again, the method presented in the paper is meant to address the problem of large-scale and nonlinear optimization. As mentioned above, many existing methods can handle smaller-scale problems very well, such as the regular steepest descent or conjugate gradient methods. Other methods to solve large scale problems may not have global convergence (eg the WWP line search and PRP direction). The authors explore the use of the modified YWL line search and three-term gradient search direction.

The algorithm can be used both for large-scale unconstrained optimization problems and to find roots of nonlinear functions/ solve systems of nonlinear equations.

The two models addressed are the optimization of functions:

$$x^* \text{such that } f(x^*) = minf(x)$$

where $f(x)$ is a smooth (at least twice differentiable) linear or nonlinear, scalar-valued function on $\mathbb{R}^n$,

and the root finding of functions

$$x^* \text{such that } h(x^*) = 0$$

where $h(x)$ is a smooth, monotone $((h(x)-h(y))(x-y) > 0, \forall x, y)$, vector-valued function on $\mathbb{R}^n$. Note that this function can be thought of as the gradient of a scalar-valued function on $\mathbb{R}^n$, or a system of nonlinear equations, $h(x) = (f_1(x), f_2(x)...f_n(x))$.

3 Numerical Method

3.1 Optimization

The numerical method is a combination of the steepest descent method and conjugate gradient method, by using a modified backtracking line search algorithm to find the α_k step length in a three-term conjugate gradient search direction d_k

For the optimization algorithm, as usual we successively iterate to find α_k and d_k to obtain the next point of $x_k = x_{k-1}+\alpha_k d_k$, stopping when the accuracy condition is satisfied.

The paper uses the Yuan-Wei-Lu (YWL) search method, which incorporates an additional term to allow for global convergence [6], as the WWP algorithm doesn't necessarily converge globally to the minimum value.

The line search algorithm for finding α_k builds on the existing weak Wolfe Powell (WWP) algorithm: both the function and gradient are used as conditions to find the optimal step length. This is a backtracking line search, where an initial α is chosen and scaled down until it satisfies both conditions, as opposed to evaluating a formula in the conjugate gradient method (either the minimum of the interpolating polynomial of $minf(x_k + \alpha_k d_k)$ or the α formula for quadratic functions). See step a) in Algorithm below for YWL line search.

The search direction includes an additional term to the usual formula $d_{k+1} = -g_{k+1} + \beta_k d_k$, for some $\beta_k \in \mathbb{R}$. This modified three term conjugate gradient method includes appropriate scaling constants in order to guarantee sufficient descent and additional trust region properties. See step c) in Algorithm below for three-term conjugate gradient search direction.

Altogether, the algorithm is:

Algorithm 1: Unconstrained Optimization

Notation: $f_k = f(x_k), g_k = g(x_k) = \nabla(f_k), s_k = x_k - x_{k-1}$

Inputs:

$x_0, l \in (0, 0.5), l_1 \in (0, l), \eta_i > 0, (i = 1, 2, 3, 4, 5), \tau \in (l, 1), \epsilon \in (0, 1), e_1, e_2 \in (0, 1)$,

N iteration steps

Let $k = 0, d_0 = -g_0$

a) Find step length α_k such that both:

1. $f(x_k + \alpha_k d_k) \leq f(x_k) + l\alpha_k(g_k, d_k) + \alpha_k \min[-l_1(g_k, d_k), l\alpha_k \|d_k\|^2/2]$
2. $(g(x_k + \alpha_k d_k)d_k) \geq \tau((g_k), d_k) + \min[-l_1(g_k, d_k), l\alpha\|d_k\|^2/2]$

b) New step $x_{k+1} = x_k + \alpha_k d_k$

c) New search direction

$$d_{k+1} = -\eta_1 g_{k+1} + (1 - \eta_1)((d_k, g_{k+1})y_k^* - (g_{k+1}, y_k^*)d_k)/\delta$$

where:

$$\delta = \max(\min(\eta_5|(s_k, y_k^*)|, |(d_k, y_k^*)|), \eta_2\|y_k^*\|\|d_k\|, \eta_3\|g_k\|^2) + \eta_4\|d_k\|^2$$

$$y_k^* = g_{k+1} - \frac{\|g_{k+1}\|^2}{\|g_k\|^2} g_k$$

d) Stop conditions, either:

1. If $|f_k| > e_1, then \frac{|f_{k+1} - f_k|}{|f_k|} < e_2$
2. $\|g_{k+1}\| < \epsilon$
3. $k > N$

e) k = k + 1; repeat steps a) - e)

3.2 Nonlinear Equations

To find the root of a vector-valued function, a projection method is used to iterate through the sequence of values. An initial point is found as usual by $w_k = x_{k-1} + \alpha_k d_k$, with the three-term conjugate gradient direction d_k, and is then used to project to an improved point

[9]. The line search and search direction is similar to the ones used in the optimization algorithm. See step a) for the line search condition, b) for the new interim point, c) for the projection onto to the hyperplane and d) for the three-term direction in th Algorithm below.

Assuming $h(x)$ is monotone, we have $(h(w_k) - h(x*))(x * -w_k) \leq 0$ for the true root $x*$. With an appropriate α_k such that $h(w_k)(x_k - w_k) > 0$, this implies that the plane $\{x|(h(w_k), (x - w_k)) = 0\}$ separates the current x_k from the true roots. So the improved point, x_{k+1} is calculated as the projection of x_k onto this plane [8]. It has been shown that this projection method globally converges for monotone and Lipschitz continuous functions [3].

The algorithm is:

Algorithm 2: Nonlinear Root

Notation: $h_k = h(x_k), s_k = x_{k-1} - x_k$

Inputs: $x_0, \sigma > 0, s > 0, \rho \in (0, 1), \eta_i > 0, (i = 1, 2, 3, 4, 5), \epsilon \in (0, 1), e_1, e_2 \in (0, 1)$, N iteration steps

Let $k = 0, d_0 = -h_0$

a) Find step length α_k such that:

$$(-h(x_k + \alpha_k d_k), d_k) \geq \sigma \alpha_k \|h(x_k + \alpha_k d_k)\| \|d_k\|^2$$

where $\alpha_k = max(s, s\rho, s\rho^2 ...)$

b) New step $w_k = x_k + \alpha_k d_k$

c) If $\|h(w_k)\| \leq \epsilon$ then stop and set $x_{k+1} = w_k$; otherwise:

$$x_{k+1} = x_k - \frac{(h(w_k), (x_k - w_k))h(w_k)}{\|h(w_k)\|^2}$$

d) New search direction

$$d_{k+1} = -\eta_1 h_{k+1} + (1 - \eta_1)((d_k, h_{k+1})y_k^* - (h_{k+1}, y_k^*)d_k)/\delta$$

where

$$\delta = \max(\min(\eta_5 |(s_k, y_k^*), |(d_k, y_k^*)), \eta_2 \|y_k^*\| \|d_k\|, \eta_3 \|h_k\|^2) + \eta_4 \|d_k\|^2$$

$$y_k^* = h_{k+1} - \frac{\|h_{k+1}\|^2}{\|h_k\|^2} h_k$$

e) Stop if $\|h(x_{k+1})\| \leq \epsilon$

f) k = k + 1; repeat steps a) - f)

4 Theoretical Results

4.1 Optimization

The following proofs will demonstrate the sufficient descent, trust region and global convergence properties of the optimization algorithm.

Lemma 1: If the search direction is as defined in the algorithm, then (1) and (2) hold:
(1) $g_k^T d_k = -\eta_1 \|g_k\|^2$
Proof: Condition trivially holds for $k = 0$. For $k \geq 1$,

$$d_{k+1} = \left[-\eta_1 g_{k+1} + (1 - \eta_1)\left(d_k^T g_{k+1} y_k^* - g_{k+1}^T y_k^* d_k\right)/\delta\right]$$

therefore

$$\begin{aligned} g_{k+1}^T d_{k+1} =& g_{k+1}^T \big[-\eta_1 g_{k+1} + (1-\eta_1)\big(d_k^T g_{k+1} y_k^* - g_{k+1}^T y_k^* d_k\big)/\delta\big] \\ =& -\eta_1 \|g_{k+1}\|^2 + (1-\eta_1)\big(g_{k+1}^T d_k^T g_{k+1} y_k^* - g_{k+1}^T g_{k+1}^T y_k^* d_k\big)/\delta \\ =& -\eta_1 \|g_{k+1}\|^2 \end{aligned}$$

(2) $\|d_k\| \leq \big(\eta_1 + 2(1-\eta_1)/\eta_2\big)\|g_k\|$

Proof: by triangle inequality. Condition holds trivially for $k = 0$. For $k \geq 1$:

$$\begin{aligned} \|d_{k+1}\| =& \big\|-\eta_1 g_{k+1} + (1-\eta_1)\big(d_k^T g_{k+1} y_k^* - g_{k+1}^T y_k^* d_k\big)/\delta\big\| \\ \leq& \eta_1 \|g_{k+1}\| + 2(1-\eta_1)\|g_{k+1}\|\big\|y_k^*\big\|\|d_k\|/\delta \\ \leq& \eta_1 \|g_{k+1}\| + 2(1-\eta_1)\|g_{k+1}\|\big\|y_k^*\big\|\|d_k\|/\big(\eta_2\big\|y_k^*\big\|\|d_k\|\big) \\ =& \big(\eta_1 + 2(1-\eta_1)/\eta_2\big)\|g_{k+1}\|. \end{aligned}$$

This shows both the sufficient descent feature of the three-term search direction and trust region trait when optimizing.

The next theorem proves the global convergence of the resulting sequence of x values generated by the algorithm.

Theorem 1: If $f(x)$ has a continuous second derivative, has a lower bound, and $\nabla f(x) = g(x)$ is Lipschitz continuous, then $\lim_{k\to\infty} \|g_k\| = 0$.

Proof: by the algorithm formulas and the previous Lemma:

$$\begin{aligned} f(x_k + \alpha_k d_k) \leq& f_k + \iota\alpha_k g_k^T d_k + \alpha_k \min\big[-\iota_1 g_k^T d_k, \iota\alpha_k \|d_k\|^2/2\big] \\ \leq& f_k + \iota\alpha_k g_k^T d_k - \alpha_k \iota_1 g_k^T d_k \\ \leq& f_k + \alpha_k(\iota - \iota_1) g_k^T d_k \\ \leq& f_k - \eta_1 \alpha_k (\iota - \iota_1)\|g_k\|^2 \end{aligned}$$

therefore summing these values and by the Lipschitz continuity
$\sum_{k=0}^{\infty} \eta_1 \alpha_k (\iota - \iota_1)\|g_k\|^2 \leq f(x_0) - f_\infty < +\infty$
which implies that $\lim_{k\to\infty} \alpha_k \|g_k\|^2 = 0$

Similarly for the gradient condition:

$$\begin{aligned} g(x_k + \alpha_k d_k)^T d_k \geq& \tau g_k^T d_k + \min\big[-\iota_1 g_k^T d_k, \iota\alpha_k \|d_k\|^2\big] \\ \geq& \tau g_k^T d_k \end{aligned}$$

therefore

$$\begin{aligned} -\eta_1(\tau - 1)\|g_k\|^2 \leq& (\tau - 1) g_k^T d_k \\ \leq& \big[g(x_k + \alpha_k d_k) - g(x_k)\big]^T d_k \\ \leq& \big\|g(x_k + \alpha_k d_k) - g(x_k)\big\|\|d_k\| \\ \leq& \alpha_k \zeta \|d_k\|^2, \end{aligned}$$

because the gradient is Lipschitz continuous with ζ constant. Then:
$\alpha_k \geq \frac{(1-\tau)\eta_1\|g_k\|^2}{\zeta\|d_k\|^2} \geq \frac{(1-\tau)\eta_1\|g_k\|^2}{(\zeta(\eta_1+2(1-\eta_1)/\eta_2)^2\|g_k\|^2))} = \frac{(1-\tau)\eta_1}{(\zeta(\eta_1+2(1-\eta_1)/\eta_2)^2)}$
and so as needed
$\lim_{k\to\infty} \|g_k\|^2 = 0$

4.2 Nonlinear Root

For the nonlinear root algorithm, we will prove that similar properties hold for the line search with three-term search direction and projection technique. Assuming that there is a solution to $h(x) = 0$ and that $h(x)$ is Lipschitz continuous, then we have that for the

three-term search direction both

$$h_k^T d_k = -\eta_1 \|h_k\|^2$$

and

$$\|d_k\| \leq (\eta_1 + 2(1-\eta_1)/\eta_2 \|h_k\|$$

In other words, the sufficient descent and trust region traits still hold (proofs are similar to above). From this, we also have that $\|h_k\| \leq \theta, \theta > 0$

The next lemmas will be used to prove global convergence.

Lemma 2: Assuming the sequence x_k is produced by the algorithm, with true root x^* then:

$$\left\|x_{k+1} - x^*\right\|^2 \leq \left\|x_k - x^*\right\|^2 - \|x_{k+1} - x_k\|^2$$

And the sequence x_k either attains the solution in a finite number of steps, or it is infinite and

$$\sum_{k=0}^{\infty} \|x_{k+1} - x_k\|^2 < \infty.$$

A proof has been provided in [8] based on the properties of the hyperplance and projection operators.

Lemma 3: The algorithm generates an iteration point in a finite number of α_k steps Proof: denote $\Psi = N \cup \{0\}$. We assume that $\|h_k\| \geq \varepsilon_*, k \in \Psi$ (ie proof by contradiction).

Suppose there is k such that $-h(x_k + \alpha_k d_k)^T d_k < \sigma\alpha_k \left\|h(x_k + \alpha_k d_k)\right\| \|d_k\|^2$,

Then by the sufficient descent trait ($h_k^T d_k = -\eta_1 \|h_k\| \|h_k\|$) so

$$\begin{aligned}
\|h_k\|^2 =& -\eta_1 h_k^T d_k \\
=& \eta_1 \left[\left(h\left(x_k + \alpha_k^{(l)} d_{k^*}\right) - h(x_k)\right)^T d_k - \left(h\left(x_k + \alpha_k^{(l)} d_k\right)^T d_k\right)\right] \\
<& \eta_1 \left[L + \sigma \left\|h\left(x_k + \alpha_k^{(l)} d_k\right)\right\|\right] \alpha_k^{(l)} \|d_k\|^2, \quad \forall l \in \Psi.
\end{aligned}$$

By the properties from before, $\|d_k\| \leq \left(\eta_1 + 2(1-\eta_1)/\eta_2\right)\|h_k\|$ and $h_k^T d_k = -\eta_1 \|h_k\| \|h_k\|$ then for Lipschitz constant L,

$$\begin{aligned}
\left\|h\left(x_k + \alpha_k^{(l)} d_{k^*}\right)\right\| \leq& \left\|h\left(x_k + \alpha_k^{(l)} d_k\right) - h(x_k)\right\| + \left\|h(x_k)\right\| \\
\leq& L\alpha_k^{(l)} \|d_k\| + \theta \\
\leq& Ls\theta\left(\eta_1 + 2(1-\eta_1)/\eta_2\right) + \theta
\end{aligned}$$

therefore

$$\begin{aligned}
\alpha_k^{(l)} >& \frac{\|h_k\|^2}{\eta_1 [L + \sigma \|h(x_k + \alpha_k^{(l)} d_{k^*})\|] \|d_k\|^2} \\
>& \frac{\|h_k\|^2}{\eta_1 [L + \sigma(Ls\theta(\eta_1 + 2(1-\eta_1)/\eta_2) + \theta)] \|d_k\|^2} \\
>& \frac{\eta_2^2}{\eta_1 [L + \sigma(Ls\theta(\eta_1 + 2/\eta_3) + \theta)](2(1-\eta_1) + \eta_2\eta_1)^2}, \quad \forall l \in \Psi.
\end{aligned}$$

Clearly this is not the definition of α_k as in the algorithm, which yields a contradiction. Therefore the line search in the algorithm generates a α_k in a finite number of steps. The next theorem is again for global convergence.

Theorem 2: Assuming that there is a solution to $h(x) = 0$ and that $h(x)$ is Lipschitz continuous then

$$\liminf_{k \to \infty} \|h_k\| = 0.$$

Proof: Again a proof by contradiction. Assume there is $\varepsilon_0 > 0$ and k_0 such that $\|h_k\| \geq \varepsilon_0, \quad \forall k \geq k_0$.
Then by the assumption $\|h_k\| \leq \theta$ and property $\|d_k\| \leq \big(\eta_1 + 2(1-\eta_1)/\eta_2\big)\|h_k\|$:

$$\|d_k\| \leq \big(\eta_1 + 2(1-\eta_1)/\eta_2\big)\|h_k\| \leq \big(\eta_1 + 2(1-\eta_1)/\eta_2\big)\theta, \quad \forall k \in \Psi.$$

and since $h_k^T d_k = -\eta_1 \|h_k\| \|h_k\|$ then $\|d_k\| \geq \eta_1 \|h_k\| \geq \eta_1 \theta$.
From this, the sequence $\{d_k\}$ is bounded so there is the point d^* and integer set N_1 such that $\lim\limits_{k\to\infty} d_k = d^*, \quad k \in N_1$.

And we know by Lemma 2 the sequence x_k is bounded as well and there is point x^* and set $N_2 \subset N_1$ such that

$$\lim_{k\to\infty} x_k = x^*, \quad \forall k \in N_2.$$

By Lemmas 2 and 3 above, $\alpha_k \|d_k\| \to 0, \quad k \to \infty$.
Since $\{d_k\}$ is bounded then $\lim\limits_{k\to\infty} \alpha_k = 0$.
By the formula for α_k and $\alpha_k^* = \alpha_k/\rho$ defined in the algorithm:

$$-h\big(x_k + \alpha_k^* d_k\big)^T d_k \leq \sigma \alpha_k^* \big\| h\big(x_k + \alpha_k^* d_k\big)\big\| \|d_k\|^2,$$

taking the limits from both sides of the above formula and sufficient descent property yields

$$h\big(x^*\big)^T d^* > 0$$

and

$$h\big(x^*\big)^T d^* \leq 0.$$

This is a contradiction, so the algorithm does produce a globally convergent sequence of values for the nonlinear root.

Because of this theorem and the fact that $d_{k+1} = d_k$ when the gradient is zero, the limit of the series is zero.

5 Numerical Results

5.1 Optimization

The paper compares the numerical results of Algorithm 1, with two other algorithms using existing different search directions [7] [10] and the three-term search direction:

$$d_{k+1} = -g_{k+1} + \frac{((g_{k+1}, y_k)d_k - (d_k, g_{k+1}y_k)}{(g_{k+1}, g_k)}$$

$$d_{k+1} = -y_k + \frac{(y_k, y_k)}{(y_k, d_k)} d_k \frac{(y_{k-1}, y_k)}{(y_{k-1}, d_{k-1})} d_{k-1}$$

for $y_k = g_{k+1} - g_k$
The iteration number, total number of function and gradient evaluations, and the CPU calculation time are compared for a standard list of unconstrained optimization problems of dimensions 1200, 3000, 6000 and 9000 (including functions such as extended and perturbed exponential, trigonometric and quadratic functions) [11].

The performance profiles for dimension 9000 (cumulative distributions of performance ratios - performance of the given algorithm relative to the best algorithm) [12] show that

the proposed algorithm (red) requires fewer iterations, calculations and less time than the other two (the graphs show a steeper and higher line, indicating it is efficient in all metrics as per the performance profile benchmarking method). This shows that the proposed algorithm using the YWL line search and three term search direction is better suited than the other algorithms. (top left: iteration number, bottom left: number of evaluations, right: calculation time).

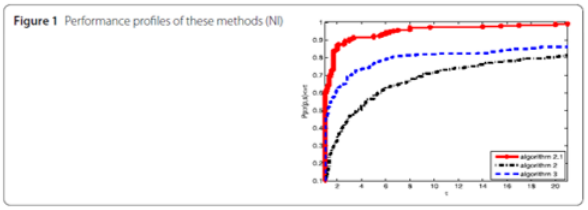

Figure 1 Performance profiles of these methods (NI)

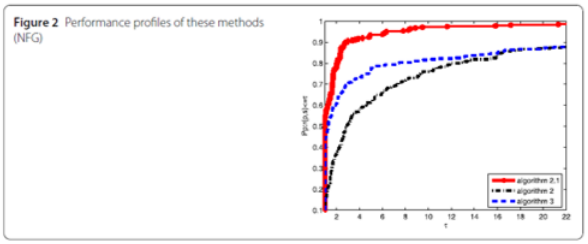

Figure 2 Performance profiles of these methods (NFG)

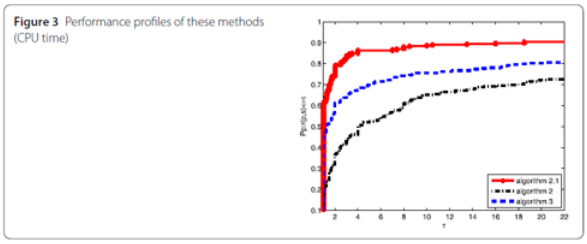

Figure 3 Performance profiles of these methods (CPU time)

5.2 Nonlinear Equations

A similar approach is used for the proposed nonlinear Algorithm 2 by comparing it with another algorithm that uses the first alternative search direction above. Functions tested include exponential, trigonometric and singular functions of dimensions 3000, 6000, 9000. The same metrics are compared and the resulting graphs show that the method is as efficient in terms of calculations and CPU time, and takes slightly fewer iterations. Overall, this indicates that the proposed method is well-suited for large-scale nonlinear root-finding problems. (top left: number of evaluations, bottom left: calculation time, right: iteration number).

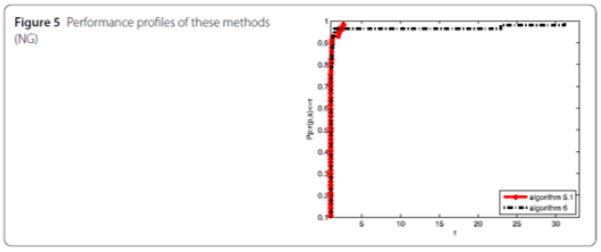

Figure 5 Performance profiles of these methods (NG)

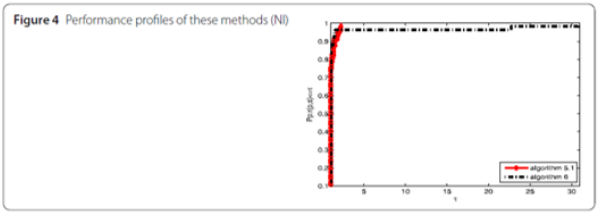

Figure 4 Performance profiles of these methods (NI)

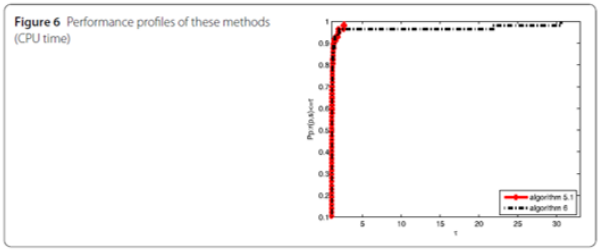

Figure 6 Performance profiles of these methods (CPU time)

6 Applications and Numerical Comparisons

6.1 Neural Networks

Neural networks are trained by optimizing a function that tells them how well they were able to identify an input. So when they are trying to recognize a input in the future, they can use the objective function to determine which input it is. The variables in these equations could easily be in the thousands. An example would be taking a picture and trying to match each row of pixels in the image to a pattern using the cost function. A photo taken on a Samsung Galaxy A5 is 4608x3456, which means that such pictures would require solving a 3456 term optimization problem to train the neural network for a single picture. And since thousands of photos are required to train a neural network, there needs to be an efficient way of doing so. Clearly, the proposed algorithms are well-suited to solve these large scale optimization problems.
We have chosen as an example a small-scale theoretical objective function that could be used to train the neural network to recognize patterns in images that are 10 pixels high. The optimization problem finds the optimal slope β_1 and intercept β_0 given the following 10 data points, by minimizing the least squares function $f(x)$ or finding the root of its gradient function $h(x)$. Clearly the data is indicative of a straight line through the origin, and the resulting values support this claim.

$$(x, y) = \{(0.88, 1), (2, 2.03), (3, 3.17), (4, 4.01), (5, 4.95), (6, 6.12), (7, 6.99), (8, 8.05), (9, 8.99), (10, 10.05)\}$$

$$f(\beta_0, \beta_1) = \sum_{i=1}^{10}(y_i - \beta_0 - \beta_1 x_i)$$
$$h(\beta_0, \beta_1) = [20\beta_0 + 110\beta_1 - 2762/25, 110\beta_0 + 770\beta_1 - 3867/5]$$

Using a tolerance of 10^{-3}, initial value of $(0, 0)$ and constants as given in the paper, the optimization algorithm for f yields a slope (0.9982) and intercept (0.0984) in 4 iterations. Similarly, the root for h is found to be at slope (0.9919) and intercept (0.0769) in 38 steps.

6.2 Numerical Comparisons

The algorithms presented thus far are naturally comparable to a number of in-class methods. For example, we can compare the first algorithm with Conjugate Gradient descent methods for linear and non-linear functions. Consider first, the complexity of both Algorithm 1 and the (Pre-conditioned) Conjugate Gradient descent. It is known from class that for the latter method, iteration number, and therefore calculation time, is proportional to dimension. By measuring the convergence properties in increasing-scale equations, we can find a similar result below for time to convergence in Algorithm 1 as in the PCG method.

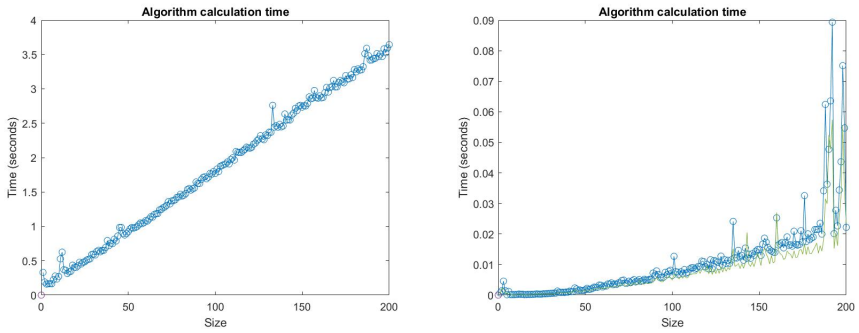

Figure: Algorithm 1 vs PCG and CG

We can also look at the optimization of non-linear functions by comparing Algorithm 1 to the non-linear conjugate gradient method (using an interpolation polynomial for the exact step size, and Fletcher-Reeves search direction formula). For the following conic non-linear functions:

$$g_1(x,y) = (x-5)^2 + (y-2)^2 - 3$$
$$g_2(x,y) = -cos(xy) + x^2 + y^2$$
$$g_3(x,y) = 3(x-5)^2 + 2(y-2)^2 - 3$$

and initial guess $x_0 = [5,5]$ we obtain the following convergence properties:

	Alg 1 Time (s)	Non-linear CG Time(s)	Alg 1 Iter	Non-linear CG Iter
g_1	0.0247	0.0235	2	2
g_2	0.1058	0.1161	10	3
g_3	0.0222	0.0219	2	2

For the case of non-linear functions, it seems that Algorithm 1 is comparable to the nonlinear conjugate gradient method in both calculation time and number of iterations. This result is somewhat surprising since the algorithm has a larger number of evaluations/calculations at each step. Also the step size is determined through an inexact backtracking line search, as opposed to an exact formulaic value for conjugate gradient, which could suggest lower efficiency (perhaps such is the case for function g_2.) However, even if the algorithm takes more steps, its calculation time is in line with the conjugate gradient method. Clearly the algorithm performs as well as the nonlinear conjugate gradient method even for small-scale functions.

Consider now the following systems of equations:

$$f_1(x,y) = x^2 - y^2 \ \textbf{(1)}$$
$$f_2(x,y) = 3xy^2 - x^3 - 1$$

and,

$$f_1(x,y) = 3x - 7y^2 + 5 \ \textbf{(2)}$$
$$f_2(x,y) = -23x + 19y - 10$$

Solutions for these systems of equations can be solved for using Newton's method, which is a similar gradient descent method. Such solutions are equivalent to finding the root of a function $h(x,y) = (f_1, f_2)$. Comparing Newton's method and Algorithm 2 with initial condition $x_0 = [0.1, 0.1]$, we obtain:

	Alg 2 Time (s)	Newton's Method Time(s)	Alg 2 Iter	Newton's Method Iter
(1)	0.1083	0.3920	12	14
(2)	0.1594	0.4231	122	8

The result is interesting. Algorithm 2 is faster than Newton's method, which is a good measure for performance. The most likely cause for the Newton inefficiency is the inverse matrix and multiplication calculations using the Jacobian/gradient at each step. Also, the large number of steps for the second system may be attributed to the inexact line search for the step size. Overall, this suggests that as we continue to scale our problem larger, eventually, the proposed algorithm will be able to outperform other common numerical methods by a meaningful amount to justify its implementation (supporting the fundamental claim of Yuan and Hu). As such, we now have a demonstrated tool to numerically optimize increasingly large-scale problems with increasingly difficult solutions.

References

[1] Gonglin Yuan and Wujie Hu. "A conjugate gradient algorithm for large-scale unconstrained optimization problems and nonlinear equations". In: *Journal of Inequalities and Applications* (2018). DOI: https://doi.org/10.1186/s13660-018-1703-1.

[2] Mehiddin Al-Baali, Emilio Spedicato, and Francesca Maggioni. "Broyden's quasi-Newton methods for a nonlinear system of equations and unconstrained optimization: a review and open problems". In: *Optimization Methods and Software* 29.5 (2014), pp. 937–954. DOI: 10.1080/10556788.2013.856909.

[3] Q. Li and D. Li. "A class of derivative-free methods for large-scale nonlinear monotone equations". In: *IMA Journal of Numerical Analysis* 31.4 (2011), pp. 1625–1635. DOI: 10.1093/imanum/drq015.

[4] R. Fletcher and C. M. Reeves. "Function minimization by conjugate gradients". In: *The Computer Journal* 7.2 (1964), pp. 149–154. DOI: 10.1093/comjnl/7.2.149.

[5] Steven Wright and Jorge Nocedal. *Line Search Methods. In: Numerical Optimization.* Springer Series in Operations Research and Financial Engineering. Springer, New York, NY, 2006.

[6] Yuan G. Wei and Z. Lu X. "Global convergence of BFGS and PRP methods under a modified weak Wolfe–Powell line search". In: *Journal of Applied Mathematical Modelling* 47 (2017), pp. 811–825.

[7] L. Zhang, Zhou, and Li D. W. "A descent modified Polak–Ribière–Polyak conjugate gradient method and its global convergence". In: *Journal of Numerical Analysis* 47 (2006), pp. 629–640.

[8] Michael V Solodov and Benav F Svaiter. "Reformulation: Nonsmooth, Piecewise Smooth, Semismooth and Smoothing Methods". In: (1998), pp. 355–369.

[9] Gonglin Yuan and Maojun Zhang. "A three-terms Polak–Ribière–Polyak conjugate gradient algorithm for large-scale nonlinear equations". In: *Journal of Computational and Applied Mathematics* 286 (2015), pp. 186–195. DOI: https://doi.org/10.1016/j.cam.2015.03.014.

[10] L. Nazareth. "A conjugate direction algorithm without line searches". In: *Journal of Optimization Theory Applications* 23 (1997), pp. 373–387.

[11] Andrei Neculai. "An unconstrained optimization test functions collection". In: *Advanced Modelling Optimization* 10 (2008).

[12] E. Dolan and J. Moré. "Benchmarking optimization software with performance profiles". In: *Mathematical Programming* 91 (2002), pp. 201–213. DOI: https://doi.org/10.1007/s101070100263.