# A Scaled Gradient Descent Method for Unconstrained Optimiziation Problems With A Priori Estimation of the Minimum Value

# A SCALED GRADIENT DESCENT METHOD FOR UNCONSTRAINED OPTIMIZIATION PROBLEMS WITH A PRIORI ESTIMATION OF THE MINIMUM VALUE

BY

CURTIS D'ALVES, B.Sc.

A THESIS

SUBMITTED TO THE DEPARTMENT OF COMPUTER SCIENCE

AND THE SCHOOL OF GRADUATE STUDIES

OF MCMASTER UNIVERSITY

IN PARTIAL FULFILMENT OF THE REQUIREMENTS

FOR THE DEGREE OF

MASTER OF SCIENCE

Master of Science (2016) McMaster University

(Computer Science) Hamilton, Ontario, Canada

| | |
|---|---|
| TITLE: | A Scaled Gradient Descent Method for Unconstrained Optimiziation Problems With A Priori Estimation of the Minimum Value |
| AUTHOR: | Curtis D'Alves<br>B.Sc. (Computer Science)<br>McMaster University, Hamilton, Canada |
| SUPERVISOR: | Dr. Christopher Anand |
| NUMBER OF PAGES: | xv, 40 |

*For the glory of science!*

# Abstract

This research proposes a novel method of improving the Gradient Descent method in an effort to be competitive with applications of the conjugate gradient method while reducing computation per iteration. Iterative methods for unconstrained optimization have found widespread application in digital signal processing applications for large inverse problems, such as the use of conjugate gradient for parallel image reconstruction in MR Imaging. In these problems, very good estimates of the minimum value at the objective function can be obtained by estimating the noise variance in the signal, or using additional measurements.

The method proposed uses an estimation of the minimum to develop a scaling for Gradient Descent at each iteration, thus avoiding the necessity of a computationally extensive line search. A sufficient condition for convergence and proof are provided for the method, as well as an analysis of convergence rates for varying conditioned problems. The method is compared against the gradient descent and conjugate gradient methods.

A method with a computationally inexpensive scaling factor is achieved that converges linearly for well-conditioned problems. The method is tested with tricky non-linear problems against gradient descent, but proves unsuccessful without augmenting with a line search. However with line search augmentation the method still outperforms

gradient descent in iterations. The method is also benchmarked against conjugate gradient for linear problems, where it achieves similar convergence for well-conditioned problems even without augmenting with a line search.

# Acknowledgements

I would like to express my sincere gratitude to my supervisor Dr. Christopher Anand who helped me through this entire process. He has guided me not just through my graduate career but also undergraduate career at McMaster. He has been as good a supervisor and mentor as one could ask for.

I would also like to thank my fellow graduate student and friend Al-hassy Musa for taking the time to proof-read and criticize this thesis into shape at the final stretch.

# Notation and abbreviations

- CG - Conjguate Gradient
- WLOG - Without Loss Of Generality
- MR Imaging - Magnetic Resonance Imaging
- Q.E.D - quod erat demonstrandum (which is what had to be shown)
- BFGS - Broyden Fletcher Goldfarb Shanno algorithm

# Contents

# List of Tables

# List of Figures

# Chapter 1

# Introduction

## 1.1 Introduction

Well-known non-linear, unconstrained continuous optimization methods such as Quasi-Newton methods focus on minimizing a function $f(x) : \mathbb{R}^n \to \mathbb{R}$ assuming no knowledge of the value of the minimum value of $f(x)$. The goal of an optimization method is to locate a point $x_{\min}$ at which the minimum occurs and the minimum value $f(x_{\min})$. Certain problem contexts, however, provide a simple means for obtaining the minimum value, and require a method only to find the point $x_{\min}$ at which this value occurs. For example, in inverse imaging problems problems, such as MR Image Reconstruction, the minimum value is determined by the standard deviation of the normally distributed measurement noise, which can be estimated without reconstructing the image. Iterative methods like conjugate gradient (CG) have become popular for large, sparse problems like parallel image reconstruction [2], and as a consequence many solutions for removing/reducing the line search have been developed [5].

This thesis is an experimental approach for creating a scaled gradient descent method

computing the point $x_{\min}$ minimizing a function $f(x) : \mathbb{R}^n \to \mathbb{R}$ using a priori knowledge of the minimum value.

## 1.2 Gradient Descent

The gradient descent method, also known as the method of steepest descent, is an iterative method for unconstrained optimization that takes an initial point $x_0$ and attempts to sequence converging to the minimum of a function $f(x)$ by moving in the direction of the negative gradient $(-\nabla f(x))$. In order to find a true minimum, the method requires a sufficiently smooth, convex function $f(x)$ [6]. The step size (how far to move along the direction of the gradient at each iteration) is usually decided by a line search method, which seeks a sufficient decrease at each iteration for better convergence [4]. The line search uses extra computation, and versions of descent methods without the line search have been created to avoid the line search while still converging, given proper estimates of certain parameters at each iteration [7].

Other sources of inefficiencies with gradient descent include a "zig-zagging" pattern that is created on the path to the minimum due to orthogonality of gradients picked in successive iterations [6]. For this reason the conjugate gradient method was developed.

## 1.3 Conjugate Gradient

For this thesis' purpose, we will compare our novel algorithm to the linear CG method, which seeks a solution to the linear system $Ax = b$ by minimizing the quadratic form

$$f(x) = \frac{1}{2}x^T Ax - b^T x + c. \tag{1.1}$$

The function (1.1) is known to be convex if the matrix $A$ is positive definite. If the matrix is *symmetric positive definite*, one can calculate the gradient of (1.1) to be

$$\nabla f(x) = Ax - b. \tag{1.2}$$

The negative of the gradient is thus the *residual* of the system at each iteration

$$-\nabla f(x_i) = r_i = b - Ax \tag{1.3}$$

We can then use the method of Gradient Descent to solve the system like so

$$\begin{aligned} r_i &= b - Ax_i, \\ \alpha_i &= \frac{r_i^T r_i}{r_i^T A r_i}, \\ x_{i+1} &= x_i + \alpha_i r_i \end{aligned} \tag{1.4}$$

Here $\alpha_i$ serves as a simple line search selected by our knowledge of the quadratic form we are minimizing. As mentioned, the problem with this approach is the orthogonality of the gradients at each iteration that creates a "zig-zagging" pattern making it difficult to navigate valley's along the paraboloid. The CG method adds extra steps

to avoid this, increasing the computation at each iteration but vastly improving the total number of iterations involved in convergence. All together the method of CG can be summarized by iterating the following steps

$$\begin{aligned} d_0 = r_0 &= b - Ax_0, \\ \alpha_i &= \frac{r_i^T r_i}{d_i^T A d_i} \\ x_{i+1} &= x_i + \alpha_i d_i, \\ r_{i+1} &= r_i - \alpha_i A d_i, \\ B_{i+1} &= \frac{r_{i+1}^T r_{i+1}}{r_i^T r_i}, \\ d_{i+1} &= r_{i+1} + B_{i+1} d_i \end{aligned} \tag{1.5}$$

## 1.4 Line Search

A line search is an iterative method for computing the scale factor $\alpha$ at each iteration. More formally, the line search solves the problem:

$$\alpha = \underset{\alpha_i}{\operatorname{argmin}} \qquad f(x_i + \alpha_i d_i) \tag{1.6}$$

All approaches of a line search involve iterating until at least the first condition of the strong Wolfe-conditions are fulfilled as stated in [1]. The strong Wolfe-conditions are stated as

$$f(x_k + \alpha_i d_k) \leq f(x_k) + c_1 \alpha_i (d_k \cdot \nabla f(x_k)) \tag{1.7}$$

$$|\nabla f(x_k + \alpha_i d_k) \cdot d_k| \leq c_2 \nabla f(x_k) \cdot d_k \tag{1.8}$$

where $0 < c_1 < c_2 < 1$ are often imposed on the line search at different values regarding precision. $c_1$ is often chosen to be quite small (i.e $10e^{-6}$), while $c_2$ is chosen to be closer to 1.

The first rule (1.8) is known as the *Armijo rule* and is considered the least qualifying condition for a "good" step-size. It requires computing $f(x_k)$ and the gradient $\nabla f(x_k)$ at each iteration of the search. Hence line searches can be particularly expensive when computation of the gradient $\nabla f(x_k)$ is expensive.

## 1.5 Motivation

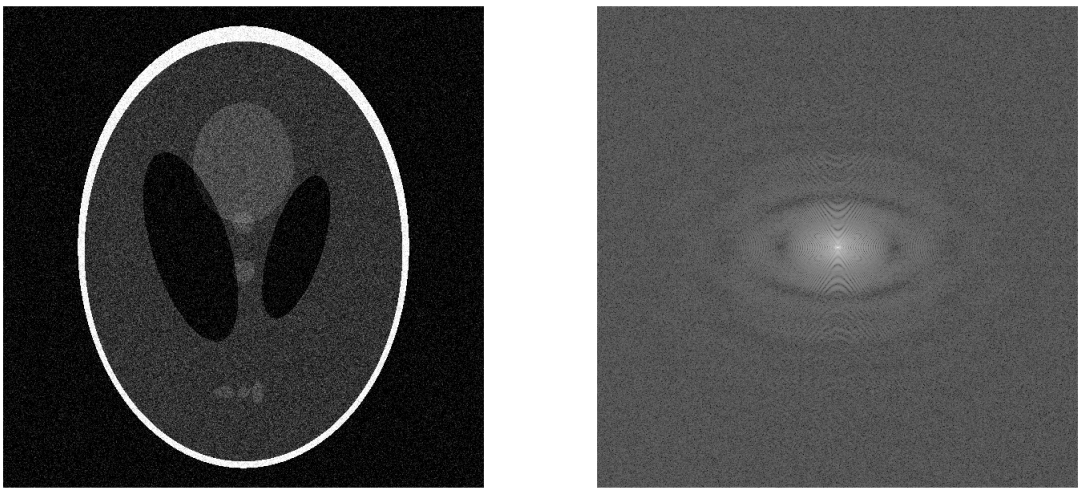
For our motivation we gave an example of an unconstrained optimization problem commonly solved by CG for which a reasonable estimate of the minimum can be precomputed. In parallel MR Imaging, specifically the SENSE problem, the solution is found by solving the least squares problem

$$\min_\rho \sum_{c \in \{0,1,\ldots,C-1\}} \|M_c - \pi F T(S_c \rho)\|^2 \tag{1.9}$$

where $\rho$ is the reconstructed 2d image or 3d volume we attempt to solve for and $M_c$ are measurements taken in *k-space* (the image transformed into the frequency domain) collected by a given coil $c$ over the total number of coils $C$. The image is

found by relating it to the measurements collected by multiplying pointwise by the coil sensitivities ($S_c$) and taking the Fourier transform ($FT$), then finally performing a projection $\pi$ to project to the subset of measurements covered by $M_c$ (each coil takes a subset of measurements).

The difference between the measurements and the true signal is the noise. The actual difference can be smaller in the case of over-fitting, but with millions or billions of data points, this effect will not be significant, and we can assume that the minimum of the least squares problem is the sum of the squared noise in the measurements taken. This can often be estimated by calculating the sample standard deviation of the data at the outer edges of k-space where high-resolution information is held, and where the signal is much smaller than the noise, and hence can be replaced by zero.



(a) Shepp-Logan Phantom (b) K-Space

Figure 1.1: Shepp-Logan phantom with Gaussian noise and K-Space

Figure 1.1 shows the Shepp-Logan phantom with added Gaussian noise and it's corresponding k-space. Notice the center of k-space contains all the important, low-resolution information, whereas the outer edges are nothing but noise. Thus to solve for the noise variance using the outer dimensions of k-space we compute:

$$\sigma = \sqrt{\frac{1}{Cn_0} \sum_{c \in \{0,1,\ldots,C-1\}} \sum_{i \in \{0,..,n_0-1\}} m_{(c,i)}^2} \tag{1.10}$$

where $n$ is the total size of the measurements taken (size of $M$), $m$ is the data in the outer row of $M$, $n_0$ is the size of $m$, and $\sigma$ is the computed standard deviation per pixel. This gives the estimate (number of measurements)$\times\sigma$ for the minimum value.

## 1.6 Overview

This thesis introduces two methods that attempt to improve upon gradient descent and CG. Derivations of two methods, referred to simply as Method 1 and 2, are given and a convergence proof of Method 1 is supplied. Analysis of the performance of both methods against gradient descent and conjugate gradient is done by collecting statistics on the number of iterations required to solve commonly used benchmark problems for unconstrained optimization and randomly generated linear problems for comparison with CG.

A method with an efficiently computed scale factor is given, that converges linearly for well-conditioned problems. The method is tested with tricky non-linear problems against gradient descent, but proves unsuccessful without augmenting with a line search. However with line search augmentation the method still outperforms gradient descent in iterations. The method is also benchmarked against conjugate gradient

for quadratic problems, where it achieves similar convergence for well-conditioned problems even without augmenting with a line search.

# Chapter 2

# Method 1

# A Scaled Gradient Descent Method

In this chapter we introduce a method that attempts to improve gradient descent by constructing a means to determine a scale factor based using knowledge of the minimum value. Thus avoiding the use of a line search, in the hopes of reducing computation per iteration. We start by giving an explanation of the derivation of the algorithm, which is really just a derivation of the scale factor. The chapter finishes by presenting the resulting algorithm in mathematical notation as an iteration from $x_k$ to $x_{k+1}$.

## 2.1 Algorithm Derivation

The algorithm is constructed by reformulating Steepest Descent to take advantage of knowledge of the function value at the minimum, which provides an estimate of the step length in the direction of the usual negative gradient direction. The

algorithm seeks a minimum by fitting the slope (magnitude of the gradient, $\nabla f(x_k)$) and function value, $f(x_k)$, at the current guess plus the known value at the minimum to a paraboloid:

$$\Phi(d) = f(x_k) + \nabla f(x_k)d + ad^T d \tag{2.1}$$

Note that this is a round, convex paraboloid — since its Hessian is the identity matrix scaled by $a$ — whose minimum from the current point lies in the direction of the negative gradient. We know the minimum of $d$ occurs at ca

$$\text{let} \quad \nabla\Phi(d) = 0 \qquad \text{then} \qquad d = -\frac{\nabla f(x_k)}{2a} \tag{2.2}$$

We know $d$ is in the direction of the negative gradient, we seek to solve for the scale factor based on $a$. Substituting (2.2) into (2.1) yields the minimum — of the approximating parabola. Using this knowledge we solve for $a$

$$\begin{aligned} \min = f(x_k) - \frac{1}{2a}\|\nabla f(x_k)\|^2 + \frac{a}{4a^2}\|\nabla f(x_k)\|^2 \quad &\iff \\ \min - f(x_k) = -\frac{2}{4a}\|\nabla f(x_k)\|^2 + \frac{1}{4a}\|\nabla f(x_k)\|^2 \quad &\iff \\ \min - f(x_k) = -\frac{1}{4a}\|\nabla f(x_k)\|^2 & \end{aligned}$$

resulting in

$$a = -\frac{1}{4}\frac{\|\nabla f(x_k)\|^2}{\min - f(x_k)}. \tag{2.3}$$

By substituting (2.3) back into (2.2), we know use this to solve for $d$ to obtain

$$d_k = 2\frac{\min - f(x_k)}{\|\nabla f(x_k)\|^2}\nabla f(x_k), \tag{2.4}$$

which serves as a Search Direction for an Iterative method of the form

$$x_{k+1} = x_k + \alpha\nabla f(x_k), \tag{2.5}$$

where

$$\alpha = 2\frac{\min - f(x_k)}{\|\nabla f(x_k)\|^2} \leq 0 \tag{2.6}$$

is a scaling factor, giving rise to the algorithm name. Note: It is obvious that (2.6) is non-positive.

For an initial guess $x_0$, a minimum within precision $\epsilon$ can be obtained by iterating (2.5) on $x_0$ until $|\min - f(x_k)| \leq \epsilon$. The scaling factor (2.6) is always negative, hence the method always moves in the negative direction of the gradient just like Gradient Descent.

## 2.2 Resulting Method — Method 1

Altogether, the resulting method is defined recursively as

$$\begin{aligned} r_k = \nabla f(x_k) &= b - Ax_k \\ \alpha_k &= 2\frac{\min - f(x_k)}{\|r_k\|^2} \\ x_{k+1} &= x_k + \alpha_k\nabla r_k \end{aligned} \tag{2.7}$$

and can be iterated until a desired tolerance $\epsilon$ is achieved. With our knowledge of the minimum value, the condition $\epsilon < |f(x_k) - \min|$ can be used, however one must take care to ensure $\epsilon$ exceeds any error in the estimation of min.

The method only requires $O(n)$ storage (where $n$ is the number of dimensions), like gradient descent and CG but less than other Quasi-Newton methods that require computation of the Hessian estimate. The method requires roughly the same amount of computation per iteration as Gradient Descent and almost half that of CG, making it superior for cases converging in similar numbers of iterations.

# Chapter 3

# Convergence of Method 1

The convergence of the method is of concern as reducing the amount of computation per iteration is futile if the method does not converge at a similar rate to gradient descent or CG. This chapter provides a *theorem* for convergence under appropriate conditions and a proof of said theorem.

While Method 1 can be applied to any function, at present, we can only prove convergence for convex quadratic functions with condition number less than two.

**Theorem** *If the condition number of symmetric, positive definite $A$ is less than $2-\epsilon$, for some positive value $\epsilon$, then the method in* (2.7)*, applied to a quadratic problem with Hessian $A$ converges linearly.*

**Proof** Define $\Gamma$ to be $\Phi|\mu$ ($\Phi$ restricted to $\mu$) where $\mu$ is a plane containing the minimum point and the current search direction. Without loss of generality $\Gamma$ is translated so the minimum is 0, and can be expressed as

$$\Gamma\left(\begin{smallmatrix} x \\ y \end{smallmatrix}\right) = \left(\begin{smallmatrix} x \\ y \end{smallmatrix}\right)^T A \left(\begin{smallmatrix} x \\ y \end{smallmatrix}\right) \tag{3.1}$$

with gradient

$$\nabla\Gamma\left(\begin{smallmatrix} x \\ y \end{smallmatrix}\right) = 2A\left(\begin{smallmatrix} x \\ y \end{smallmatrix}\right) \tag{3.2}$$

Where $A$ is a $2 \times 2$ matrix. We wish to relate the convergence to the eigenvalues of $A$, and thus, without loss of generality, we can work in a basis such that $A$ is a diagonal matrix,

$$A = \left(\begin{smallmatrix} \lambda_1 & \\ & \lambda_2 \end{smallmatrix}\right). \tag{3.3}$$

Our algorithm will converge if each step produces a sufficient decrease, i.e. if for each iteration $k$,

$$\Gamma\left(\begin{smallmatrix} x_{k+1} \\ y_{k+1} \end{smallmatrix}\right) - \Gamma\left(\begin{smallmatrix} x_k \\ y_k \end{smallmatrix}\right) < \epsilon,$$

for $\epsilon < 1$, defined as part of the theorem definition.

To prove this we need to simplify

$$\Gamma\left(\begin{smallmatrix} x_{k+1} \\ y_{k+1} \end{smallmatrix}\right) = \Gamma\left(\left(\begin{smallmatrix} x_k \\ y_k \end{smallmatrix}\right) - s_k \frac{\nabla\Gamma\left(\begin{smallmatrix} x_k \\ y_k \end{smallmatrix}\right)}{\|\nabla\Gamma\left(\begin{smallmatrix} x_k \\ y_k \end{smallmatrix}\right)\|}\right), \tag{3.4}$$

where $s_k$ is the step in the negative gradient direction. The step length in our algorithm is the distance to the minimum of an approximating parabola which is tangent to $\Gamma$ at $(x_k, y_k)$ and has a minimum at zero. It can therefore be written as

$$f(s) = \Gamma\left(\left(\begin{smallmatrix} x_k \\ y_k \end{smallmatrix}\right) - u \frac{\nabla\Gamma\left(\begin{smallmatrix} x_k \\ y_k \end{smallmatrix}\right)}{\|\nabla\Gamma\left(\begin{smallmatrix} x_k \\ y_k \end{smallmatrix}\right)\|}\right) \tag{3.5}$$

$$= a(s_k - u)^2, \tag{3.6}$$

for some positive constant $a$. The tangency gives us two equations

$$as_k^2 = f(0) = \Gamma\left(\begin{smallmatrix} x_k \\ y_k \end{smallmatrix}\right) \tag{3.7}$$

$$= \lambda_1 x_k^2 + \lambda_2 y_k^2, \text{ and} \tag{3.8}$$

$$-2as_k = f'(0) = \|\nabla\Gamma\left(\begin{smallmatrix} x_k \\ y_k \end{smallmatrix}\right)\| \tag{3.9}$$

$$= \sqrt{\lambda_1^2 x_k^2 + \lambda_2^2 y_k^2}. \tag{3.10}$$

Dividing the first by negative the second yields

$$s_k = \frac{\lambda_1 x_k^2 + \lambda_2 y_k^2}{\sqrt{\lambda_1^2 x_k^2 + \lambda_2^2 y_k^2}}. \tag{3.11}$$

Substituting this into (3.4),

$$\begin{aligned}
\Gamma\left(\begin{smallmatrix} x_{k+1} \\ y_{k+1} \end{smallmatrix}\right) &= \Gamma\left(\left(\begin{smallmatrix} x_k \\ y_k \end{smallmatrix}\right) - s_k \frac{\nabla\Gamma(x_k)}{\|\nabla\Gamma(x_k)\|})\right) \\
&= \Gamma\left(\left(\begin{smallmatrix} x_k \\ y_k \end{smallmatrix}\right) - \frac{\lambda_1 x_k^2 + \lambda_2 y_k^2}{\sqrt{\lambda_1^2 x_k^2 + \lambda_2^2 y_k^2}} \frac{\left(\begin{smallmatrix} \lambda_1 x_k \\ \lambda_2 y_k \end{smallmatrix}\right)}{\sqrt{\lambda_1^2 x_k^2 + \lambda_2^2 y_k^2}}\right) \\
&= \Gamma\begin{pmatrix} x_k(1 - \lambda_1 \frac{\lambda_1 x_k^2 + \lambda_2 y_k^2}{\lambda_1^2 x_k^2 + \lambda_2^2 y_k^2}) \\ y_k(1 - \lambda_2 \frac{\lambda_1 x_k^2 + \lambda_2 y_k^2}{\lambda_1^2 x_k^2 + \lambda_2^2 y_k^2}) \end{pmatrix} \\
&= \lambda_1(x_k(1 - \lambda_1\Upsilon))^2 + \lambda_2(y_k(1 - \lambda_2\Upsilon))^2
\end{aligned}$$

where, to ease future simplifications, we define the term

$$\Upsilon = \frac{\lambda_1 x_k^2 + \lambda_2 y_k^2}{\lambda_1^2 x_k^2 + \lambda_2^2 y_k^2}. \tag{3.12}$$

With this expression, we are able to simplify

$$\begin{aligned}
\Gamma\left(\begin{smallmatrix} x_{k+1} \\ y_{k+1} \end{smallmatrix}\right) - \Gamma\left(\begin{smallmatrix} x_k \\ y_k \end{smallmatrix}\right) &= \lambda_1(x_k(1-\lambda_1\Upsilon))^2 + \lambda_2(y_k(1-\lambda_2\Upsilon))^2 - (\lambda_1 x_k^2 + \lambda_2 y_k^2) \\
&= \lambda_1 x_k^2(1 - 2\lambda_1\Upsilon + \lambda_1^2\Upsilon^2) - \lambda_1 x_k^2 \\
&\quad + \lambda_2 y_k^2(1 - 2\lambda_2\Upsilon + \lambda_2^2\Upsilon^2) - \lambda_2 y_k^2 \\
&= \Upsilon\left((\lambda_1^3 x_k^2\Upsilon - 2\lambda_1^2 x_k^2) + (\lambda_2^3 y_k^2\Upsilon - 2\lambda_2^2 y_k^2)\right) \\
&= (\lambda_1\Upsilon)\lambda_1 x_k^2(\lambda_1\Upsilon - 2) + (\lambda_2\Upsilon)\lambda_2 y_k^2(\lambda_2\Upsilon - 2)
\end{aligned}$$

Now remark that the positive definiteness of $A$ implies that

$$\begin{aligned}
\lambda_1, \lambda_2 > 0, &\text{ which implies} \\
\Upsilon > 0, &\text{ which implies} \\
\lambda_1\Upsilon, \lambda_2\Upsilon > 0 &
\end{aligned}$$

and that the expressions $\lambda_1\Upsilon$ and $\lambda_2\Upsilon$ are invariant under scaling of $A$, and that if $\frac{1}{2-\epsilon} < \lambda_1/\lambda_2 < 2 - \epsilon$, then

$$\lambda_1\Upsilon, \lambda_2\Upsilon > \frac{1}{2}, and$$

$$\begin{aligned}
\Gamma\left(\begin{smallmatrix} x_{k+1} \\ y_{k+1} \end{smallmatrix}\right) - \Gamma\left(\begin{smallmatrix} x_k \\ y_k \end{smallmatrix}\right) &< -\frac{1}{2}\min\{\epsilon, \frac{1}{2}\}\left(\lambda_1 x_k^2 + \lambda_2 y_k^2\right) && (3.13) \\
&= -\frac{1}{2}\min\{\epsilon, \frac{1}{2}\}\Gamma\left(\begin{smallmatrix} x_k \\ y_k \end{smallmatrix}\right), && (3.14)
\end{aligned}$$

which is exactly the condition we need to guarantee linear convergence. Q.E.D.

# Chapter 4

# Method 2:

# Augmented with Line Search

This chapter seeks to improve the previous method by augmenting the algorithm with a simple line search, while still taking advantage of the scale factor developed. We start by explaining our choice of line search, how it is performed and why it is used. The chapter finishes by presenting the resulting method, Method 1 augmenting with the line search. We provide an implementation of method in Matlab code.

## 4.1 A Line Search for Method 1

Since the descent direction is already being scaled the addition of a conventional line search should be redundant. However a line search may provide an advantage for highly non-linear or ill-conditioned problems. We wish to take create a method that performs a line search while still taking advantage of the initial step-size (2.6). Since Method 1 is computed using a paraboloid in n-dimensions, it makes sense to perform a

search over a univariate function by making a succession of quadratic approximations. We thus propose a quadratic approximation line search where ideally only one fit using (2.6) is necessary.

The quadratic fitting line search can be described as follows, into the step

$$x_{k+1} = x_k + \alpha \nabla f(x_k), \tag{4.1}$$

we attempt to find a better $\alpha$ by fitting a parabola

$$m(\alpha) = a\alpha^2 + b\alpha + c \tag{4.2}$$

$$b = \nabla f(x_k) \cdot d_k \tag{4.3}$$

$$c = f(x_k) \tag{4.4}$$

to our data by iterating

$$a = (f(x_k + \alpha_i d_k) - b\alpha_i - c)/\alpha_i^2 \tag{4.5}$$

$$\alpha_{i+1} = \frac{-b}{2a} \tag{4.6}$$

on an initial $\alpha_0$ (the $\alpha$ derived in (2.6)) until the Armijo Rule

$$f(x_k + \alpha_i d_k) \leq f(x_k) + c_1 \alpha_i (d_k \cdot \nabla f(x_k)) \qquad \text{(Armijo Rule)}$$

is fulfilled. In an ideal case, the fitted parabola is a path on $\Phi(d)$, and it's minimum is the minimum of $\Phi(d)$ (2.1). In this case the parabola will fit and bring us to the

minimum in one step.

$$2\frac{\min - f(x_k)}{\|\nabla f(x_k)\|^2} \tag{4.7}$$

However, even in less ideal cases as long as the **Armijo Rule** (Armijo Rule) is met we are ensured $f(x_{k+1})$ decreases "sufficiently" and haven't wasted our time.

In the case a few succession of fittings does not satisfy the Armijo Rule, $\alpha$ is unlikely to be useful and the method has failed. If quadratic fitting fails, a line search can be computing using a different (backup) method. For my implementation, if the a parabola fails to fit four times, a Backtracking Line Search using the Wolfe Conditions is performed. This can easily be modified to incorporate other methods to suit different needs.

## 4.2 Resulting Method — Method 2

This algorithm can be implemented in in Matlab using the following code

```
function [ xp ] = method2( f, grad, x, min, eps )
    xp = x;
    fi = f(x);
    while abs(fi - min) > eps
        % Compute Function Values for next iteration
        fi = f(x);
        gradi = grad(x);
        % Compute Descent Direction
        d = 2*((min - fi) / dot(gradi,gradi)) * gradi;
```

```
        % line search
        alpha = quadraticFitSearch(f,fi,xp,gradi,d);
        % Iterate
        xp = xp + alpha*d;
    end
end
```

```
function alpha = quadraticFitSearch(f,fv,xp,grad,dir)
    % use computed scaling for initial fit
    alpha = 1.0;
    % compute fitted parabola
    c          = fv;
    b          = dot(dir,grad);
    fv_new     = f(xp+alpha*dir);
    a          = ( fv_new - b*alpha - c) / alpha^2;
    alpha      = - b / (2*a);
    fv_new = f(xp+alpha*dir);

    iter = 0;
    c_1  = 1e-4;
    % iterate until armijo rule is fulfilled
    while fv_new > fv + c_1*alpha*dot(dir,grad);
        iter = iter + 1;
        a     = (fv_new - b*alpha - c) / alpha^2;
        alpha = - b / (2*a);
        if iter > 4
            warning(['Quadratic approximation failed more than 4 times\n']);
            return;
        end
        fv_new = f(xp+alpha*dir);
    end
end
```

# Chapter 5

# Performance Metrics

This chapter provides performance metrics done by measuring the number of iterations, taken for sample benchmark problems for Method 1 and 2. We start by comparing each method against gradient descent for non-linear optimization benchmark functions, that are well-known for having tricky valley's. We then compare that method against CG for tricky linear problems, and randomly generated linear problems over varying condition numbers.

## 5.1 Method 1/2 vs. Gradient Descent

Method 1 and 2 were benchmarked against an implementation of the Gradient Descent Method using a backtracking line search. We benchmark against a variety of well-known test functions for optimization, known for being particularly tricky, as well as a quartic function in significantly large dimensions. It is expected that Method 1 should make larger step sizes than Gradient Descent at most iterations, as well as save iterations on the line search.

Table 5.1 lists the test functions used as well as the sample initial points used to obtain the results in Table 5.2. Instead of testing against a termination condition, $\epsilon > |f - min|$, the termination condition $1.0e^{-7} > \|x_k - x_{k-1}\|$ was used, based on the step size.

| | |
|---|---|
| Function Name | **Rosenbrock** |
| Function | $f(x,y) = (1-x)^2 + 100(y-x^2)^2$ |
| Initial Point | (-1.0,1.5) |
| Final Point | (1.0,1.0) |
| Minimum Value | 0 |
| Function Name | **Beale's** |
| Function | $f(x,y) = (1.5 - x + xy)^2 + (2.25 - x + xy^2)^2 + (2.625 - x + xy^3)^2$ |
| Initial Point | (-1.5,4.5) |
| Final Point | (3,0.5) |
| Minimum Value | 0 |
| Function Name | **Easom** |
| Function | $f(x,y) = -\cos(x)\cos(y)e^{-((x-\pi)^2+(y-\pi)^2)}$ |
| Initial Point | (2.2,3.8) |
| Final Point | ($\pi$,$\pi$) |
| Minimum Value | -1 |
| Function Name | **Booth's** |
| Function | $(x+2y-7)^2 + (2x+y-5)^2$ |
| Initial Point | (4.5,1.5) |
| Final Point | (1.0,3.0) |
| Minimum Value | 1 |
| Function Name | **Sphere** |
| Function | $x^2 + (y-1)^2 + 1$ |
| Initial Point | (1.5,1.5) |
| Final Point | (0.0,1.0) |
| Minimum Value | 1 |
| Function Name | **Large Quartic Function** |
| Function | $\sum_{i=1}^{10000} (i - x_i)^4$ |
| Initial Point | $x_i = 0$ forall $i$ |
| Final Point | $x_i$ = i |
| Minimum Value | 1 |

Table 5.1: Benchmark Functions

| | Method 1 Iter. | Method 2 Iter. | Gradient Descent Iter. |
|---|---|---|---|
| Rosenbrock | 1230* | 140 | 7369* |
| Easom | 3* | 5 | 23 |
| Booth's | 96 | 18 | 24 |
| Sphere | 1 | 1 | 1 |
| Beale's | $> 10000$ | $> 10000$ | $> 10000$ |
| Large Quartic | 23 | 25 | $> 10000$ |

Table 5.2: Methods 1 & 2 vs. Gradient Descent Total Iterations. Number of iterations, in the case that the algorithm terminated in fewer than 10 000 iterations. *Halted at false minimum.

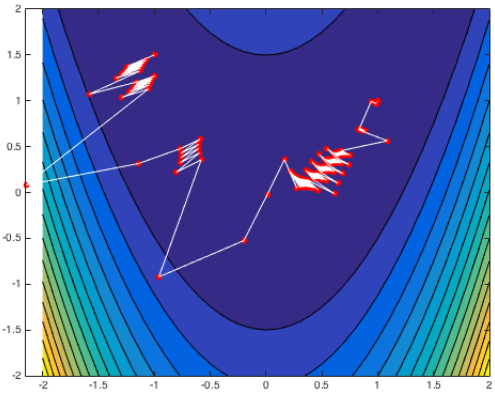
### 5.1.1 Total Iterations

The total iterations (including line search iterations) are displayed in Table 5.2. The Method 2 shows an improvement on all benchmark functions, except the trivial case of the sphere and Beale's function which failed to converge for both methods. The troublesome Rosenbrock function, known for its difficult to navigate valleys, converged in relatively few iterations compared to Gradient Descent or Method 1.

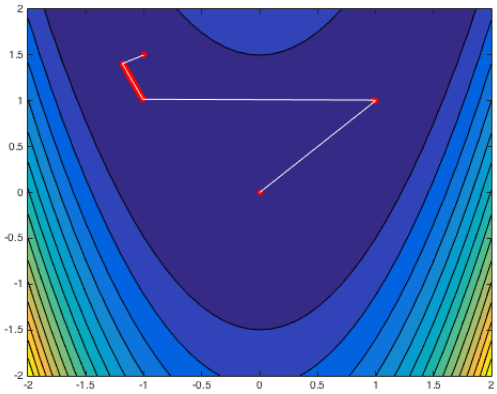
A particularly interesting result is the Large Quartic function, which Gradient Descent failed to converge within a reasonable amount of iterations (maximum iterations was set to 10,000). Method 1 was actually slightly more successful than Method 2, evidently making unnecessary extra steps in the line search, meaning initial quadratic fit did not satisfy the **Armijo Rule** even though overall they were satisfactory step sizes.

### 5.1.2 Convergence Paths

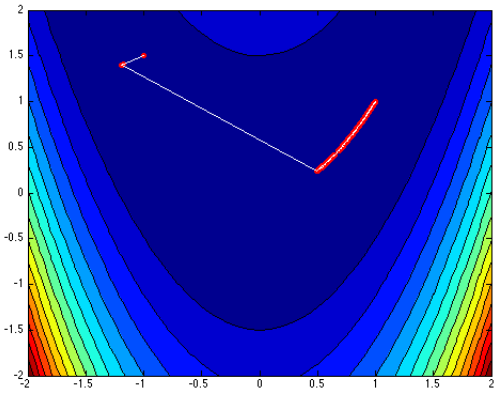
The following plots exhibit the paths taken by each method over a contour plot of the specified function



(a) Method 1

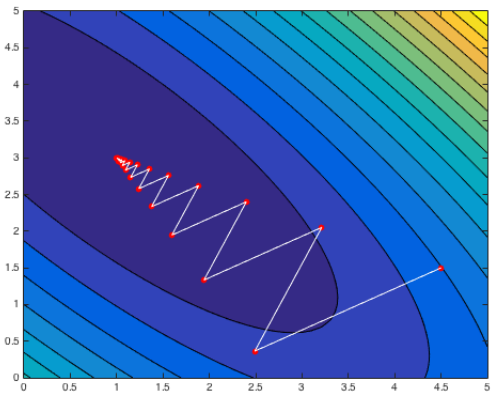


(b) Method 2

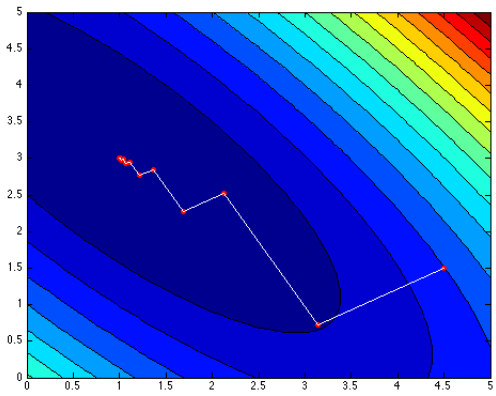


(c) Gradient Descent

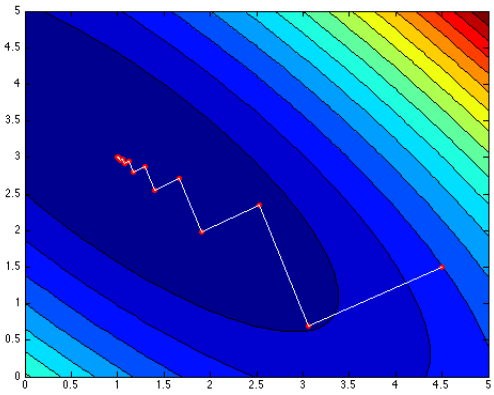
Figure 5.1: Rosenbrock Function Convergence Path




(a) Method 1

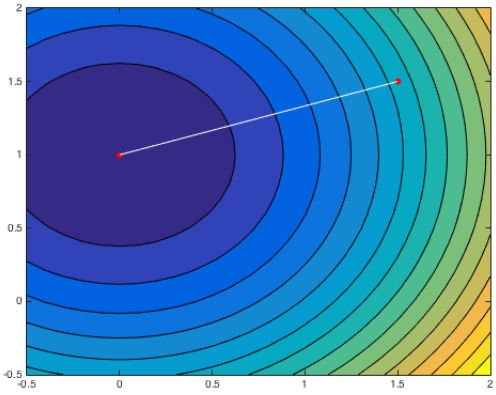



(b) Method 2

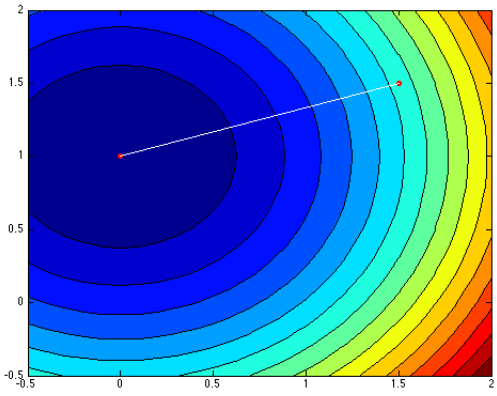



(c) Gradient Descent

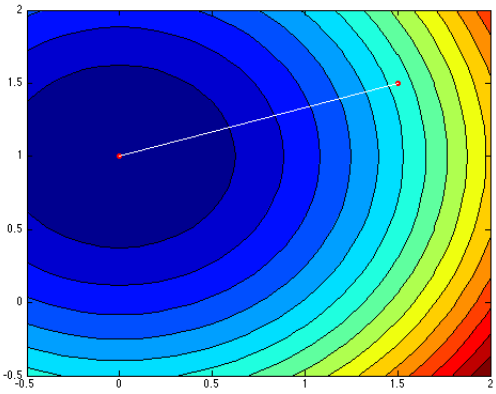
Figure 5.2: Booth's Function Convergence Path




(a) Method 1

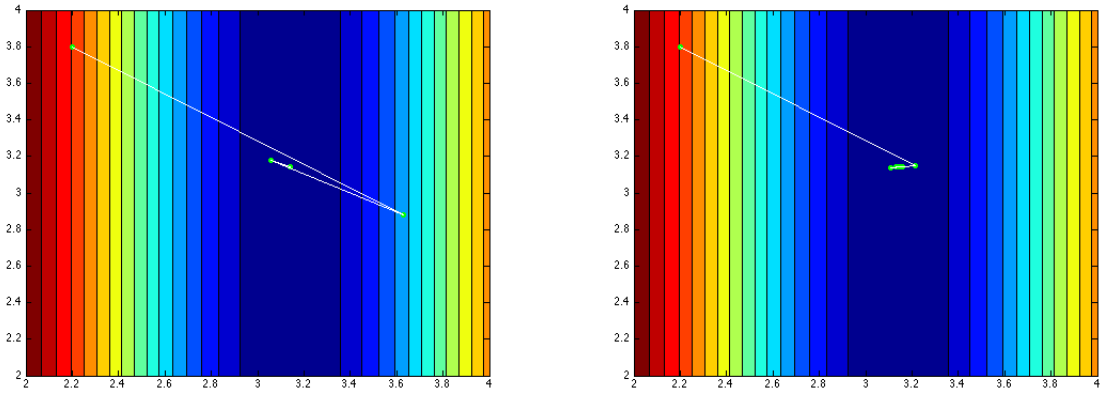



(b) Method 2




(c) Gradient Descent

Figure 5.3: Sphere Function Convergence Path



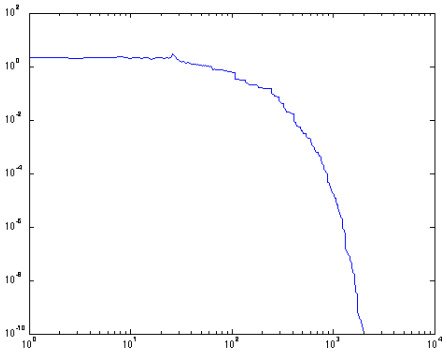

(a) Method 2 (b) Gradient Descent

Figure 5.4: Easom's Function Convergence Path

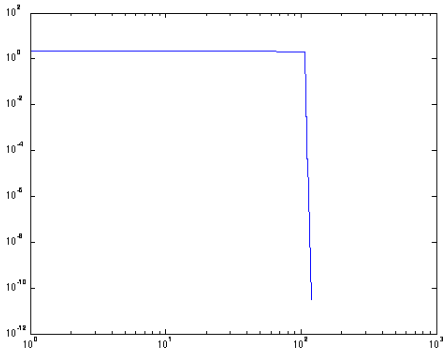
Method 2 shows a marginal improvement in step-size over Gradient Descent for Rosenbrock, Booth's, and Easom's functions, where Method 2 largely fails. As one would expect all methods converges in a single step on a sphere. Of particular notice is the sporadic nature of the path for Method 1, which failed to converge at the true minimum. Also the lack of a Method 1 path for Easom's function, which quickly diverged making too small step sizes. Also Method 1 made noticeably less successful step sizes than even Gradient Descent for Booths Function.

### 5.1.3 Convergence Rates

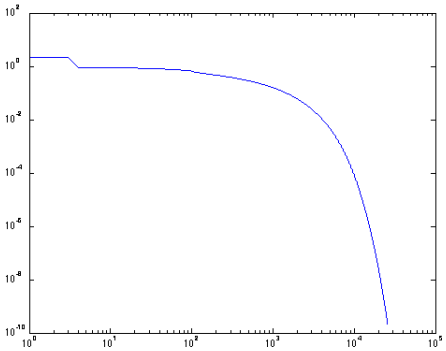
The following plots represent the distance $\|x_k - L\|$ of a point $x_k$ at iteration $k$ from the minimum point $L$, plotted on a log-log scale.



(a) Method 1

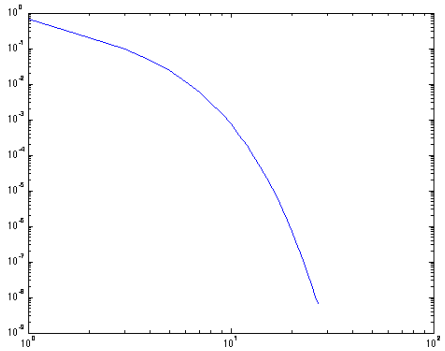


(b) Method 2

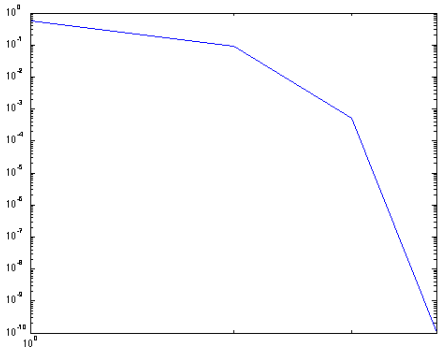


(c) Gradient Descent

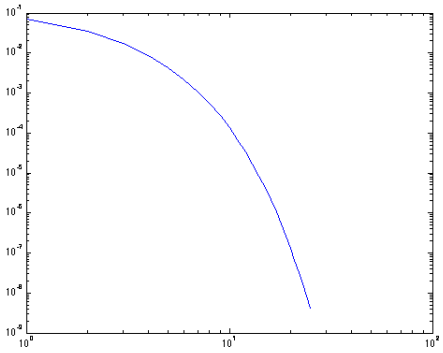
Figure 5.5: Rosenbrock's Function Convergence Rate



(a) Method 1

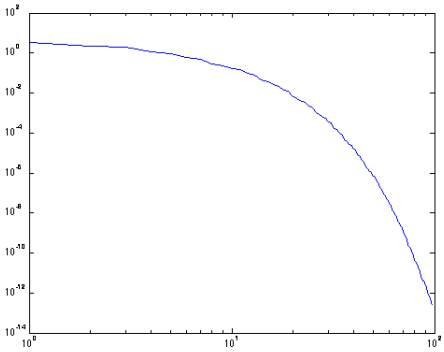


(b) Method 2

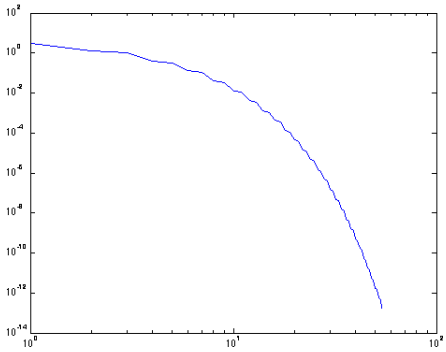


(c) Gradient Descent

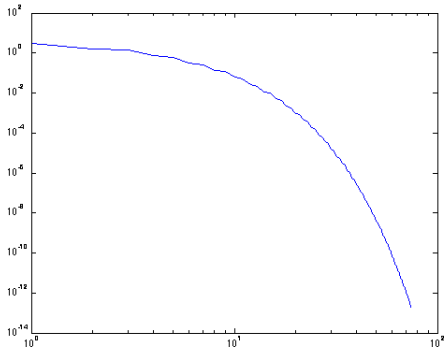
Figure 5.6: Easom's Function Convergence Rate



(a) Method 1

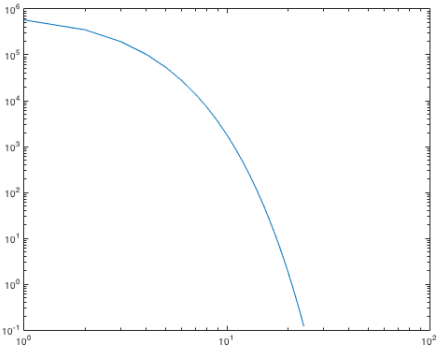


(b) Method 2

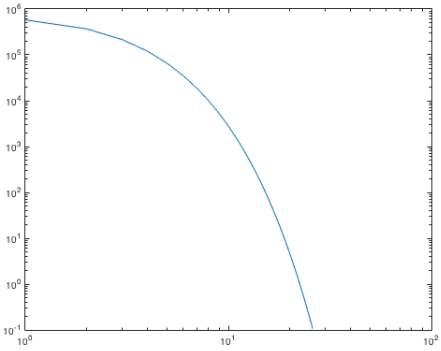


(c) Gradient Descent

Figure 5.7: Booth's Function Convergence Rate



(a) Method 1



(b) Method 2

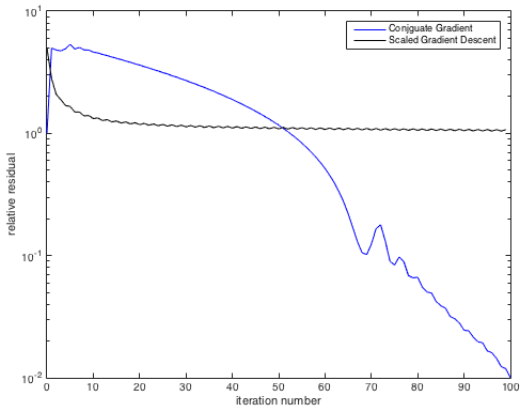
Figure 5.8: Large Quartic Convergence Rate

| Condition # | Max/Min EigenValue | CG Iter. | Method 1 | Method 2 |
|---|---|---|---|---|
| 1.00 | 1.0 / 1.0 | 1 | 1 | 1 |
| 1.05 | 1.0 / 0.95 | 4 | 5 | 5 |
| 1.10 | 1.0 / 0.90 | 5 | 6 | 6 |
| 1.20 | 1.0 / 0.85 | 6 | 7 | 7 |
| 1.25 | 1.0 / 0.80 | 6 | 8 | 8 |
| 1.30 | 1.0 / 0.75 | 7 | 9 | 9 |
| 1.40 | 1.0 / 0.70 | 7 | 10 | 10 |
| 1.50 | 1.0 / 0.65 | 8 | 11 | 11 |
| 1.65 | 1.0 / 0.60 | 8 | 12 | 12 |
| 1.80 | 1.0 / 0.55 | 9 | 14 | 14 |
| 2.0 | 1.0 / 0.50 | 10 | 16 | 15 |
| 2.2 | 1.0 / 0.45 | 10 | 17 | 17 |
| 2.5 | 1.0 / 0.40 | 11 | 20 | 19 |

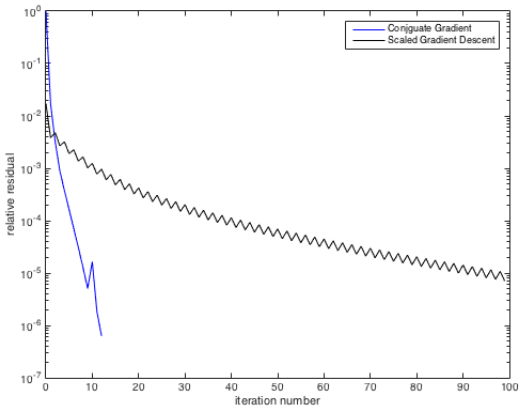
Table 5.3: Linear CG Vs. Methods 1 & 2 Iterations Over Varying Condition Numbers

## 5.2 Method 1/2 vs Conjugate Gradient

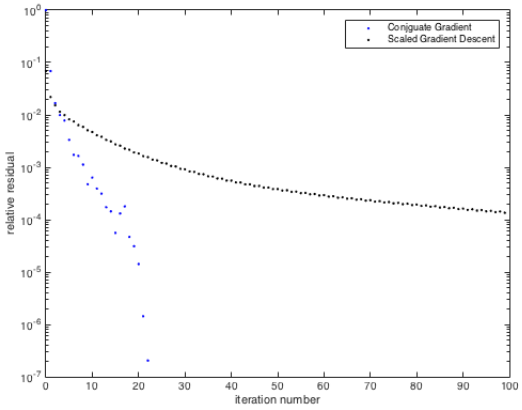
A comparison of the Methods 1/2 and CG was done for linear problems using matlab's pcg function and a few test matrices from the matlab gallery (Matlab 2015a $^{\text{TM}}$) which were preconditioned with an incomplete Cholesky factorization as described in [8], and some sparse randomly generated matrices with varying condition numbers. As required by CG, all matrices were SPD (Symmetric Positive-Definite). All matrices were size $100 \times 100$ and we're solved to a tolerance $\|b - Ax_k\| < 1.0e^{-7}$.




(a) deslq Matrix Convergence Comparison

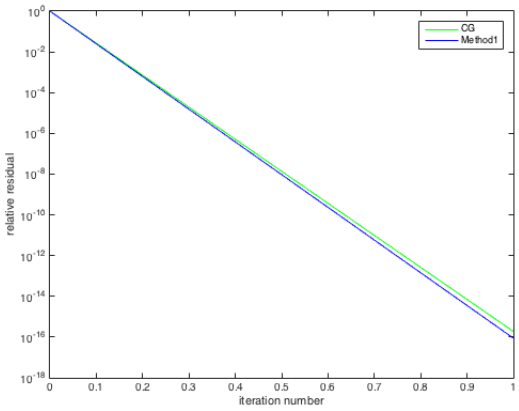



(b) Molar Matrix Convergence Comparison

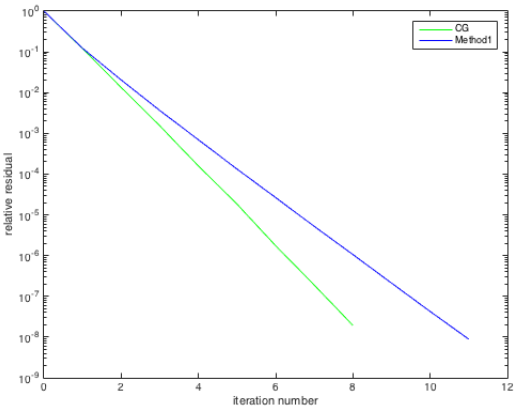



(c) GCD Matrix Convergence Comparison

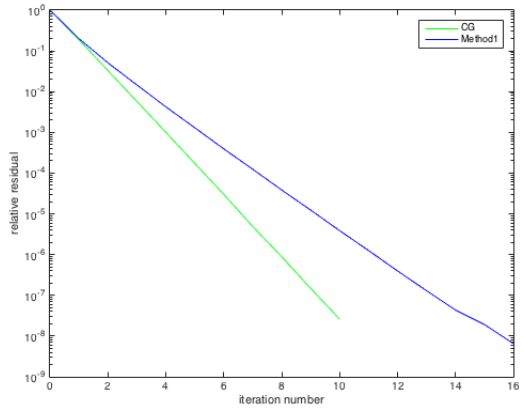
Figure 5.9: Matlab Gallery Test Matrices Convergence Comparison



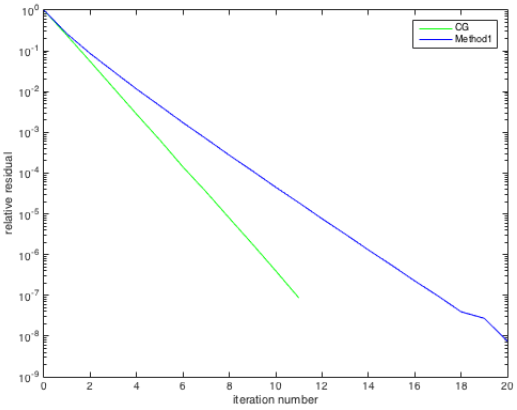

(a) Condition Number = 1.0




(b) Condition Number = 1.5




(c) Condition Number = 2.0




(d) Condition Number = 2.5

Figure 5.10: Random SPD Matrix Convergence Comparison

Method 1 seems to converge linearly for well-conditioned matrices, as well as Method 2 which would largely overlap with Method 1. As expected it performs best for the lowest condition number but quickly degrades when close to the minimum as the condition number exceeds 2.0.

# Chapter 6

# Concluding Remarks

Method 2 showed a general improvement over Gradient Descent for tricky, highly nonlinear and ill-conditioned problems but Method 1 was largely unsuccessful. Interestingly, quadratic problems (including least-squares problems associated with linear systems) (1.1), were solved in roughly the same number of iterations by Methods 1 and 2. Both methods were competitive with CG as long as they were sufficiently well-conditioned.

Method 2 seemed to only be necessary for ill-conditioned problems, however didn't seem to consume to many extra iterations for well-conditioned cases. Augmenting with a line search is therefore still best unless you have high confidence the problem is well-conditioned.

For well-conditioned linear problems, Method 1 seems superior to CG, given it that is competitive in iterations and is less computationally exhaustive at each iteration. The simplicity of Method 1 should also be noted, as it leaves much less room for error.

## Future Work

The method was implemented as an extension of only gradient descent as to maintain simplicity and low-cost per iteration. However the scaling can possibly be used with CG as well, perhaps improving the step size and reducing "zig-zagging" behaviour enough to justify the extra complexity and computation.

Although generic preconditioning did not succeed, it would still be interesting to try problem-specific preconditionners. An alternative iterative method for unconstrained optimization not mentioned in this this is BFGS. BFGS is a quasi-newton method similar to Gradient Descent and CG. The approach of Method 1 could be used to improve BFGS by interleaving iterations or using a similar technique to adjust the scaling as was done in [3] with conjugate gradient. Versions of BFGS that avoid the use of a line search have also been developed, such as in [9], and should be compared to Methods 1 and 2.

# Bibliography

[1] Y-H Dai and L-Z Liao. New conjugacy conditions and related nonlinear conjugate gradient methods. *Applied Mathematics and Optimization*, 43(1):87–101, 2001.

[2] Hyun Keol Kim and Andr Charette. Frequency domain optical tomography using a conjugate gradient method without line search. *Journal of Quantitative Spectroscopy and Radiative Transfer*, 104(2):248 – 256, 2007. {EUROTHERM} {SEMINAR} 78 - {COMPUTATIONAL} {THERMAL} {RADIATION} {IN} {PARTICIPATING} {MEDIA} {IIEUROTHERM} 78.

[3] Dong C. Liu and Jorge Nocedal. On the limited memory bfgs method for large scale optimization. *Mathematical Programming*, 45(1):503–528, 1989.

[4] Jorge J Moré and David J Thuente. Line search algorithms with guaranteed sufficient decrease. *ACM Transactions on Mathematical Software (TOMS)*, 20(3):286–307, 1994.

[5] Larry Nazareth. A conjugate direction algorithm without line searches. *Journal of Optimization Theory and Applications*, 23(3):373–387, 1977.

[6] Jonathan Richard Shewchuk. An introduction to the conjugate gradient method without the agonizing pain, 1994.

[7] Zhen-Jun Shi and Jie Shen. Convergence of descent method without line search. *Applied Mathematics and Computation*, 167(1):94 – 107, 2005.

[8] Jae-Heon Yun and Yu-Du Han. Modifled incomplete cholesky factorization preconditioners for a symmetric positive definite matrix. *Bulletin of the Korean Mathematical Society*, 39(3):495–509, 2002.

[9] Li Zhang. A globally convergent bfgs method for nonconvex minimization without line searches. *Optimization Methods and Software*, 20(6):737–747, 2005.