Convergence and behaviour of various iterative minimization algorithms

Niels Scholte
Mathematics
Bachelor Thesis

Supervisor:
Tristan van Leeuwen
8 June 2018

Utrecht University

Contents

1 Introduction **4**

2 Conjugate Gradients **5**

2.1 Linear Conjugate Gradients 6

2.1.1 Conjugate Directions 6

2.1.2 Conjugate Gradients 9

2.1.3 Convergence in n steps* 11

2.1.4 Convergence in r steps* 12

2.2 Non-linear conjugate gradients 16

2.2.1 Line searches 18

2.2.2 Global Convergence 20

2.2.3 Convergence speed 25

3 Nesterov Accelerated Gradient **29**

3.1 Momentum 29

3.2 Convergence rate 31

3.2.1 Convexity 31

3.2.2 β-smoothness and convexity 33

3.2.3 Convergence analysis 33

3.3 Momentum in practise 35

4 Discussion and conclusions **37**

References **38**

5 Appendix **39**

5.1 Lemmas 39

5.2 NAG reimplemented 39

1 Introduction

Artificial intelligence, or function approximation, is becoming very popular for solving tasks in a wide variëty of fields. These fields include biology [Angermueller et al., 2017], natural language processing [Wu et al., 2016] and image processing [He et al., 2016]. The rise of artifical inteligence, and more specifically deep learning, was catalysed by a breakthrough in image classification in 2012 [Krizhevsky et al., 2012], where it was demonstrated to be much more very effective than previous methods. Deep learning is at its core solving an optimisation problem. Namely, an objective function is minimized by changing parameters based on the gradient of this objective function.

For example, in medical imaging it might be essential to analyze X-ray images effectively and cost efficiently using computer programs, with less need for human experts. A desired function could take the X-ray image pixel data as input and output confidence scores for a certain diagnosis. One could acquire such a function through deep learning by cleverly defining parameters on the input and changing these parameters such that inputs give desired outputs. In other words, an objective function is used to measure how different the realized outputs are from the desired outputs, for training samples in a dataset. This objective is a function of the parameters in the model, thus the objective function can be minimized over these parameters resulting in a highly useful function.

Such functions must contain non linear components if they are to be more accurate than linear classifiers. To accomplish this, a function is built by alternating matrix multiplications, containing parameters, and non linear functions, for increased complexity. This results in a highly complex optimization problem, often containing tens of millions of parameters. Because of this, iterative methods are necessary.

A method of choosing the steps by which the parameters are changed is called an optimizer. In this bachelor thesis, convergence properties and behaviour of several optimizeres are discussed and visualised. The first class of optimizers are variants of the Conjugate Gradients method (Conjugate gradients or CG), which are more classically used methods. Most proofs involving these methods were obtained from [Jorge Nocedal, 2006]. In this chapter, the sections marked with * contain proofs which, although very interesting, are of lesser importance to the overarching theme of this thesis and can therefore be skipped.

The second class of optimizers that we review are the Momentum Based Methods (MBM), in which an average gradient is used to update the parameters. More specifically, Nesterov Accelerated Gradients (NAG) will be discussed in detail. Most proof in this section, more specifically 3.2, are obtained from [Bubeck, 2015].

2 Conjugate Gradients

The Conjugate Gradient method (Also referred to as Conjugate Gradients or CG) was introduced in [Hestenes and Stieffel, 1952] as an alternative method to solving linear systems of the form $Ax = b$, where $b, x \in \mathbb{R}^n$ and $A \in \mathbb{R}^{n \times n}$ a symmetric positive definite matrix. Because this is equivalent to minimizing the quadratic function $f : \mathbb{R}^n \to \mathbb{R}$ given by

$$f(x) := \frac{1}{2} x^T A x - b^T x, \tag{1}$$

the Conjugate Gradient method will be viewed as a way of minimizing an objective function where the function in this particular case is given by equation (1). Since its introduction, many variants have been introduced to also minimize general continuously differentiable functions. The method based on minimizing equation (1) will be called the *linear* Conjugate Gradient method whereas methods for other functions will be called *non-linear* Conjugate Gradient methods.

What makes CG particularly interesting is that for systems in equation (1), CG converges in at most n iterations and often faster depending on the distribution of eigenvalues of the matrix A. These results, assuming exact arithmetic, turn out to partially carry over to non-linear CG. This property of acquiring quick approximate solutions is very valuable for big complex systems as computation can be big.

Because of this, the convergence rate of linear CG is discussed first. Then, the global convergence, the convergence to a stationary point from any starting point, is established for a non-linear CG variant. Finally, the convergence rates of this variant will be compared to other variants through experimental results.

2.1 Linear Conjugate Gradients

2.1.1 Conjugate Directions

The Conjugate Gradient method relies on generating so called *conjugate* directions which we define as follows:

Definition 2.1. The non-zero vectors $p_0, p_1, ..., p_{n-1} \in \mathbb{R}^n$ are said to be conjugate with respect to a symmetrical positive definite matrix $A \in \mathbb{R}^{n \times n}$ if

$$p_i^T A p_j = 0, \qquad \text{for all } i \neq j.$$

The results discussed in this chapter are in and of itself great ways of obtaining intuition for this definition. Therefore, we will refer to figure 1 for a visualisation of conjugate direction vectors in action and only remark that conjugacy implies linear independence. This can be proven by contradiction by assuming that the conjugate vectors are not linearly independent. If that is the case then there exist a $p_m \in \{p_0, p_1, ..., p_{n-1}\}$, and an $l \neq m$, for which $0 \neq a_l$, such that

$$p_m = \sum_{\substack{k=0 \\ k \neq m}}^{n-1} a_k p_k,$$

where $a_i \in \mathbb{R}$. Because $p_0, p_1, ..., p_{n-1}$ are conjugate we find that

$$\begin{aligned} 0 &= p_l^T A p_m \\ &= p_l^T A \sum_{\substack{k=0 \\ k \neq m}}^{n-1} a_k p_k \\ &= a_l p_l^T A p_l \neq 0, \end{aligned}$$

because A is positive definite and $a_l \neq 0$.

Using such a set of conjugate directios, the method finds the minimizer x^* of equation (1) by minimizing the objective function along each conjugate direction separately. This results in an iterative algorithm where at each iteration, the next point x_{k+1} is generated by minimizing the objective function f along a conjugate direction p_k. The step size α_k along p_k, required to get to this minimum, can be found by deriving $f(x_k + \alpha p_k)$ with respect to α. So define

$$\begin{aligned} g_k(\alpha) &= f(x_k + \alpha p_k) \\ &= \frac{1}{2}(x_k + \alpha p_k)^T A(x_k + \alpha p_k) - b^T(x_k + \alpha p_k) \\ &= \alpha^2(\frac{1}{2} p_k^T A p_k) + \alpha\big((x_k^T A - b^T)p_k\big) + (\frac{1}{2} x_k^T A x_k - b^T x_k). \end{aligned}$$

Since A is positive definite, $g_k''(\alpha) = p_k^T A p_k > 0$. Minimizing over α then gives

$$\alpha_k = -\frac{(Ax_k - b)^T p_k}{p_k^T A p_k}. \tag{2}$$

Because this step size applies to all sets of conjugate directions, we can define an algorithm that, unlike GC, does not generate its own conjugate directions and study its convergence. This algorithm, shown in algorithm 1, is called the conjugate direction method.

Algorithm 1 Conjugate Direction Method

1: Given x_0 and a set of conjugate vectors $\{p_0, p_1, ..., p_{n-1}\}$;
2: Set $k \leftarrow 0$;
3: **while** $Ax_k \neq b$ **do**

$$\alpha_k \leftarrow -\frac{(Ax_k - b)^T p_k}{p_k^T A p_k}$$

$$x_{k+1} \leftarrow x_k + \alpha_k p_k$$

$$k \leftarrow k + 1$$

This algorithm has various interesting properties. Before investigating those, we introduce the residuals which are the gradients of f at the iterates, given by

$$r_k = Ax_k - b.$$

These can be written as a function of the previous residual, namely

$$\begin{aligned} r_{k+1} &= A(x_k + \alpha_k p_k) - b \\ &= r_k + \alpha_k A p_k. \end{aligned} \tag{4}$$

Using this we find that

$$\begin{aligned} r_{k+1}^T p_k &= (r_k + \alpha_k A p_k)^T p_k \\ &= r_k^T p_k - \frac{r_k^T p_k}{p_k^T A p_k} p_k^T A p_k \\ &= 0. \end{aligned} \tag{5}$$

This result is not surprising considering the fact that α_k minimizes the objective function f along p_k, therefore an orthogonal gradient is to be expected. However, using induction, it can also be shown that the residuals produced by Algorithm (1) are orthogonal to all previous directions. In other words,

$$r_{k+1}^T p_i = 0, \qquad \text{for } i = 0, 1, ..., k. \tag{6}$$

To proof this, first note that in particular for $k = 0$, we have that $r_1^T p_0 = 0$ by equation (5). Then, assuming the induction hypothesis

$$r_{k+1}^T p_i = 0, \qquad \text{for } i = 0, 1, ..., k,$$

we note that $r_{k+2}^T p_{k+1} = 0$ by equation (5). Similarly we have for $i \neq k+1$ that

$$\begin{aligned} r_{k+2}^T p_i &= (r_{k+1} + \alpha_{k+1} A p_{k+1})^T p_i \\ &= r_{k+1}^T p_i + \alpha_{k+1} p_{k+1}^T A p_i \\ &= 0, \end{aligned}$$

since $p_{k+1}^T A p_i = 0$ by conjugacy and $r_{k+1}^T pi = 0$ using the induction hypothesis.

A second interesting property is that algorithm 1 converges to the minimum in at most n steps, which will be used to show that CG converges in at most n steps. In other words, we can show that $x_n = x^*$ if $x^* \neq x_k$ for all $k \in \{0, 1, ..., n-1\}$. To prove this we first note that by the definition of the iterates we have

$$x_n - x_0 = \alpha_0 p_0 + ... + \alpha_{n-1} p_{n-1}.$$

Secondly, we note that the conjugate vectors $p_0, p_1, ..., p_{n-1}$ are linearly independent as shown earlier. Therefore, they span $\mathbb{R}^n$ and we have that

$$x^* - x_0 = \mu_0 p_0 + ... + \mu_{n-1} p_{n-1} \tag{7}$$

for some $\mu_0, ..., \mu_{n-1} \in \mathbb{R}$. Using conjugacy this results in

$$\begin{aligned} \mu_k p_k^T A p_k &= p_k^T A(\mu_0 p_0 + ... + \mu_{n-1} p_{n-1}) \\ &= p_k^T A(x^* - x_0), && \text{by equation (7);} \\ &= p_k^T A(x^* - x_k + x_k - x_0) \\ &= p_k^T A(x^* - x_k) + p_k^T A(x_k - x_0). \end{aligned}$$

By the definition of the algorithm we have for the second term on the RHS that

$$p_k^T A(x_k - x_0) = p_k^T A(\alpha_0 p_0 + ... + \alpha_{k-1} p_{k-1}) = 0,$$

by conjugacy. So,

$$\mu_k p_k^T A p_k = p_k^T \big(A(x^* - x_k)\big) = -p_k^T (A x_k - b)$$

resulting in $\mu_k = -\frac{(Ax_k - b)^T p_k}{p_k^T A p_k} = \alpha_k$, for $k = 0, 1, ..., n-1$. Therefore, $x^* = x_n$.

We can, however, make this result slightly stronger using a multivariate approach.

Lemma 2.1. The iterates x_k generated by the algorithms 1 minimize $f(x) = \frac{1}{2} x^T A x - b^T x$ from equation (1) over the set

$$\{x + x_0 | x \in \text{span}\{p_0, p_1, ..., p_{k-1}\}\} \tag{8}$$

Proof. Notice first that if the minimizer of f over the given set exists, it is of the form $\hat{x}_{k+1} = x_0 + \hat{a}_0 p_0 + \hat{a}_1 p_1 + ... + \hat{a}_k p_k = x_0 + P_k \hat{\mathbf{a}}_k$, where $P_k = \big(p_0, p_1, ..., p_k\big)$ and $\hat{\mathbf{a}}_k = \big(\hat{a}_0, \hat{a}_1, ..., \hat{a}_k\big)^T \in \mathbb{R}^k$. Since $\hat{x}_{k+1}$ is only variable in $\hat{\mathbf{a}}_k$ for minimizing over the given set, we now define $g_k : \mathbb{R}^k \to \mathbb{R}$ as

$$\begin{aligned} g_k(\mathbf{a}_k) &:= f(x_0 + P_k \mathbf{a}_k) \\ &= \frac{1}{2}(x_0 + P_{k-1}\mathbf{a}_k)^T A(x_0 + P_k \mathbf{a}_k) - b^T(x_0 + P_k \mathbf{a}_k) \\ &= \frac{1}{2}\mathbf{a}_k^T (P_k^T A P_k)\mathbf{a}_k + \big((x_0^T A - b^T)P_k\big)\mathbf{a}_k + \frac{1}{2}x_0^T A x_0, \end{aligned}$$

and notice that that $P_k^T A P_k$ is a diagonal matrix due to conjugacy. We also notice that it has only positive elements on the diagonal, due to A being positive definite. Because the standard unit vectors are eigenvectors of a diagonal matrix, the eigenvalues lie on the diagonal. Therefore, $\nabla^2 g_k(a) = P_k^T A P_k$ is positive definite and thus g_k is a convex quadratic function. Because of this, the minimizer $\hat{x}_k$ exists and we have that $\hat{\mathbf{a}}_k$ minimizes g_k. Therefore $\nabla g_k(\hat{\mathbf{a}}_k) = 0 \in \mathbb{R}^k$, as ∇g_k is continuous. Combining this with the fact that $P_k^T A P_k$ is diagonal we now notice that for $\alpha = \big(\alpha_0, \alpha_1, ..., \alpha_k\big)$ as in in algorithm 1, $\alpha = \hat{a}$. Therefore, $\big(\alpha_0, \alpha_1, ..., \alpha_k\big)$ minimizes g, thus minimizing f over the set given by equation (8). □

In particular, this also implies that algorithm 1 terminates in at most n steps.

2.1.2 Conjugate Gradients

In contrast to algorithm 1, CG generates its own conjugate directions. It does so by setting the first direction to be $p_0 = -r_0$ and its next directions to be

$$p_{k+1} = -r_{k+1} + \beta_{k+1} p_k, \tag{9}$$

where β_{k+1} is determined by imposing $p_k^T A p_{k+1} = 0$. This results in

$$\begin{aligned} 0 &= p_k^T A\big(p_{k+1}\big) \\ &= p_k^T A\big(-r_{k+1} + \beta_{k+1} p_k\big) \\ &= -p_k^T A r_{k+1} + \beta_{k+1} p_k^T A p_k, \end{aligned}$$

which yields $\beta_{k+1} = \frac{r_{k+1}^T A p_k}{p_k^T A p_k}$. The resulting algorithm is given in Algorithm 2 and an illustration of the algorithm in two dimensions is given in figure 1.

Algorithm 2 Conjugate Gradient Method

1: Given x_0;
2: Set $r_0 \leftarrow Ax_0 - b$, $p_0 \leftarrow -r_0$, $k \leftarrow 0$;
3: **while** $r_k \neq 0$ **do**

$$\alpha_k \leftarrow -\frac{r_k^T p_k}{p_k^T A p_k}$$

$$x_{k+1} \leftarrow x_k + \alpha_k p_k$$

$$r_{k+1} \leftarrow Ax_{k+1} - b$$

$$\beta_{k+1} \leftarrow \frac{r_{k+1}^T A p_k}{p_k^T A p_k}$$

$$p_{k+1} \leftarrow -r_{k+1} + \beta_{k+1} p_k$$

$$k \leftarrow k + 1$$

Although two dimensions is not enough to illustrate the method's behaviour beyond the direct consequences of how it was constructed, it still gives insight in how the method operates.

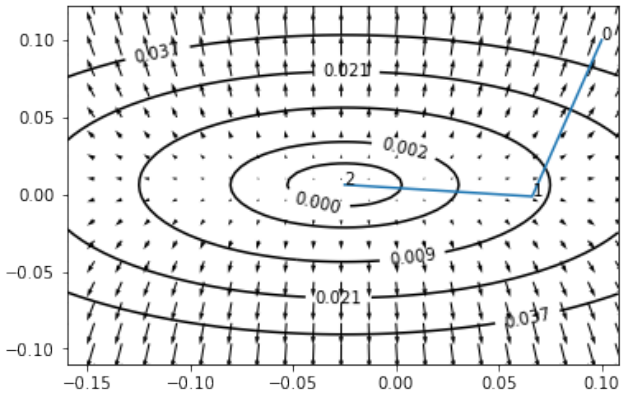


Figure 1: Conjugate gradients applied to the system $f(x) = \frac{1}{2}x^T \left(\begin{smallmatrix} 2 & 0 \\ 0 & 8 \end{smallmatrix}\right) x - \left(\begin{smallmatrix} -.05 \\ .05 \end{smallmatrix}\right) x$. The ellipses are (rounded) level curves of f, the vector field illustrates the gradient ∇f and the blue line shows the steps taken by CG.

2.1.3 Convergence in n steps*

This section focuses on showing that CG generates conjugate directions, enabling us to use Lemma 2.1 to show that CG converges in at most n steps. It can be skipped if one is only interested in methods for non-linear objective functions.

To show that the directions p_k are conjugate, we first need to show that

$$\text{span}\{p_0, p_1, ..., p_k\} = \text{span}\{r_0, r_1, ..., r_k\} = \text{span}\{r_0, Ar_0, ..., A^k r_0\}. \quad (11)$$

We note that only the case where $x_k \neq x^*$ is interesting, since $x_k = x^*$ implies that p_k and r_k are zero. The proof is by induction. First note that for $k = 0$, equation (11) holds as $p_0 = -r_0$. Assuming that equation (11) is true for k we show that the same holds for $k + 1$. Beginning with the right equality, we first show that

$$\text{span}\{r_0, r_1, ..., r_k, r_{k+1}\} \subset \text{span}\{r_0, Ar_0, ..., A^k r_0, A^{k+1} r_0\}.$$

Note that by the induction hypothesis $r_k, p_k \in \text{span}\{r_0, Ar_0, ..., A^k r_0\}$. Because of this, there exist $\gamma_0, ..., \gamma_k \in \mathbb{R}$ such that

$$p_k = \gamma_0 r_0 + \gamma_1 A r_0 + ... + \gamma_k A^k r_0,$$

and therefore

$$Ap_k = \gamma_0 A r_0 + \gamma_1 A^2 r_0 + ... + \gamma_k A^{k+1} r_0.$$

Since $\alpha_k \in \mathbb{R}$, we conclude by equation (4) that r_{k+1} is also a linear combination of $r_0, Ar_0, ..., A^{k+1} r_0$. Combining this with the induction hypothesis, every element in $\text{span}\{r_0, ..., r_{k+1}\}$ can be written as a linear combination of $r_0, Ar_0, ..., A^{k+1} r_0$, so we have

$$\text{span}\{r_0, r_1, ..., r_k, r_{k+1}\} \subset \text{span}\{r_0, Ar_0, ..., A^k r_0, A^{k+1} r_0\}.$$

For the reverse inclusion we note that by a similar reasoning as above we have

$$\begin{aligned} A^{k+1} r_0 &= A(A^k r_0) \\ &\in \text{span}\{Ap_0, Ap_1, ..., Ap_k\}, && \text{By the induction hyp.;} \\ &= \text{span}\{\frac{r_1 - r_0}{\alpha_0}, \frac{r_2 - r_1}{\alpha_1}, ..., \frac{r_{k+1} - r_k}{\alpha_k}\}, && \text{By equation(4);} \\ &\subset \text{span}\{r_0, r_1, ..., r_{k+1}\}. \end{aligned}$$

Therefore we have

$$\text{span}\{r_0, r_1, ..., r_k, r_{k+1}\} = \text{span}\{r_0, Ar_0, ..., A^k r_0, A^{k+1} r_0\}. \quad (12)$$

To prove the left equality of equation (11) for $k+1$, we note that

$$\begin{aligned}
&\text{span}\{p_0, p_1, ..., p_k, p_{k+1}\} \\
&= \text{span}\{p_0, p_1, ..., p_k, r_{k+1}\}, && \text{by equation (9);} \\
&= \text{span}\{r_0, r_1, ..., r_k, r_{k+1}\}, && \text{by the induction hypothesis;}
\end{aligned}$$

proving equation (11).

We can now show that the conjugate gradient method convergences in at most n steps by using the precious result.

Theorem 2.2. The Conjugate Gradient method, given by algorithm 2, generates conjugate direction vectors and converges in at most n steps.

Proof. The proof by induction. First note that by construction of β_k, $p_{k+1}Ap_k = 0$, so in particular for $k = 0$ we have $p_1 Ap_0 = 0$. Now assume that $p_k Ap_i = 0$ for all $i = 0, 1, ..., k-1$ and prove that this holds for $k+1$.

We note that β_k was constructed such that $p_{k+1}Ap_k = 0$, so we only need to consider $p_{k+1}Ap_i$ for $i = 0, 1, ..., k-1$. By the definition of p_{k+1} in equation (9) we then have that

$$p_{k+1}^T Ap_i = (-r_{k+1} + \beta_{k+1}p_k)^T Ap_i \quad = -r_{k+1}^T Ap_i + \beta_{k+1}p_k^T Ap_i$$

We note that by equation (11), Ap_i is a linear combination of $p_0, p_1, ..., p_{i+1}$. It follows that $r_{k+1}^T Ap_i = 0$ by equation (6) and that $p_k^T Ap_i = 0$ by the induction hypothesis. Therefore, $p_{k+1}^T Ap_i = 0$ for $i = 0, 1, ..., k$. Because the directions are conjugate, it follows by Lemma 2.1 that algorithm 2 terminates in at most n steps. □

2.1.4 Convergence in r steps*

This section focuses on showing that CG converges in at most r steps, where r is the number of distinct eigenvalues of A in equation (1). This result shows that CG has the potential to converge very fast, increasing the interest in extending it to methods for non-linear objectives. It can be skipped if one is only interested in methods for non-linear objective functions.

We first introducing the distance induced by the norm[1]:

$$\|z\|_{\bar{A}}^2 := z^T \bar{A} z,$$

[1]Note that $\langle z_1, z_2 \rangle := z_1^T \bar{A} z_2$ defines the inner product that induces the norm, showing that $\|\cdot\|_{\bar{A}}$ is indeed a norm.

where $\bar{A} \in \mathbb{R}^{n\times n}$ is a symmetrical positive definite matrix. Using this norm we will show that

$$\|x_k - x^*\|_A^2 \leq \max_{1\leq i\leq n} \left(1 + \lambda_i P_k(\lambda_i)\right)^2 \|x_0 - x^*\|_A^2,$$

for the iterates of algorithm 2, for all polynomials P_k of degree k. This will then be used to prove Theorem 2.3 which demonstrates the faster convergence. We begin by noticing that for all $x \in \mathbb{R}^n$ we have that

$$\begin{aligned}
\frac{1}{2}\|x - x^*\|_A^2 &= \frac{1}{2}(x - x^*)^T A(x - x^*) \\
&= \frac{1}{2}x^T Ax - x^T Ax^* + \frac{1}{2}x^{*^T} Ax^* \\
&= (\frac{1}{2}x^T Ax - x^T b) - (\frac{1}{2}x^{*^T} Ax^* - x^{*^T} Ax^*) \\
&= (\frac{1}{2}x^T Ax - b^T x) + (\frac{1}{2}x^{*^T} Ax^* - b^T x^*) \\
&= f(x) - f(x^*).
\end{aligned}$$

Because by Lemma 2.1 the the iterate x_{k+1} of algorithm 2 minimizes f over the set $L_k := \{x + x_0 | x \in \text{span}\{p_0, p_1, ..., p_k\}\}$ and $f(x^*)$ is a constant, they also minimize $\|x - x^*\|_A^2$ over this same set. In other words

$$\min_{x\in L_k} \|x - x^*\|_A^2 = \|x_{k+1} - x^*\|_A^2.$$

We will now rewrite x_k to the more convenient form

$$\begin{aligned}
x_{k+1} &= x_0 + \alpha_0 p_0 + ... + \alpha_k p_k \\
&= x_0 + \gamma_0 r_0 + ... + \gamma_k A^k r_0, \qquad \text{by equation (11)},
\end{aligned}$$

for some $\gamma_0, ..., \gamma_k \in \mathbb{R}$. We can rewrite this further by defining a polynomial $P_k^* : \mathbb{R}^{n\times n} \to \mathbb{R}^{n\times n}$ of degree k as

$$P_k^*(A) = \gamma_0 I + \gamma_1 A + ... + \gamma_k A^k.$$

Since $x_{k+1} = x_0 + P_k^*(A)r_0$, we have

$$\begin{aligned}
\min_{x\in L_k} \|x - x^*\|_A^2 &= \|x_{k+1} - x^*\|_A^2 \\
&= \|x_0 - x^* + P_k^*(A)r_0\|_A^2 \\
&= \min_{P_k} \|x_0 - x^* + P_k(A)r_0\|_A^2,
\end{aligned}$$

as $P_k \neq P_k^*$ implies $f(x_0 + P_k(A)r_0) > f(x_k)$, resulting in a greater value of the norm. Next we will rewrite the expression $\|x_0 - x^* + P_k(A)r_0\|_A^2$. Using the notation λ_i and v_i for the eigenvalues and the corresponding eigenvectors of A,

we have by Lemma 5.1 that

$$\begin{aligned}\|x\|_A^2 &= x^T A x \\ &= x^T \sum_{i=1}^n \lambda_i v_i v_i^T x \\ &= \sum_{i=1}^n \lambda_i (x^T v_i)(v_i^T x) \\ &= \sum_{i=1}^n \lambda_i (v_i^T x)^2, \qquad \text{for all } x \in \mathbb{R}^n.\end{aligned} \tag{13}$$

We also note that because the orthonormal eigenvectors of A span $\mathbb{R}^n$, we can write

$$x_0 - x^* = \sum_{i=1}^n \xi_i v_i, \tag{14}$$

for some $\xi_i \in \mathbb{R}$. Therefore we have by using the orthonormality of the eigenvectors that

$$\begin{aligned}\|x_0 - x^*\|_A^2 &= \|\sum_{j=1}^n \xi_j v_j\|_A^2 \\ &= \sum_{i=1}^n \lambda_i \big(v_i^T \sum_{j=1}^n \xi_j v_j\big)^2 \\ &= \sum_{i=1}^n \lambda_i \big(\sum_{j=1}^n \xi_i (v_i^T v_j)\big)^2 \\ &= \sum_{i=1}^n \lambda_i \xi_i^2.\end{aligned} \tag{15}$$

Using the same notation we also find

$$\begin{aligned} x_0 - x^* + P_k(A)(r_0) &= (x_0 - x^*) + P_k(A)\big(A(x_0 - x^*)\big) \\ &= \big(I + P_k(A)A\big)(x_0 - x^*) \\ &= \big(I + P_k(A)A\big)\sum_{i=1}^{n} \xi_i v_i \\ &= \sum_{i=1}^{n} \Big(I + \big(P_k(A)\big)A\Big)\xi_i v_i \\ &= \sum_{i=1}^{n} \Big(I + \big(\gamma_0 I + ... + \gamma_k A^k\big)A\Big)\xi_i v_i \\ &= \sum_{i=1}^{n} \Big(1 + \big(\gamma_0 + ... + \gamma_k \lambda_i^k\big)\lambda_i\Big)\xi_i v_i \\ &= \sum_{i=1}^{n} \Big(1 + \big(P_k(\lambda_i)\big)\lambda_i\Big)\xi_i v_i, \qquad \text{for all } P_k. \end{aligned} \tag{16}$$

By combining the equations (13), (14), (15) and (16) and by again using the orthonormality of the eigenvectors of A we now find

$$\begin{aligned} \|x_{k+1} - x^*\|_A^2 &= \min_{P_k} \|x_0 - x^* + P_k(A)r_0\|_A^2 \\ &= \min_{P_k} \sum_{i=1}^{n} \lambda_i \Big(v_i^T \sum_{j=1}^{n} \big(1 + P_k(\lambda_j)\lambda_j\big)\xi_j v_j\Big)^2 \\ &= \min_{P_k} \sum_{i=1}^{n} \lambda_i \Big(\sum_{j=1}^{n} \big(1 + P_k(\lambda_j)\lambda_j\big)\xi_j (v_i^T v_j)\Big)^2 \\ &= \min_{P_k} \sum_{i=1}^{n} \lambda_i \big(1 + P_k(\lambda_i)\lambda_i\big)^2 \xi_i^2 \\ &\leq \min_{P_k} \max_{1\leq j\leq n} \big(1 + P_k(\lambda_j)\lambda_j\big)^2 \sum_{i=1}^{n} \lambda_i \xi_i^2 \\ &= \min_{P_k} \max_{1\leq j\leq n} \big(1 + P_k(\lambda_j)\lambda_j\big)^2 \|x_0 - x^*\|_A^2 \end{aligned}$$

Because this inequality holds for the minimizer of the right hand side, it holds for all polynomials P_k. We will now use this to prove the following result.

Theorem 2.3. If A in equation (1) has r distinct eigenvalues, then algorithm 2 will find the solution in at most r iterations.

Proof. We will construct a polynomial such that

$$0 \leq \|x_r - x^*\|_A^2 \leq \max_{1\leq i\leq n} \big(1 + \lambda_i P_{r-1}(\lambda_i)\big)^2 \|x_0 - x^*\|_A^2 = 0.$$

Because the eigenvalues of the positive definite matrix A take on the r distinct values $0 < \tau_1, \tau_2, ..., \tau_r$ we can define the polynomial

$$Q_r(x) := (-1)^r \frac{(x-\tau_1)(x-\tau_2)...(x-\tau_r)}{\tau_1\tau_2...\tau_r}.$$

Now note that $Q_r(\tau_i) = 0$ and that $Q_r(0) = 1$. Therefore $Q_r(x) - 1$ can be written as

$$Q_r(x) - 1 = c_1x + c_2x^2 + ... + c_rx^r$$

for some constants $c_i \in \mathbb{R}$. This means that we can define

$$P_{r-1} := \frac{Q_r(x) - 1}{x} = c_1 + c_2x^1 + ... + c_rx^{r-1}.$$

This yields

$$\begin{aligned}
0 &\leq \|x_r - x^*\|_A^2 \\
&\leq \max_{1\leq i\leq n} \big(1 + \lambda_i P_{r-1}(\lambda_i)\big)^2 \|x_0 - x^*\|_A^2 \\
&= \max_{1\leq i\leq n} \big(1 + \lambda_i \frac{Q_r(\lambda_i) - 1}{\lambda_i}\big)^2 \|x_0 - x^*\|_A^2 \\
&= \max_{1\leq i\leq n} Q_r(\lambda_i)^2 \|x_0 - x^*\|_A^2 \\
&= 0, \qquad \text{since } Q_r(\lambda_i) = 0,
\end{aligned}$$

thus proving that $x_r = x^*$ □

2.2 Non-linear conjugate gradients

Now that we have shown that the conjugate gradient method can potentially converge quite fast, a natural thing to do is to extend the method to general non-linear, but still continuously differentiable functions $\phi : \mathbb{R}^n \to \mathbb{R}$. To do so, two adjustments need to be made to the method. That is, methods of finding the search directions and the step sizes are required, as these previously depended on the exact formulation of f. The Fletcher-Reeves method (Fletcher-Reeves or FR) proposes a method for both of these issues.

To find the new search direction, FR simply replaces the gradient of f by the gradient of the new function ϕ. However, it does so in a slightly rewritten the version of CG, where for convenience $\{\phi_k, f_k, \nabla\phi_k, \nabla f_k\}$ will be used to denote $\{\phi(x_k), f(x_k), \nabla\phi(x_k), \nabla f(x_k)\}$. The algorithm will be rewritten using that fact that

$$r_j^T r_i = 0, \qquad \text{for } i \neq j. \tag{17}$$

This will be shown using Theorem 2.2. As the generated directions of algorithm 2 are conjugate, we have by equation (6) that

$$\begin{aligned} 0 &= r_k^T p_i \\ &= r_k^T(-r_i + \beta_i p_{i-1}) \\ &= -r_k^T r_i + \beta_i r_k^T p_{i-1}, \qquad \text{for } i = 1, 2, ..., k-1. \end{aligned}$$

Therefore, $r_k^T r_i = \beta_i (r_k^T p_{i-1}) = 0$. For $i = 0$ we have by the same equation that $r_k^T r_0 = -r_k^T p_0 = 0$.

To rewrite algorithm 2, first note that by equation (4) we have that $Ap_k = \frac{r_{k+1} - r_k}{\alpha_k}$. Combining this with the equations (6) and (17) yields

$$\begin{aligned} \beta_{k+1} &= \frac{r_{k+1}^T A p_k}{p_k^T A p_k} \\ &= \frac{r_{k+1}^T (r_{k+1} - r_k)\frac{1}{\alpha_k}}{p_k^T (r_{k+1} - r_k)\frac{1}{\alpha_k}} \\ &= -\frac{r_{k+1}^T r_{k+1}}{p_k^T r_k} \end{aligned}$$

For $k = 0$ this gives

$$\beta_1 = \frac{r_1^T r_1}{-p_0^T r_0} = \frac{\nabla f_1^T \nabla f_1}{\nabla f_0^T \nabla f_0}.$$

For other k we find by equation (6) that

$$\beta_{k+1} = -\frac{r_{k+1}^T r_{k+1}}{(-r_k + \beta_k p_{k-1})^T r_k} = \frac{r_{k+1}^T r_{k+1}}{r_k^T r_k} = \frac{\nabla f_{k+1}^T \nabla f_{k+1}}{\nabla f_k^T \nabla f_k}.$$

This is sufficient for defining FR, however, as a curiosity, we note that using the same equations we can also rewrite α_k to yield a more efficient algorithm. Namely, for $k = 0$ we have

$$\alpha_0 = \frac{r_0^T p_0}{p_0^T A p_0} = -\frac{\nabla f_0^T \nabla f_0}{p_0^T A p_0},$$

and for other k we have

$$\alpha_k = \frac{r_k^T p_k}{p_k^T A p_k} = \frac{r_k^T(-r_k + \beta_k p_{k-1}))}{p_k^T A p_k} = -\frac{\nabla f_k^T \nabla f_k}{p_k^T A p_k},$$

resulting in

$$\begin{aligned} \alpha_k &= \frac{\nabla f_k^T \nabla f_k}{p_k^T A p_k}; \\ \beta_{k+1} &= \frac{\nabla f_{k+1}^T \nabla f_{k+1}}{\nabla f_k^T \nabla f_k}, \qquad \text{for all } k. \end{aligned}$$

This results in the new $\beta_k^{FR} = \frac{\nabla \phi_{k+1}^T \nabla \phi_{k+1}}{\nabla \phi_k^T \nabla \phi_k}$.

2.2.1 Line searches

To find the new step size FR uses a different scheme. It uses a *line search* to determine a sufficiently decreased next iterate. In a line search one searches for a step size α_k in the direction p_k until a step size has been found that satisfies a stop condition. At that new point $x_{k+1} = x_k + \alpha_k p_k$, the objective function ϕ is deemed to have decreased sufficiently and to be suficiently close to the local minimum. The *strong Wolfe Conditions* are commonly used stop conditions given by

$$\phi(x_k + \alpha_k p_k) \leq \phi(x_k) + c_1 \alpha_k \nabla\phi(x_k)^T p_k \tag{18a}$$

$$|\nabla\phi(x_k + \alpha_k p_k)^T p_k| \leq c_2 |\nabla\phi(x_k)^T p_k|, \qquad \text{for } 0 < c_1 < c_2 < 1. \tag{18b}$$

The first of these equations ensures that $\{\phi(x_i)\}$ is a monotonically decreasing sequence. It also asserts that large steps require a greater drop in the objective function to be accepted. The second equation ensures that x_{k+1} is closer to a local minimum than x_k along the direction p_k. To be able to stability use the strong Wolfe conditions as a stop condition for the line search in FR, we must ensure that it is always possible to find such a step size. This a guarantee will be given in the following Lemma.

Lemma 2.4. Let $\phi : \mathbb{R}^n \to \mathbb{R}$ be continuously differentiable and let p_k be a descent direction. If ϕ is bounded below along the line $\{x_k + \alpha p_k | \alpha > 0\}$, then there exists at least one interval of step sizes α that satisfy the strong Wolfe conditions formulated in the equations (18a) and (18b).

Proof. We begin by naming the RHS and the LHS of equation (18a) as

$$g_k(\alpha) := \phi(x_k + \alpha p_k);$$
$$l_k(\alpha) := \phi(x_k) + c_1 \alpha \nabla\phi(x_k)^T p_k,$$

and note that

$$g_k'(0) = \nabla\phi(x_k)^T p_k < c_1 \nabla\phi(x_k)^T p_k = l_k'(0), \tag{19}$$

since $0 < c_1 < 1$. Also note that $g_k(0) = \phi(x_k) = l_k(0)$. Therefore, g_k lies below l_k near $\alpha = 0$.

Because $l_k(\alpha) \to -\infty$ when $\alpha \to \infty$ due to p_k being a descent direction and because g_k is bounded below, we have by the intermediate value theorem[2] that l_k and g_k will intersect at least once. Choose $\tilde{\alpha}_k$ to be the first value of α for which this intersection occurs. Explicitly this gives

$$\phi(x_k + \tilde{\alpha} p_k) = \phi(x_k) + \tilde{\alpha} c_1 \nabla\phi(x_k)^T p_k. \tag{20}$$

[2] using the continuity of g_k and l_k

We now have that equation (18a) holds for the all $\alpha < \tilde{\alpha}_k$. To find an interval that satisfies the second equation as well, we use the mean value theorem to find that a $\xi_k \in (0, \tilde{\alpha}_k)$ exists such that

$$\begin{aligned} g_k(\tilde{\alpha}_k) - g_k(0) &= g_k(\xi_k)(\tilde{\alpha}_k - 0); \\ \phi(x_k + \tilde{\alpha}_k p_k) - \phi(x_k) &= \tilde{\alpha}_k \nabla\phi(x_k + \xi_k p_k)^T p_k. \end{aligned}$$

Together with equation (20) and the fact that $0 < c_1 < c_2$ we then have that

$$|\nabla\phi(x_k + \xi_k p_k)^T p_k| = c_1 |\nabla\phi(x_k)^T p_k| < c_2 |\nabla\phi(x_k)^T p_k|$$

Because ϕ is continuously differentiable we then have that there exists an interval around ξ_k such that the inequality above, which is equivalent to equation (18b), holds for all elements in that interval. Therefore, there exists an interval such that the strong Wolfe conditions hold. □

Given an objective function that satisfies the conditions above, an algorithm that produces descent directions and a line search algorithm that finds and picks an α_k, we can now guarantee that the Wolfe conditions can be used in practise.

Note that in the design of the objective function it can be asserted that ϕ is bounded below. Because the first search direction p_0 of FR is the negative gradient $-\nabla\phi_0$, it follows from Lemma 2.4 that the strong Wolfe conditions can be used to produce the next iterate. To be able to use Lemma 2.4 to show that we can use the strong Wolfe conditions for all iterates, we have to show that FR does indeed produce descent directions whenever the previous iterate was produced using the strong Wolfe conditions. We will show this for $0 < c_1 < c_2 < \frac{1}{2}$, in Lemma 2.5.

Lemma 2.5. If the iterate, $x_k = x_{k-1} + \alpha_{k-1} p_{k-1}$, in FR is produced using a descent direction p_{k-1} and an α_{k-1} that satisfies the strong Wolfe conditions with $0 < c_2 < \frac{1}{2}$, then the next search direction p_k is a descent direction that satisfies

$$-\frac{1}{1 - c_2} \leq \frac{\nabla\phi_k^T p_k}{\|\nabla\phi_k\|^2} \leq \frac{2c_2 - 1}{1 - c_2}. \tag{21}$$

Proof. The proof is by induction. We first introduce the function $t : [0, \frac{1}{2}] \to \mathbb{R}$ given by $t(\xi) := \frac{2\xi - 1}{1 - \xi}$. We note that t is continuous on its domain and that it is injective since $t(a) = t(b)$ gives $a = b$. Because also $-1 = t(0) < t(\frac{1}{2}) = 0$, t is strictly monotonically increasing.

We will now consider the case $k = 0$. Because $c_2 \in (0, \frac{1}{2})$ we have that $\frac{\nabla\phi_0^T p_0}{\|\nabla\phi_0\|^2} = -1 = t(0) < t(c_2)$ and therefore $-\frac{1}{1-c_2} < \frac{\nabla\phi_0^T p_0}{\|\nabla\phi_0\|^2} < \frac{2c_2 - 1}{1 - c_2}$. Next we will assume

that equation (21) holds for k and show that the same holds for $k+1$. To prove the RHS of equation (21) for $k+1$ we write

$$\begin{aligned}
\frac{\nabla\phi_{k+1}^T p_{k+1}}{\|\nabla\phi_{k+1}\|^2} &= \frac{\nabla\phi_{k+1}^T(-\nabla\phi_{k+1}+\beta_{k+1}^{FR}p_k)}{\|\nabla\phi_{k+1}\|^2};\\
&= \frac{\nabla\phi_{k+1}^T(\frac{\nabla\phi_{k+1}^T\nabla\phi_{k+1}}{\nabla\phi_k^T\nabla\phi_k}p_k)}{\|\nabla\phi_{k+1}\|^2} - 1;\\
&= \frac{\nabla\phi_{k+1}^T p_k}{\|\nabla\phi_k\|^2} - 1;\\
&\leq \frac{|\nabla\phi_{k+1}^T p_k|}{\|\nabla\phi_k\|^2} - 1;\\
&\leq c_2\frac{|\nabla\phi_k^T p_k|}{\|\nabla\phi_k\|^2} - 1, && \text{by equation (18b);}\\
&= -c_2\frac{\nabla\phi_k^T p_k}{\|\nabla\phi_k\|^2} - 1, && \text{as } \nabla\phi_k^T p_k < 0;\\
&\leq \frac{c_2}{1-c_2} - 1, && \text{by the LHS of eq. (21);}\\
&= \frac{2c_2-1}{1-c_2}.
\end{aligned}$$

Similarly, to prove the LHS of equation (21) for $k+1$ we now write

$$\begin{aligned}
\frac{\nabla\phi_{k+1}^T p_{k+1}}{\|\nabla\phi_{k+1}\|^2} &= \frac{\nabla\phi_{k+1}^T p_k}{\|\nabla\phi_k\|^2} - 1;\\
&\geq -c_2\frac{|\nabla\phi_k^T p_k|}{\|\nabla\phi_k\|^2} - 1, && \text{by equation (18b);}\\
&= c_2\frac{\nabla\phi_k^T p_k}{\|\nabla\phi_k\|^2} - 1, && \text{as } \nabla\phi_k^T p_k < 0;\\
&\geq -\frac{c_2}{1-c_2} - 1, && \text{by the LHS of eq. (21);}\\
&= -\frac{1}{1-c_2}.
\end{aligned}$$

Because of this, equation (21) holds for all k. Since $c_2 \in (0, \frac{1}{2})$, we have that $\frac{\nabla\phi_k^T p_k}{\|\nabla\phi_k\|^2} \leq \frac{2c_2-1}{1-c_2} = t(c_2) < t(\frac{1}{2}) = 0$ as t strictly monotonically increases on its domain. Therefore p_k is a descent direction for all k. □

2.2.2 Global Convergence

Now we know that the strong Wolfe conditions can be used reliably as a stop criterium for the line search, FR can be formulated in algorithm 3.

Algorithm 3 Fletcher Reeves

1: Given x_0, c_1, c_2;
2: Evaluate $\nabla\phi_0 = \nabla\phi(x_0)$
3: Set $p_0 \leftarrow -\nabla\phi_0$, $k \leftarrow 0$;
4: **while** $\nabla\phi_k \neq 0$ **do**

Determine α_k

$x_{k+1} \leftarrow x_k + \alpha_k p_k$

Evaluate $\nabla\phi_{k+1}$

$$\beta_{k+1} \leftarrow \frac{\nabla\phi_{k+1}^T \nabla\phi_{k+1}}{\nabla\phi_k^T \nabla\phi_k} \tag{22a}$$

$$p_{k+1} \leftarrow -\nabla\phi_{k+1} + \beta_{k+1} p_k \tag{22b}$$

$k \leftarrow k + 1$

For this algorithm we will first demonstrate its global convergence through a mathematical proof to illustrate that non-linear conjugate gradient methods can have global convergence properties. We will then show that related non-linear conjugate gradient algorithms outperform FR, demonstrated through empirically determined convergence rates. These will be used demonstrated that

1. the earlier determined convergence in much less than n steps can sometimes carry over to non-linear conjugate gradient methods;
2. mathmatically estimating the convergence rate of FR might not be very useful.

The main tool that will be used to prove global convergence is Zoutendijk's theorem, shown in Lemma 2.6.

Lemma 2.6. Suppose that an objective function $\phi : \mathbb{R}^n \to \mathbb{R}$ is Lipschitz continuously differentiable on an open set $\mathcal{N} \supset \mathcal{L} := \{x | \phi(x) \leq \phi_0\}$ and that ϕ bounded below on $\mathcal{L}$. In other words, there exists a constant L such that

$$\|\nabla\phi(x) - \nabla\phi(y)\| \leq L\|x - y\|, \qquad \text{for all } x, y \in \mathcal{N}. \tag{23}$$

For such an objective function, consider an iteration of the form $x_{k+1} = x_k + \alpha_k p_k$ where p_k is a descent direction and α_k satisfies the Wolfe conditions formulated as

$$\phi_{k+1} \leq \phi_k + c_1 \alpha_k \nabla\phi_k^T p_k; \tag{24a}$$

$$\nabla\phi_{k+1}^T p_k \geq c_2 \nabla\phi_k^T p_k, \qquad \text{for } 0 < c_1 < c_2 < 1. \tag{24b}$$

Then

$$\sum_{k\geq 0}^{\infty} \cos^2(\theta_k)\|\nabla\phi_k\|^2 < \infty,$$

where θ_k is the angle between p_k and the steepest descent direction $-\nabla\phi_k$, defined by

$$\cos(\theta_k) = \frac{-\nabla\phi_k^T p_k}{\|\nabla\phi_k\|\|p_k\|}. \tag{25}$$

Proof. We begin by noting that because of the Lipschitz continuity of $\nabla\phi$ and the definition of x_{k+1} we have that

$$\|\nabla\phi_{k+1} - \nabla\phi_k\| \leq L\|x_{k+1} - x_k\| = L\alpha_k\|p_k\|, \qquad \text{as } \alpha_k \geq 0,$$

which gives

$$\begin{aligned}(\nabla\phi_{k+1} - \nabla\phi_k)^T p_k &\leq |(\nabla\phi_{k+1} - \nabla\phi_k)^T p_k| \\ &\leq \|\nabla\phi_{k+1} - \nabla\phi_k\|\|p_k\| \qquad \text{by Cauchy-Schwarz} \\ &\leq L\alpha_k\|p_k\|^2.\end{aligned}$$

On the other hand, by subtracting $\nabla\phi_k^T p_k$ from the second Wolfe condition in (24b) we find that

$$(c_2 - 1)\nabla\phi_k^T p_k \leq (\nabla\phi_{k+1}^T - \nabla\phi_k^T)p_k.$$

Combining the two equations above yields

$$\frac{(c_2 - 1)\nabla\phi_k^T p_k}{L\|p_k\|^2} \leq \frac{(\nabla\phi_{k+1} - \nabla\phi_k)^T p_k}{L\|p_k\|^2} \leq \alpha_k.$$

Combining this with the first Wolfe conditions (24a) and noting that $\nabla\phi_k^T p_k < 0$ we get

$$\phi_{k+1} \leq \phi_k + c_1\frac{c_2 - 1}{L\|p_k\|^2}(\nabla\phi_k^T p_k)^2,$$

which, by the definition of the angle between p_k and $-\nabla\phi_k$ in equation (25), results in

$$\phi_{k+1} \leq \phi_k - c_3\cos^2(\theta_k)\|\nabla\phi_k^T\|^2,$$

where $c_3 = c_1\frac{1-c_2}{L} > 0$. Since this holds for all k, we can repeatedly apply this inequality to obtain

$$\phi_{k+1} \leq \phi_0 - c_3\sum_{i=0}^{k}\cos^2(\theta_i)\|\nabla\phi_i^T\|^2,$$

which is identical to

$$\sum_{i=0}^{k} \cos^2(\theta_i) \|\nabla \phi_i^T\|^2 \leq \frac{\phi_0 - \phi_{k+1}}{c_3}.$$

From this last expression we see that the RHS is an upper bound for the LFH. Since ϕ is bounded below on $\mathcal{L}$ and the Wolfe conditions ensure that $\{\phi_i\}$ is monotonically decreasing, the RHS is less than a constant for all k. Therefore we have that

$$\sum_{k\geq 0}^{\infty} \cos^2(\theta_k) \|\nabla \phi_k\|^2 < \infty.$$

□

This will now be used to show that FR converges globally by showing that for the iterates of algorithm 3 we have that

$$\lim_{k\to\infty} \inf \|\nabla \phi_k\| = 0,$$

if

1. the level set $\mathcal{L} := \{x | \phi(x) \leq \phi_0\}$ is bounded;
2. on some open neighborhood $\mathcal{N}$ of $\mathcal{L}$, the objective function ϕ is Lipschitz continuously differentiable;
3. the line search in algorithm 3 satisfies the stong Wolfe conditions, with $0 < c_1 < c_2 < \frac{1}{2}$.

The proof is by contradiction. Assume that $\lim_{k\to\infty} \inf \|\nabla \phi_k\| \neq 0$. Then there exits a lower bound μ such that

$$\|\nabla \phi_k\| > \mu > 0, \qquad \text{for all k large enough.} \tag{26}$$

Note that by the assumptions 1 and 2 we have that

$$\begin{aligned} \|\nabla \phi(x)\| &\leq \|\nabla \phi(x) - \nabla \phi(y)\| + \|\nabla \phi(y)\| \\ &< L\|x - y\| + \|\nabla \phi(y)\| \\ &< M + \|\nabla \phi(y)\| \\ &=: T, \end{aligned}$$

for some $L, M, T \in \mathbb{R}$, $y \in \mathcal{L}$ and for all $x \in \mathcal{L}$. Because $\|\nabla \phi(x)\|$ is bounded we know that by the mean value theorem $\|\phi(x)\|$ is also bounded. Because of this and the fact that the strong Wolfe conditions imply the Wolfe conditions (note that $\nabla \phi_k pk < 0$) we are able to use Zoutendijk's theorem.

Rewriting the result of Lemma 2.5 gives us

$$\frac{1-2c_2}{1-c_2}\frac{\|\nabla\phi_k\|}{\|p_k\|} \leq \frac{-\nabla\phi_k^T p_k}{\|\nabla\phi_k\|\|p_k\|} \leq \frac{\|\nabla\phi_k\|}{\|p_k\|}\frac{1}{1-c_2},$$

where $\frac{1-2c_2}{1-c_2} > 0$, since $c_2 \in (0, \frac{1}{2})$. Combining this with $\cos(\theta_k) = \frac{-\nabla\phi_k^T p_k}{\|\nabla\phi_k\|\|p_k\|}$, the assumption in equation (26) and the result of Zoutendijk's theorem we get

$$\mu^4(\frac{1-2c_2}{1-c_2})^2\sum_{k=0}^{\infty}\frac{1}{\|p_k\|^2} < (\frac{1-2c_2}{1-c_2})^2\sum_{k=0}^{\infty}\frac{\|\nabla\phi_k\|^4}{\|p_k\|^2} \leq \sum_{k=0}^{\infty}\cos^2(\theta_k)\|\nabla\phi_k\|^2 < \infty.$$

Therefore we find that

$$\sum_{k=0}^{\infty}\frac{1}{\|p_k\|^2} < \infty. \tag{27}$$

Rewriting the result of Lemma 2.5 again, we get

$$c_2|\nabla\phi_{k-1}^T p_{k-1}| = -c_2\nabla\phi_{k-1}^T p_{k-1} \leq \|\nabla\phi_{k-1}\|^2\frac{c_2}{1-c_2}.$$

Combining this with the second strong Wolfe condition, equation (18b), gives

$$|\nabla\phi_k^T p_{k-1}| \leq \|\nabla\phi_{k-1}\|^2\frac{c_2}{1-c_2}. \tag{28}$$

Using this to rewrite $\|p_k\|^2$, we get

$$\begin{aligned}
\|p_k\|^2 &= (-\nabla\phi_k + \beta_k^{FR}p_{k-1})^T(-\nabla\phi_k + \beta_k^{FR}p_{k-1}) && \text{by eq. (22b)}\\
&\leq \|\nabla\phi_k\|^2 + (\beta_k^{FR})^2\|p_{k-1}\|^2 + 2\beta_k^{FR}|\nabla\phi_k^T p_{k-1}|\\
&\leq \|\nabla\phi_k\|^2 + (\beta_k^{FR})^2\|p_{k-1}\|^2 + 2\beta_k^{FR}\|\nabla\phi_{k-1}\|^2\frac{c_2}{1-c_2} && \text{by eq. (28)}\\
&= \|\nabla\phi_k\|^2 + (\beta_k^{FR})^2\|p_{k-1}\|^2 + \|\nabla\phi_k\|^2\frac{2c_2}{1-c_2} && \text{by eq. (22a)}\\
&= \frac{1+c_2}{1-c_2}\|\nabla\phi_k\|^2 + (\beta_k^{FR})^2\|p_{k-1}\|^2.
\end{aligned}$$

Using this expression repeatedly, with $c_3 = \frac{1+c_2}{1-c_2} > 0$, we get

$$\begin{aligned}
\|p_k\|^2 &\leq c_3\|\nabla\phi_k\|^2 + (\beta_k^{FR})^2\|p_{k-1}\|^2\\
&\leq c_3\|\nabla\phi_k\|^2 + (\beta_k^{FR})^2(\|c\nabla\phi_{k-1}\|^2 + (\beta_{k-1}^{FR})^2\|p_{k-2}\|^2)\\
&= c_3\frac{\|\nabla\phi_k\|^4}{\|\nabla\phi_k\|^2} + c\frac{\|\nabla\phi_k\|^4}{\|\nabla\phi_{k-1}\|^2} + (\beta_k^{FR})^2(\beta_{k-1}^{FR})^2\|p_{k-2}\|^2\\
&\leq c_3\|\nabla\phi_k\|^4\sum_{i=1}^{k}\frac{1}{\|\nabla\phi_i\|^2} + \frac{\|\nabla\phi_k\|^4}{\|\nabla\phi_0\|^4}\|p_0\|^2\\
&= c_3\|\nabla\phi_k\|^4\sum_{i=0}^{k}\frac{1}{\|\nabla\phi_i\|^2}.
\end{aligned}$$

Combining this with $0 < \mu < \|\nabla\phi_k\|$ and $\|\nabla\phi(x)\| < T$ for all $x \in \mathcal{L}$, we find

$$\|p_k\|^2 < c_3 \frac{(k+1)T^4}{\mu^2}.$$

Together with equation (27) this results in

$$\frac{\mu^2}{T^4 c_3} \sum_{k=0}^{\infty} \frac{1}{k+1} < \sum_{k=0}^{\infty} \frac{1}{\|p_k\|^2} < \infty. \tag{29}$$

However, this cannot be true as $\sum_{k=1}^{\infty} \frac{1}{k} \not< \infty$. Therefore, the assumption in equation (26) cannot be true, proving the claim that $\lim_{k\to\infty} \inf \|\nabla\phi_k\| = 0$.

2.2.3 Convergence speed

Now that global convergence of FR has been shown, we will give an impression of its convergence speed compared to other conjugate gradient methods. This will be done by comparing FR to various other non-linear conjugate gradient methods both on problems of around 1000 dimensions, as well as problems of two dimensions, which can be easily visualized.

The first alternative non-linear conjugate gradient method that will be used is Polak-Ribiere (PR), which differs from FR in using

$$\beta_k^{PR} = \frac{\nabla\phi_{k+1}^T(\nabla\phi_{k+1} - \nabla\phi_k)}{\nabla\phi_k^T\nabla\phi_k}.$$

However, this method is not guaranteed to converge globally, but such guarantees can be produced for alternatives such as

$$\beta_k^{PR+} = \max(\beta_k^{PR}, 0)$$

and

$$\beta_k^{FR-PR} = \begin{cases} -\beta_k^{FR} & \text{if } \beta_k^{PR} < -\beta_k^{FR}; \\ \beta_k^{PR} & \text{if } |\beta_k^{PR}| < \beta_k^{FR}; \\ \beta_k^{FR} & \text{if } \beta_k^{PR} > \beta_k^{FR}, \end{cases}$$

which we will not go into further. These four methods have been compared in [Gilbert and Nocedal, 1992] on various problems. An overview of convergence speeds is given in table 1, equivalent to table 3 in the same publication. The table reports for various problems P, of size N, the number of iterations and the number of function and gradient evaluations required for each of the above algorithms to converge. Convergence is declared if $\|\nabla\phi_k\|_\infty < 10^{-5}(1 + |\phi_k|)$, where $\|(x_1, x_2, ..., x_n)^T\|_\infty := \max(|x_1|, |x_2|, ..., |x_n|)$. The columns described as 'mod' indicate the number of times the methods β_k^{FR-PR} and β_k^{PR+} differ from

		FR	FR–PR		PR	PR^+	
P	N	it/f-g	it/f-g	mod	it/f-g	it/f-g	mod
2	200	703/1424	701/1420	596	701/1420	701/1420	0
3	200	2808/5617	2808/5617	2808	2631/5263	2631/5263	0
6	500	*	1107/2231	433	1068/2151	1067/2149	1
8	1000	12/39	9/34	7	6/28	10/42	2
10	1000	138/281	145/299	142	165/338	165/338	0
28	1000	533/1102	1369/2741	1366	212/473	97/229	3
31	200	2/4	2/4	1	1/5	1/5	0
38	1000	264/531	263/529	217	262/527	262/527	0
39	961	*	143/290	5	142/287	142/287	0
42	10000	6/26	6/26	5	7/28	6/26	1
43	1000	10/27	15/38	15	10/33	9/29	2
47	1000	422/849	114/233	92	113/231	113/231	0
48	10000	61/143	130/283	123	24/73	19/62	4
49	1000	568/1175	1369/2741	1366	212/473	97/229	3
50	1000	274/551	273/549	245	274/551	274/551	0
51	1000	231/467	40/91	5	40/92	40/92	0
52	10000	4/15	4/15	4	3/13	3/13	1
53	403	**	233/494	130	237/508	237/508	0
54	455	**	44/91	7	44/87	44/87	0
55	1559	23/47	23/49	15	23/47	23/47	0

Table 1: Convergence speeds of non-linear conjugate gradient methods on various problems P of size N, as reported by [Gilbert and Nocedal, 1992].

β_k^{PR}. Lastly, $*$ indicates failure by exceeding 9999 iterations and $**$ indicates failure to find a suitable α_k through the line search procedure. For a description of the individual problems and the implementation of the line search we refer to the original publication.

From this table it is clear that FR often converges more slowly and that it is less stable than its alternatives. We also note that the non-linear CG often converge faster to the solution in less than n steps as predicted. To visualize the first property in two dimensions, we construct two functions that are deemed to be sufficiently non linear, namely

$$\begin{aligned} r_1(x,y) =& y^4 - \frac{1}{2}y^2 + 2yx - y\cos(x) - \sin(y) \\ &+ x^4 - \frac{1}{2}x^2 - x\cos(y) - \sin(x) + \frac{\cos(xy)\sin(xy)}{4y}; \end{aligned} \tag{30a}$$

$$\begin{aligned} r_2(x,y) =& y^4 - \frac{1}{2}y^2 + y|x| - y|\cos(x)| - \sin(y) + xy \\ &+ x^4 - \frac{1}{2}x^2 - x\cos(y) - \sin(x) + \frac{\cos(xy)\sin(xy)}{4y}, \end{aligned} \tag{30b}$$

where we note that in contrast to the second function, the first function is smooth near $x = 0$.

In figure 2, the optimizers minimize the first function using a line search that finds a netxt iterate satisfying the strong Wolfe conditions using $c_1 = 10^{-4}$ and $c_2 = .4$. This was implemented by modifying exiting code from SciPy [Jones et al., 2001]. Convergence is declared if $\|\nabla\phi_k\|_2 < 10^{-5}$. A behaviour

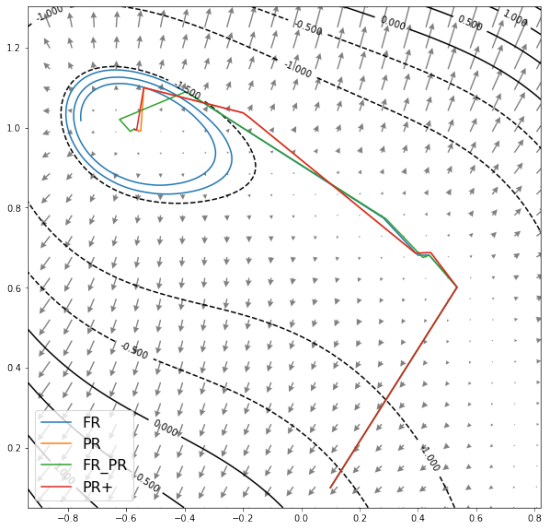


Figure 2: various non-linear conjugate gradient methods applied to the objective function in equation (30a), starting at $(.1, .1)$. Convergence speeds are reported in table 2.

that stands out is FR spiraling around the minimum, taking very small steps, whereas the other methods converge straight to the minimum once they are near. This can be understood by rewriting the RHS of equation (21) and noting that

$$\begin{aligned} 0 < \frac{1-2c_2}{1-c_2}\frac{\|\nabla\phi_k\|}{\|p_k\|}, & \quad \text{since } c_2 \in (0, \frac{1}{2}) \text{ and } x_k \neq x^*; \\ \leq \frac{-\nabla\phi_k^T p_k}{\|\nabla\phi_k\|\|p_k\|}, & \quad \text{by eq. (21);} \\ = \cos(\theta_k), & \quad \text{by eq. (25).} \end{aligned}$$

Which tells us that if the gradient and the direction p_k computed by FR are nearly orthogonal, and therefore $\cos(\theta_k) \approx 0$, then $\|\nabla\phi_k\| \ll \|p_k\|$. Since x_k is nearing the solution and the steepest descent is pointing directly to the minimum, it is reasonable to assume that a direction that is almost orthogonal to the gradient will lead to a small step. This will result in $\nabla\phi_{k+1} \approx \nabla\phi_k$, which in turn results in $\beta_{k+1} = \frac{\|\nabla\phi_{k+1}\|}{\|\nabla\phi_k\|} \approx 1$. Because of this, p_k will likely dominate $p_{k+1} = -\nabla\phi_{k+1} + \beta_{k+1}p_k$, leading to a long series of small steps in ineffective directions. In contrast, $\beta_{k+1}^{PR} \approx 0$ when $\nabla\phi_{k+1} \approx \nabla\phi_k$, resetting the method to the steepest descent direction. To find out how reliant the method is on the smoothness conditions, the experiments are repeated using function (30b). As can be seen in figure 3, the method quickly stagnates as it fails to find a next iterate satisfying the strong Wolfe conditions. Since the strong Wolfe conditions assume that gradient vanishes at the minimum, the line search is adjusted to

	FR	PR	PR+		FR-PR	
figure	it/ϕ, $\nabla\phi$	it/ϕ, $\nabla\phi$	it/ϕ, $\nabla\phi$	mod	it/ϕ, $\nabla\phi$	mod
2	$\overset{*}{400/520}$	11/30	10/28	2	15/37	12
3	$\overset{**}{4/115}$	$\overset{**}{3/96}$	$\overset{**}{3/96}$	0	$\overset{**}{4/118}$	3
4	$\overset{**}{35/277}$	$\overset{**}{43/254}$	$\overset{**}{43/254}$	0	$\overset{**}{38/288}$	3

Table 2: Convergence speeds of the optimization procedures displayed in the figures 2, 3 and 4. The table is structures in the same way as table 1, with the exception that the number of iterations and function evaluations are still reported when methods did not converge. Transitions in the line search parameter c_2 happen after 100 unfruitful function and gradient evaluations

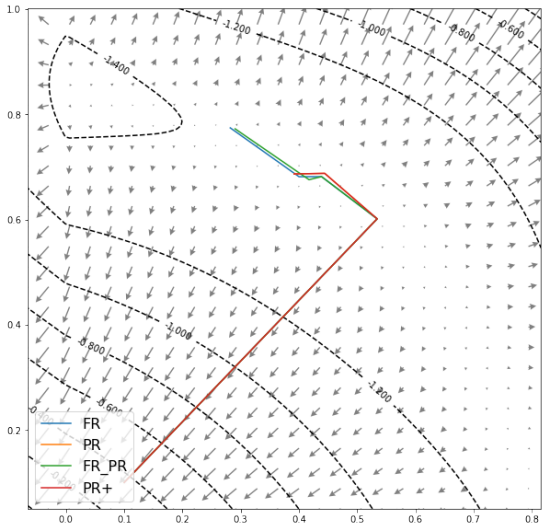
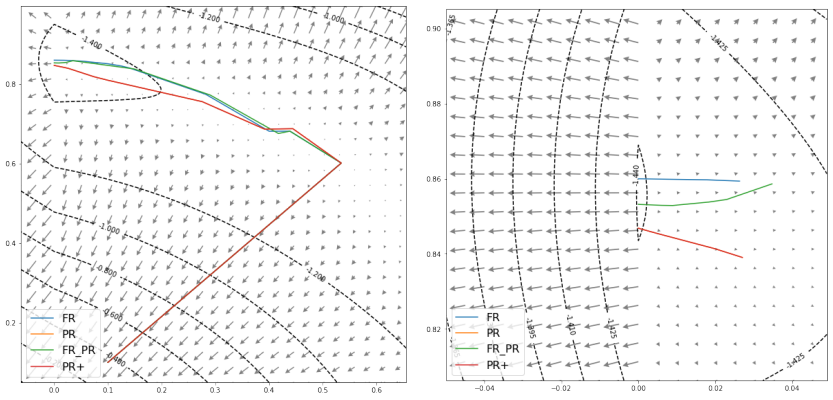


Figure 3: various non-linear conjugate gradient methods applied to the objective function in equation (30b), starting at $(.1, .1)$. Convergence speeds are reported in table 2.

try $c_2 = 1.2$ and $c_2 = 3.6$ after failure[3]. The results are displayed in figure 4. We note that convergence is never declared since

$$\lim_{x\uparrow 0} \frac{\partial r_2(x,y)}{\partial x} = -\cos(y) - \frac{3}{4}; \qquad \lim_{x\downarrow 0} \frac{\partial r_2(x,y)}{\partial x} = 2y - \cos(y) - \frac{3}{4}.$$

[3]It wa later verified that these c_2 values essentially remove the second strong wolfe condition since $c_2 = 10^{20}$ gives similar results (not shown). Because of this CG is actually 100-200 evaluations faster than in table 2. Because the gradient is very steep for $x < 0$ near 0, the interval on the other side of the ridge is too narrow for a line search to find. Because of this using the wolfe conditions instead of the strong wolfe conditions leads to the same results.



(a) (b)

Figure 4: various non-linear conjugate gradient methods applied to the objective function in equation (30b), using the adjusted line search algorithm, starting at $(.1, .1)$. (a) displays the entire trajectory, (b) is zoomed in on the minimum. Convergence speeds are reported in table 2.

To still give a measure of how well the solution is approximated, we note that the solution is located at $x = 0$, which yields the equation

$$\left.\frac{\partial r_2(x,y)}{\partial y}\right|_{x=0} = 4y^3 - y - cos(y) - 1 = 0.$$

This results in the approximate solution $(0, .856301)$. This approximation is then used to determine the distances to the last iterates. These distances are displayed in table 3 and it is worth noting that although the algorithm broke down, great approximations are still obtained.

FR	PR	PR+	FR-PR
$3.67 \cdot 10^{-3}$	$9.49 \cdot 10^{-3}$	$9.49 \cdot 10^{-3}$	$3.12 \cdot 10^{-3}$

Table 3: The distances of the last iterates displayed in figure 4 to a minimum of the function (30b)

3 Nesterov Accelerated Gradient

3.1 Momentum

In the previous chapter we saw that non-linear CG can arrive at the solution of a system quite quickly, especially considering the algorithm only has access

to local information in a highly non linear landscape. However, the algorithm breaks down when the objective function is no longer smooth. Although the smoothness was only violated in one dimension, CG still fails to make progress in the other dimensions. This could be a big problem if the parameter space is large and one dimension can stagnate all the others.

In this chapter, another approach will be introduced, namely gradient descent with momentum. In such an algorithm, a running average of the gradient is used to take the descent steps without a line search, making it resistant to fluctuations in the gradient. Such an algorithms is useful when dealing with conflicting local information due to a noise or due to great local changes in the steepness of the landscape. The exact algorithm used for constructing a running average turns out to matter little in practise since the different methods exhibit similar behaviour. Nonetheless, it matters for the analysis of its convergence rate. Because of this, we analyze the variant called Nesterov Accelerated Gradient (NAG) [Nesterov, 1983], algorithm 4, in the next section.

Algorithm 4 Nesterov Accelerated Gradient

1: Given $x_0, \eta_0, lr_0, k_{max}$
2: Set $k \leftarrow 0$
3: **while** $k \leq k_{max}$ **do**

Evaluate $\nabla\phi_k$;

$y_{k+1} \leftarrow x_k - lr_k \nabla\phi_k$
$x_{k+1} \leftarrow y_{k+1} + \eta_k(y_{k+1} - y_k)$
Compute lr_{k+1}, η_{k+1}
$k \leftarrow k + 1$

Before we specify η_k and lr_k, we first obtain an intuition as to why this method can be thought of as an algorithm that uses momentum. We first consider its behaviour in the situation where the iterates enter a regime in which the gradient suddenly drops to values close to 0. That way, $x_{k+1} - x_k$ is dominated by the built up momentum in the term $\eta_k(y_{k+1} - y_k)$, where $0 < \eta_k < 1$. In a regime where the gradient is very low, this is then approximately equal to $\eta_k(x_k - x_{k-1})$. In other words, the step taken by the current iterate is given by the decayed step of the previous iterate.

Another illustration can be given by considering the case where the gradients, the learn rate (or step size) lr_k and the momentum parameter η are constant.

This gives the step sizes

$$\begin{aligned} x_{k+1} - x_k &= -c + \eta(x_k - x_{k-1}) \\ &= -c\sum_{t=0}^{k-1} \eta^t + \eta^k(x_1 - x_0) \\ &= -c\sum_{t=0}^{k-1} \eta^t + \eta^k\big(-c + \eta(x_0 - c - x_0)\big) \\ &= -c\sum_{t=0}^{k+1} \eta^t, \end{aligned}$$

for some c. Therefore the step sizes converge (or accelerate) to $\frac{-1}{1-\eta}c$, much larger in amplitude than c. Additionally, if the gradient and therefore c were to suddenly flip sign, the steps would slowly decrease, accelerating in the reverse direction.

NAG uses an ever increasing momentum term that is given by $\eta_k = \frac{\lambda_k - 1}{\lambda_{k+1}}$, where $\lambda_{-1} = 0$ and $\lambda_{k+1} = \frac{1+\sqrt{1+4\lambda_k^2}}{2}$. For $k > 0$ this means that $0 < \eta_k < 1$ and that it grows to 1 as $k \to \infty$. One can show this by noticing that λ_k is a solution to the quadratic equation $0 = \lambda_k^2 - \lambda_k - \lambda_{k-1}^2$. Since $\lambda_k > 0$ for $k > 0$, one finds that

$$\lambda_k < \lambda_{k+1} - \frac{1}{2} < \lambda_{k+2} - 1 = \frac{\lambda_{k+1}^2}{\lambda_{k+2}},$$

which in turns yields that

$$\eta_k = \frac{\lambda_k - 1}{\lambda_{k+1}} < \frac{\lambda_{k+1} - 1}{\lambda_{k+2}} = \eta_{k+1},$$

since $\frac{-1}{\lambda_{k+1}} < \frac{-1}{\lambda_{k+2}}$. This shows that the sequence is monotonically increasing. We also note that by the same inequalities we have that $\lambda_k + n < \lambda_{k+2n}$, which shows $\lambda_k \to \infty$ as $k \to \infty$, showing that η_k grows to 1 as $k \to \infty$ (write out η_k's definition). For defining its step size, NAG assumes that the objective function is Lipschitz continuously differentible with constant β. The learn rate is then given by $lr = \frac{1}{\beta}$.

3.2 Convergence rate

3.2.1 Convexity

Now that we have obtained an intuition for the algorithm, we derive a theoretical convergence rate, which also shows global convergence. For this proof there

are two assumptions. The first is that $\phi : \mathbb{R}^n \to \mathbb{R}$ is Lipschitz continuously differentiable with constant β, in other words $\|\nabla\phi(x) - \phi(y)\| \leq \beta\|x - y\|$. This will be referred to this by saying that ϕ is β-smooth. The other assumption is that ϕ is convex, where convexity is defined below.

Definition 3.1. A set $\chi \subset \mathbb{R}^n$ is said to be convex if it contains all its segments, that is

$$tx + (1-t)y \in \chi, \qquad \text{for all } (x, y, t) \in \chi \times \chi \times [0, 1].$$

A function $\phi : \chi \to \mathbb{R}$ is said to be convex if

$$\phi\big(tx + (1-t)y\big) \leq t\phi(x) + (1-t)\phi(y), \qquad \text{for all } (x, y, t) \in \chi \times \chi \times [0, 1].$$

A function $\phi : \chi \to \mathbb{R}$ is said to be α-strongly convex with constant $\alpha > 0$ if

$$\phi(x) - \phi(y) \leq \nabla\phi(x)^T(x - y) - \frac{\alpha}{2}\|x - y\|^2, \qquad \text{for all } x, y \in \chi.$$

To develop an intuition, we note that strong convexity, as the names indicate, implies convexity. To show this (momentarily ignoring that $\phi : \mathbb{R}^n \to \mathbb{R}$ and that $\mathbb{R}^n$ is convex), we note that if ϕ is strongly convex, then

$$\phi(x) - \phi(y) \leq \nabla\phi(x)^T(x - y), \qquad \text{for all } x, y \in \chi, \tag{32}$$

as $\alpha > 0$. Because χ is convex, we have that $z = tx + (1-t)y \in \chi$ for all $t \in [0, 1]$. Therefore, we have by the above inequality that

$$\begin{aligned} t\Big(\phi(z) - \phi(x)\Big) \qquad\qquad &\leq \quad t\Big(\nabla\phi(z)^T(z - x)\Big) \\ +(1-t)\Big(\phi(z) - \phi(y)\Big) \qquad\qquad &\quad + (1-t)\Big(\nabla\phi(z)^T(z - y)\Big). \end{aligned}$$

This reduces to

$$-t\phi(x) + \phi(z) + (t-1)\phi(y) \leq \nabla\phi(z)^T(ty - tx + z - y),$$

which in turn reduces to

$$\phi\big(tx + (1-t)y\big) \leq t\phi(x) + (1-t)\phi(y) \tag{33}$$

by substituting back z. From this we conclude that strong convexity implies convexity. It also holds that (33) implies (32). One can see this by noticing that by reshuffling the terms in equation (33), we get

$$\frac{\phi\big(y + t(x - y)\big) - \phi(y)}{t} \leq \phi(x) - \phi(y),$$

which by the definition of the derivative results in equation (32), as $t \to 0$ (note that x and y are swapped). Combined, this is known as the first order condition. Intuitively this means that a function is convex if and only if first order taylor approximations always give a lower bound of the function value.

3.2.2 β-smoothness and convexity

In preparation of the analysis of the convergence rate, we now show that under the assumed β-smoothness we have that

$$|\phi(x) - \phi(y) - \nabla\phi(y)^T(x-y)| \leq \frac{\beta}{2}\|x-y\|^2. \tag{34}$$

We do this by rewriting the LHS in terms of integrals and noticing that

$$\begin{aligned}
&\Big|\Big(\phi(x)-\phi(y)\Big) - \Big(\nabla\phi(y)^T(x-y)\Big)\Big| \\
&= \Big|\Big(\int_0^1 \nabla\phi\big(y+t(x-y)\big)^T(x-y)\,\mathrm{d}t\Big) - \Big(\int_0^1 \nabla\phi(y)^T(x-y)\,\mathrm{d}t\Big)\Big| \\
&= \Big|\int_0^1 \Big(\nabla\phi\big(y+t(x-y)\big) - \nabla\phi(y)\Big)^T(x-y)\,\mathrm{d}t\Big| \\
&\leq \int_0^1 \Big(\|\nabla\phi\big(y+t(x-y)\big) - \nabla\phi(y)\|\Big)\|x-y\|\,\mathrm{d}t, \quad \text{by Cauchy–Schwarz} \\
&\leq \int_0^1 \Big(\beta\|t\big(x-y\big)\|\Big)\|x-y\|\,\mathrm{d}t, \qquad\qquad \text{by } \beta\text{-Lipschitz} \\
&= \frac{\beta}{2}\|(x-y)\|^2.
\end{aligned}$$

This result can then be combined with convexity to show that, by the first order condition, we have that

$$0 \leq \phi(x) - \phi(y) - \nabla\phi(y)^T(x-y)| \leq \frac{\beta}{2}\|x-y\|^2. \tag{35}$$

The previous inequlity can in turn be used to show that for $\tilde{x} = x - \frac{1}{\beta}\nabla\phi(x)$;

$$\phi(\tilde{x}) - \phi(y) \leq \nabla\phi(x)^T(x-y) - \frac{1}{2\beta}\|\nabla\phi(x)\|^2, \qquad \text{for all } x, y \in \mathbb{R}^n. \tag{36}$$

This is done by also using the first order condition and substituting back $\tilde{x}$, which results in

$$\begin{aligned}
\phi(\tilde{x}) - \phi(y) &= \phi(x) - \phi(y) \quad + \ \phi(\tilde{x}) - \phi(x) \\
&\leq \nabla\phi(x)^T(x-y) + \nabla\phi(x)^T(\tilde{x}-x) + \frac{\beta}{2}\|\tilde{x}-x\|^2 \\
&= \nabla\phi(x)^T(x-y) - \frac{1}{2\beta}\|\nabla\phi(x)^T\|^2.
\end{aligned}$$

3.2.3 Convergence analysis

For the analysis of the convergence rate, recall that $lr = \frac{1}{\beta}$ and $\eta_k = \frac{\lambda_k - 1}{\lambda_{k+1}}$, where $\lambda_{-1} = 0$ and $\lambda_{k+1} = \frac{1+\sqrt{1+4\lambda_k^2}}{2}$. We will show that using these parame-

ters, for any starting point $x_0 = y_0$, we have that

$$\phi(y_k) - \phi(x^*) \leq \frac{2\beta\|x_0 - x^*\|^2}{(k+1)^2}, \qquad \text{for all } k > 0. \tag{37}$$

We begin by noting that by equation (36) and the definition of y_{k+1} we have that

$$\begin{aligned}\phi(y_{k+1}) - \phi(y_k) &\leq \nabla\phi(x_k)^T(x_k - y_k) - \frac{1}{2\beta}\|\nabla\phi(x_k)\|^2 \\ &= \beta(x_k - y_{k+1})^T(x_k - y_k) - \frac{\beta}{2}\|x_k - y_{k+1}\|^2.\end{aligned} \tag{38}$$

In the same way we also find that

$$\phi(y_{k+1}) - \phi(x^*) \leq \beta(x_k - y_{k+1})^T(x_k - x^*) - \frac{\beta}{2}\|x_k - y_{k+1}\|^2. \tag{39}$$

Multiplying equation (38) by $(\lambda_k - 1)$ and adding it to equation (39) we get by reshuffling the terms of both sides that

$$\begin{aligned}&\lambda_k\big(\phi(y_{k+1}) - \phi(x^*)\big) + (1-\lambda_k)\big(\phi(y_k) - \phi(x^*)\big) \\ &\leq \beta(x_k - y_{k+1})^T\big(\lambda_k x_k + (1-\lambda_k)y_k - x^*\big) - \lambda_k\frac{\beta}{2}\|x_k - y_{k+1}\|^2.\end{aligned}$$

Next, using the notation $\delta_k = \phi(y_k) - \phi(x^*)$, we take advantage of the identity $2a^Tb - \|a^2\| = \|b\|^2 - \|b-a\|^2$ by multiplying the above inequality by λ_k. This yields

$$\begin{aligned}&\lambda_k^2\delta_{k+1} + \lambda_k(1-\lambda_k)\delta_k \\ &\leq \frac{\beta}{2}\Big(2\big(\lambda_k(x_k - y_{k+1})\big)^T\big(\lambda_k x_k + (1-\lambda_k)y_k - x^*\big) - \|\lambda_k x_k - y_{k+1}\|^2\Big) \\ &= \frac{\beta}{2}\Big(\|\lambda_k x_k + (1-\lambda_k)y_k - x^*\| - \|\lambda_k y_{k+1} + (1-\lambda_k)y_k - x^*\|\Big).\end{aligned} \tag{40}$$

This can be further rewritten by noticing that by the definition of the algorithm we have that

$$\begin{aligned}x_{k+1} &= y_{k+1} + \eta_k(y_{k+1} - y_k); \\ \lambda_{k+1}x_{k+1} &= \lambda_{k+1}y_{k+1} + (\lambda_k - 1)(y_{k+1} - y_k); \\ \lambda_{k+1}x_{k+1} + (1-\lambda_{k+1})y_{k+1} &= \lambda_k y_{k+1} + (1-\lambda_k)y_k.\end{aligned}$$

With this, the contents of the norms in equation (40) can be rewritten using the notation $v_k = \lambda_k x_k + (1-\lambda_k)y_k - x^*$. Together with the fact that $\lambda_{k-1}^2 = \lambda_k^2 - \lambda_k$, this results in

$$\lambda_k^2\delta_{k+1} - \lambda_{k-1}^2\delta_k \leq \frac{\beta}{2}\Big(\|v_k\| - \|v_{k+1}\|\Big), \qquad \text{for all } k \geq 0.$$

Since $\lambda_{-1} = 0$ it can be concluded that

$$\begin{aligned}
\lambda_{k-1}^2 \delta_k &= \lambda_{k-1}^2 \delta_k - \lambda_{-1}^2 \delta_0 \\
&= \sum_{t=0}^{k-1} \lambda_t^2 \delta_{t+1} - \lambda_{t-1}^2 \delta_t \\
&\leq \sum_{t=0}^{k-1} \frac{\beta}{2} \Big(\|v_t\| - \|v_{t+1}\| \Big) \\
&= \frac{\beta}{2} \Big(\|v_0\| - \|v_k\| \Big) \\
&\leq \frac{\beta}{2} \|v_0\|.
\end{aligned}$$

This gives

$$\phi(y_k) - \phi(x^*) \leq \frac{\beta}{2\lambda_{k-1}^2} \|v_0\|, \qquad \text{for all } k > 0.$$

By induction it follows directly that $\frac{k+1}{2} \leq \lambda_{k-1}$ since $\lambda_0 = 1$ and $\frac{t+1}{2} \leq \lambda_{k-1} \leq \lambda_k$ if the inequality holds for k. Therefore we conclude that equation (37) holds.

3.3 Momentum in practise

In contrast to the analysis in the previous section, when using Momentum Based Methods (MBM) in practise, the momentum term η is often chosen constant. To understand why a lower momentum might be favourable, one should realize that the momentum term was chosen such that it had favourable properties on convex problems. However, when using high dimensional very non linear and rugged functions, this might not be desirable as the landscape can change drastically at any point in the optimization procedure. For example, when training a model in computer vision, first rough features such as colour shades and contours are detected before the model can distinguish details, such as fur types to differentiate between different breeds of dogs. However, when momentum becomes too high, the optimizer 'remembers' gradients from too far back. It will then keep moving in the direction it remembered to be favourable when it was learning about rudimentary shapes and will therefore fail at learning more detail quickly.

In a widely used optimizer called Adam, [Kingma and Ba, 2014]. The momentum parameter is chosen to be .9. Adam mainly differs from NAG in using the magnitutde of the gradients to make equal progress in all directions, regardless of steepness. Other than providing even more stability, in machinelearning, this intuitively allows learning to take place when features are sparse.

Another detail in which practical applications differ from the approach above is that the step size is often exponentially decreased in which case it can be given

by $lr_k = \rho^k lr_0$ for some $0 \ll \rho \leq 1$. A decaying step size is often used to capture fine grain details or in other words, find the local minima more precisely.

We now visualize the behaviour of Adam and NAG using the functions (30a) and (30b) from the previous chapter. The trajectories are displayed in the figures 5 and 6. Here, Adam uses its default parameters. Similarly, for NAG $\eta = .9$ is chosen. The learn rates used are $lr_k^{Adam} = \frac{1}{10} \cdot .99^k$ and $lr_k^{NAG} = \frac{2}{100} \cdot .99^k$. In the appendix, 5.2, a demonstration is given of NAG using its increasing momentum with the same learn rate.

figure	Adam it	NAG it
5	184	149
6	152	215

Table 4: Convergence rates of the optimization procedures displayed in the figures 5 and 6. In contrast to the previous section, the table displays only the iterations required for convergence as no line search is used.

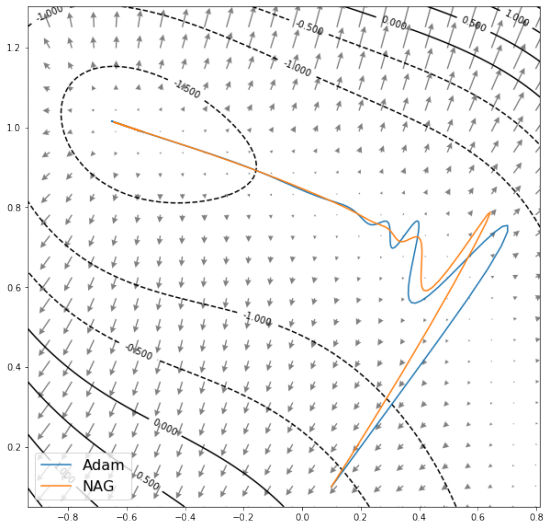
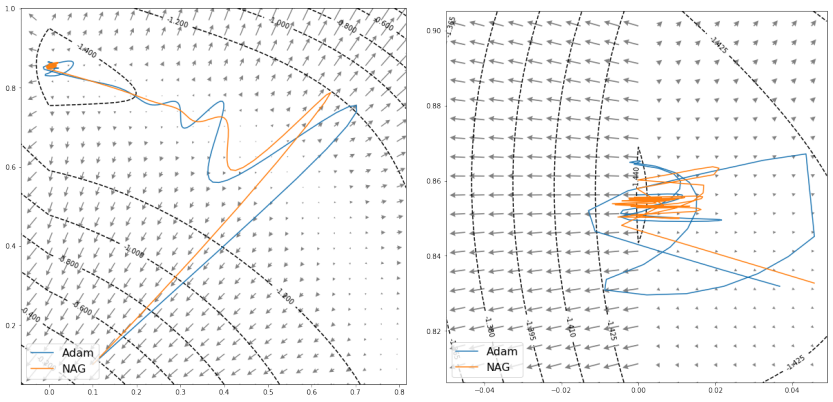


Figure 5: The momentum based optimizers Adam and NAG applied to the objective function in equation (30a), starting at $(.1, .1)$. The convergence rates are reported in table 4.

What makes MBM particularly suitable for the optimization of non smooth functions is that they are able to traverse the ridge at $x = 0$ of function 30b, as can be observed in figure 6b. Even though the gradient does not vanishing at the minimum, an indication of convergence speed is still given by declaring convergence when $\|x_k - (0, .856301)^T\|_2 < 10^{-3}$. These rates, along with the



(a) (b)

Figure 6: The momentum based optimizers Adam and NAG applied to the objective function in equation (30b), starting at $(.1, .1)$. (a) displays the entire trajectory, (b) is zoomed in on the minimum. The convergence rates are reported in table 4.

convergence rates of trajectories in figure 5, are shown in table 4. When comparing these results to those of table 2 it can be noted that although CG can get stuck, they converged better on our problems.

Another argument for the use of CG can be found in the noise visible in figure 6b. This noise is the optimizer requiring a lower learn rate to converge properly. This problem is automatically addressed by Adam as it scales its learn rate for each dimension individually. However, NAG does not solve this problem automatically. Therefore it 'waits' for the step size to decrease or for it to be declared 'converged', simply by chance.

4 Discussion and conclusions

We have seen various optimizers with different properties. Because our initial interest was to apply the optimizers in machine learning, we note that in supervised learning (learning using given datasets), models tend to perform very well if they contain tens of millions of parameters and use the non-linearity $\max(x, 0)$, also referred to as the Rectified Linear Unit (ReLU). Given the stability of MBM, Adam is often used as the default optimizer for such tasks. Another active field of research is Reinforcement Learning (RL, an agent learns from a self-created dataset), in which much smaller models tend to work better. In these models, non linear functions such as the tanh are more common. Although line searches are difficult to use in RL, CG could potentially excel at

such tasks. In fact, the recently published method called Trust Region Policy Optimization (TRPO), introduced in [Schulman et al., 2015] uses CG in their algorithm.

References

C. Angermueller, H. J. Lee, W. Reik, and O. Stegle. Deepcpg: Accurate prediction of single-cell dna methylation states using deep learning. *Genome biology*, 18(1), 2017.

Yonghui Wu, Mike Schuster, Zhifeng Chen, Quoc V. Le, Mohammad Norouzi, Wolfgang Macherey, Maxim Krikun, Yuan Cao, Qin Gao, Klaus Macherey, Jeff Klingner, Apurva Shah, Melvin Johnson, Xiaobing Liu, Lukasz Kaiser, Stephan Gouws, Yoshikiyo Kato, Taku Kudo, Hideto Kazawa, Keith Stevens, George Kurian, Nishant Patil, Wei Wang, Cliff Young, Jason Smith, Jason Riesa, Alex Rudnick, Oriol Vinyals, Greg Corrado, Macduff Hughes, and Jeffrey Dean. Google's neural machine translation system: Bridging the gap between human and machine translation. *CoRR*, abs/1609.08144, 2016.

K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, volume 2016-January, pages 770–778, 2016.

A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in Neural Information Processing Systems*, volume 2, pages 1097–1105, 2012.

S. Wright Jorge Nocedal. *Numerical Optimization*. Springer New York, 2006. doi: 10.1007/978-0-387-40065-5. URL `https://doi.org/10.1007/978-0-387-40065-5`.

Sébastien Bubeck. Convex optimization: Algorithms and complexity. *Foundations and Trends® in Machine Learning*, 8(3-4):231–357, 2015. doi: 10.1561/2200000050. URL `https://doi.org/10.1561/2200000050`.

M. R. Hestenes and E. Stieffel. Methods of conjugate gradients for solving linear systems. *Journal of Research of the National Bureau of Standards*, 49: 409–436, 1952. doi: 10.1101/081380.

J. Gilbert and J. Nocedal. Global convergence properties of conjugate gradient methods for optimization. *SIAM Journal on Optimization*, 2:21–42, 1992. doi: 10.1137/0802003.

Eric Jones, Travis Oliphant, Pearu Peterson, et al. SciPy: Open source scientific tools for Python, 2001. URL `http://www.scipy.org/`.

Yurii Nesterov. A method of solving a convex programming problem with convergence rate $O(1/(k^2))$. *Soviet Mathematics Doklady*, 27:372–376, 1983.

Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *CoRR*, abs/1412.6980, 2014. URL `http://arxiv.org/abs/1412.6980`.

John Schulman, Sergey Levine, Philipp Moritz, Michael I. Jordan, and Pieter Abbeel. Trust region policy optimization. *CoRR*, abs/1502.05477, 2015. URL `http://arxiv.org/abs/1502.05477`.

5 Appendix

5.1 Lemmas

Lemma 5.1. if A is a symmetric matrix, it can be written in the form $A = \sum_{i=1}^{n} \lambda_i v_i v_i^T$ where λ_i and v_i are the eigenvalues and corresponding eigenvectors of A.

Proof. Because A is symmetric it has eigenvectors that are orthonormal. Define the matrix $R = [v_0, v_1, ..., v_n]$ and notice that $R^T R = I$ as $v_i^T v_j = 0$ for $i \neq j$ and 1 otherwise due to orthonormality. Because the eigenvectors are orthonormal, they are linearly independent, thus R is invertible. Because of this, $R^T = R^{-1}$ and therefore $RR^T = I$. We can now write

$$A = AI = ARR^T = A\sum_{i=1}^{n} v_i v_i^T = \sum_{i=1}^{n} A v_i v_i^T = \sum_{i=1}^{n} \lambda_i v_i v_i^T$$

□

5.2 NAG reimplemented

In this section NAG is reimplemented using its initial parameter $\eta_k = \frac{\lambda_k - 1}{\lambda_{k+1}}$ and the learn rate $lr_k = \frac{2}{100} \cdot .99^k$ as used in section (3.3) for a fair comparison. The results are shown in the table and graphs below.

figure	NAG it
7	369
8	165

Table 5: Convergence rate of the reimplemented NAG displayed in the figures 7 and 8.

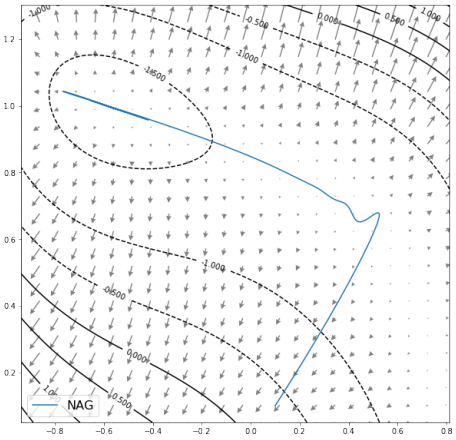
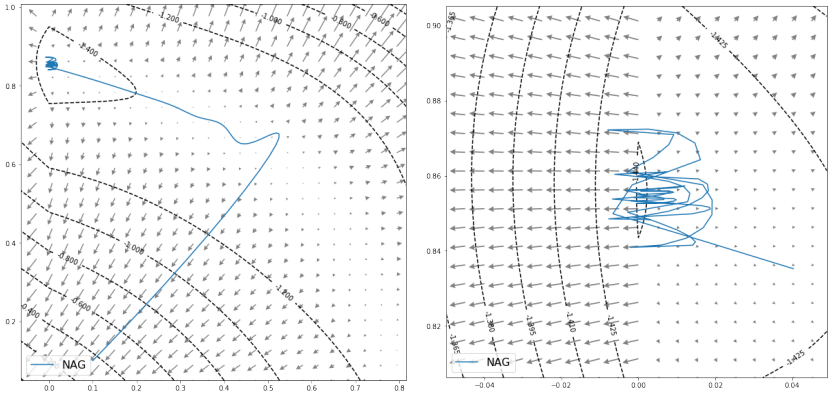


Figure 7: The reimplemented NAG applied to the objective function in equation (30a), starting at $(.1, .1)$. The convergence rate is reported in table 5. The iterate are smoothly directed to the minimum when momentum is still low, but overshoot the minimum as momentum grows.



(a) (b)

Figure 8: The reimplemented NAG applied to the objective function in equation (30b), starting at $(.1, .1)$. (a) displays the entire trajectory, (b) is zoomed in on the minimum. The convergence rates are reported in table 5.