Conjugate Gradients: The short and painful explanation with oblique projections

Robert M. Gower

January 12, 2015

Abstract

Conjugate Gradient (CG) algorithm and its variants are among the most popular methods for iteratively solving linear systems. Their adaptation for nonlinear optimization, the nonlinear CG methods, also enjoys wide spread notoriety. Many available essays describe the algorithm in a very algorithmic way; as a series of strange scalar and vector multiplications that achieve the desired result. It is hard to see the mathematics behind these cake recipes. Here we emphasize a mathematical description and leave implementation issues separate for, well, when we need to implement. We particularly enjoy writing the method using oblique projections. This is useful, as mathematics has a lot to say about projections and their properties. Preconditioned and projected CG is also covered and an additional new method for achieving conjugate directions by restarting CG method with a preconditioner that "remembers" previous conjugate directions. For most readers, we refer to the excellent, intuitive, though rather long explanation offered by J. Shewchuk [6]. This essay, rather, offers a short and painful description of the CG method.

Contents

Disclaimer: This text is aimed at readers who need to quickly understand details of the CG method. I personally wrote this essay as I needed to consider a number of adaptations of the CG method to mesh with different optimization methods. These alterations required understanding restricting the search space, projecting the iterates, preconditioning, restarting, and a mix of the above. Comments and suggestions are always welcome: gowerrobert@gmail.com.

1 CG through projections

The conjugate gradient method, by Hestenes and Stiefel [4], is an iterative method for finding the solution to

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

where $x, b \in \mathbb{R}^n$ and $A \in \mathbb{R}^{n \times n}$ is a positive definite matrix which guarantees that the critical point defined by $\nabla\phi(x) = Ax - b = 0$ is the unique solution. Given a $x_0 \in \mathbb{R}^n$, the method iteratively finds x_k, the minimum of $\phi(x)$ constrained to $x_0 \oplus \mathcal{K}_k$, where

$$\mathcal{K}_k := \text{span}(\{\nabla\phi(x_0), A\nabla\phi(x_0), \ldots, A^{k-1}\nabla\phi(x_0)\}),$$

is the kth *Krylov subspace*. The Krylov subspaces are nested, in that $\mathcal{K}_k \subset \mathcal{K}_{k+1}$, thus each x_{k+1} tends to be an improvement over the previous x_k. What is more, the solution to (1) is often encountered before reaching $\mathcal{K}_n$. This depends on the number of distinct eigenvalues of A.

Lemma 1. *Let λ_i for $i = 1, \ldots, \rho$ be these distinct eigenvalues of A, then $\mathcal{K}_\rho$ is a ρ-dimensional subspace that contains the solution to* (1)*.*

Proof. As A is symmetric, $\mathbb{R}^n$ is a direct sum of the Eigenspaces of A. Thus we can decompose $\nabla\phi(x_0) = \sum_{j=1}^{\rho} w_j$ where w_j is an eigenvector associated to λ_j, for $j = 1, \ldots, \rho$. Let $x = \sum_{i=1}^{\rho} \alpha_i A^{i-1}\nabla\phi(x_0) \in \mathcal{K}_\rho$ it follows that

$$\begin{aligned} Ax &= \sum_{i=1}^{\rho} \alpha_i A^i \nabla\phi(x_0) \\ &= \sum_{i=1}^{\rho} \alpha_i A^i \sum_{j=1}^{\rho} w_j \\ &= \sum_{j=1}^{\rho}\sum_{i=1}^{\rho} (\alpha_i \lambda_j^i) w_j. \end{aligned}$$

The uniqueness of the λ_j's guarantees that the system $\sum_{i=1}^{\rho}(\alpha_i\lambda_j^i) = 1$, for $j = 1, \ldots, \rho$, has a unique solution α. Thus the solution x to (1) is in $\mathcal{K}_\rho$. Similar arguments show that $\{\nabla\phi(x_0), \ldots, A^{\rho-1}\nabla\phi(x_0)\}$ is a basis and $\mathcal{K}_\rho$ is ρ-dimensional subspace. □

Restricting the search for a solution to $\mathcal{K}_\rho$ is a good idea as it can be a smaller space then the entire $\mathbb{R}^n$. But how to orderly search through the Krylov subspaces and for $k < \rho$ use the solution restricted to $\mathcal{K}_k$ in calculating the solution restricted to the next space $\mathcal{K}_{k+1}$? The answer is to search along a basis of $\mathcal{K}_k$ of conjugate vectors in relation to A. Two vectors $d_1, d_2 \in \mathbb{R}^n$ are A-conjugate if they are orthogonal in relation to the inner product defined by A,

$$\langle d_1, d_2 \rangle_A := \langle d_1, Ad_2 \rangle = 0.$$

If the columns of $D_k := [d_0 \cdots d_{k-1}] \in \mathbb{R}^{n\times k}$ form an A-conjugate basis of $\mathcal{K}_k$, then the minimization of $\phi(x)$ over $\mathcal{K}_k$ can be written as $x_k = D_k y_k + x_0$, where

$$\min_{\substack{x_k = D_k y_k + x_0 \\ y_k \in \mathbb{R}^k}} \frac{1}{2}x^T Ax - x^T b = \min_{y\in\mathbb{R}^k} \frac{1}{2} y^T D_k^T A D_k y + y^T D_k^T \nabla\phi(x_0).$$

Taking the gradient, the solution satisfies $D_k^T A D_k y_k = -D_k^T \nabla\phi(x_0)$. The matrix $D_k^T A D_k$ is diagonal due to the conjugacy of the columns of D_k, so the solution

$$y_k^i = -\frac{\langle d_i, \nabla\phi(x_0)\rangle}{\langle d_i, d_i\rangle_A}, \quad \text{for } i = 0, \ldots, k-1.$$

Each element y_k^i depends only on one conjugate direction d_i, thus with an additional conjugate direction $d_k \in \mathcal{K}_{k+1}$, the the minima of $\phi(x)$ restricted to $\mathcal{K}_{k+1}$ is

$$\begin{aligned} x_{k+1} &= x_0 + D_k y_k - \frac{\langle d_k, \nabla\phi(x_0)\rangle}{\langle d_k, d_k\rangle_A} d_k \\ &= x_k - \operatorname{proj}_{d_k}^A \nabla\phi(x_0), \end{aligned} \tag{2}$$

where $\operatorname{proj}_d^A := d(d^T A d)^{-1} d^T$. The matrix $\operatorname{proj}_d^A A$ is an oblique projection onto the space spanned by d, and it's the solution to

$$\operatorname{proj}_d^A Av = \arg \min_{y \in \operatorname{span}\{d\}} \|v - y\|_A.$$

Conjugate directions allow us build upon previous Krylov restricted solution x_k by minimizing along d_k to find the next solution x_{k+1}. Though (2) is correct, it is not the standard update formula used in the Conjugate Gradient algorithms. Rather, as $\phi(x)$ is a quadratic function the solution to $\arg\min_\alpha \phi(x_k + \alpha d_k)$ is achieved with

$$\alpha_k := -\frac{\langle \nabla\phi(x_k), d_k\rangle}{\langle d_k, d_k\rangle_A}.$$

Therefore

$$\begin{aligned} x_{k+1} &= x_k - \frac{\langle \nabla\phi(x_k), d_k\rangle}{\langle d_k, d_k\rangle_A} d_k \\ &= x_k - \operatorname{proj}_{d_k}^A \nabla\phi(x_k). \end{aligned} \tag{3}$$

The projection is a handy tool, as it is highlights that x_{k+1} results from a minimization problem. The well known properties of projections can be also be called upon, for instance, the above step from x_k to x_{k+1} is invariant under scaling d_k.

But how to obtain a basis of A-conjugate directions? The same way we obtain any orthogonal basis: use Gram-Schmidt procedure with the inner product $\langle , \rangle_A$. Starting with a given set of linearly independent vectors $r_0, \ldots, r_k \in \mathbb{R}^n$ that span $\mathcal{K}_{k+1}$ and conjugate directions $d_0, \ldots, d_{k-1}$ that span $\mathcal{K}_k$, the next conjugate direction d_k can be obtained by "projecting out" the components of r_k that are in $\mathcal{K}_k$ using the Gram–Schmidt process,

where $r_0 = d_0$ and

$$\begin{aligned} d_k &= r_k - \sum_{j=0}^{k-1} \frac{\langle r_k, d_j \rangle_A}{\langle d_j, d_j \rangle_A} d_j \\ &= r_k - \sum_{j=0}^{k-1} \frac{1}{\langle d_j, d_j \rangle_A} d_j p_j^T A r_k \\ &= (I - \sum_{j=0}^{k-1} \text{proj}_{d_j}^A A) r_k. \end{aligned}$$

This is a computationally expensive way of obtaining d_k and would be significantly cheaper if the inner product of r_k with most of the Qd_j vectors are zero. **Selecting:** $r_k = -\nabla\phi(x_k) = b - Ax_k$ does just that, for x_k is the minima restricted to $\mathcal{K}_k$, thus $r_k \in \mathcal{K}_k^\perp \subset (A\mathcal{K}_{k-1})^\perp$ and r_k is orthogonal to each $\{Ad_0, \ldots, Ad_{k-1}\}$. Now calculating the next conjugate direction is reduced to performing a single projection

$$d_k = (I - \text{proj}_{d_{k-1}}^A A) r_k. \tag{4}$$

The vectors r_k are known as the residuals, as $\|r_k\|$ is a measure how close we are to optimality, where $\|r_k\| = 0$ signifies we have encountered the solution. The vectors d_k are known as the search directions.

Finally, to efficiently calculate the r_k gradients, we use

$$\nabla\phi(x_k + \alpha_k d_k) - \nabla\phi(x_k) = r_k - r_{k+1} = \alpha_k A d_k.$$

This to can be written as a projection and allows for the calculating of the r_k's iteratively with

$$\begin{aligned} r_{k+1} &= r_k - \alpha_k A d_k \\ &= r_k - A d_k \frac{d_k^T r_k}{\langle d_k, d_k \rangle_A} \\ &= (I - A\text{proj}_{d_k}^A) r_k. \end{aligned} \tag{5}$$

Collecting the updates of x_k (3), d_k (4) and r_k (5) we have the Conjugate Gradient Algorithm (1.1). The calculation of the conjugate search directions can also be written in a single formula

$$d_{k+1} = (I - \text{proj}_{d_k}^A A)(I - A\text{proj}_{d_k}^A) r_k.$$

Input: $r_0 = p_0 = \nabla\phi(x_0)$, $k = 0$ and tolerance $\epsilon \in \mathbb{R}_+$.

repeat

  $r_{k+1} = (I - A\text{proj}_{d_k}^A)r_k$

  $x_{k+1} = x_k + \text{proj}_{d_k}^A r_k$

  $d_{k+1} = (I - \text{proj}_{d_k}^A A)r_{k+1}$

  $k = k + 1$

until $\|r_k\| \geq \epsilon\|r_0\|$

Output: x_k.

Algorithm 1.1: CG: Conjugate Gradient Method

1.1 Affine restrictions and Preconditioning

How do we minimize $\phi(x)$ but restricted to an affine space $x_0 \oplus Z$, where $x_0 \in \mathbb{R}^n$ and $Z \subset \mathbb{R}^n$ is a subspace? This situation arises when we have to minimize $\phi(x)$ subject to linear constraints, say, $Bx = d$ where $B \in \mathbb{R}^{m\times n}$ and $d \in \mathbb{R}^m$.

Let $P : \mathbb{R}^n \to Z$ be any surjective linear function. Our restricted problem is equivalent to

$$\min_{x\in x_0\oplus P\mathbb{R}^n} \phi(x) = \min_{\bar{x}\in\mathbb{R}^n} \bar{x}^T P^T AP\bar{x}/2 + \bar{x}^T P^T \nabla\phi(x_0) =: \bar{\phi}(\bar{x}), \tag{6}$$

We could now apply the standard CG method to $\bar{\phi}(\bar{x})$ to obtain a convergent sequence $\{\bar{x}_k\}_k$, but for most applications of interest, P is only known implicitly and calculating $P^T AP$ is expensive. Thus we will examine what happens when we apply the CG method to $\bar{\phi}(\bar{x})$, then try to lift these results to the untransformed space $x_k = P\bar{x}_k + x_0$. Let $\bar{d}_k$ and $\bar{r}_k$ be the search directions and residuals at iteration k of the CG method applied to $\bar{\phi}(\bar{x})$, and $\bar{x}_0 = 0$ the initial point. The CG method is initiated with $\bar{r}_0 = -\nabla\bar{\phi}(0)$ and $\bar{p}_0 = \bar{r}_0$. Left multiplying the $\bar{x}_k$ update (3) by P and summing x_0 to both sides

$$\begin{aligned} x_{k+1} &= P\bar{x}_{k+1} + x_0 \\ &= P\bar{x}_k + x_0 + P\text{proj}_{\bar{d}_k}^{P^T AP}\bar{r}_k \\ &= x_k + P\bar{d}_k(\bar{d}_k^T P^T AP\bar{d}_k)^{-1}\bar{d}_k^T\bar{r}_k. \end{aligned}$$

Suggestively, defining $d_k = P\bar{d}_k$ and $\bar{r}_k = P^T r_k$ allows us to apply a standard

CG step in x_k

$$\begin{aligned} x_{k+1} &= x_k + d_k(d_k^T A d_k)^{-1}\overline{d}_k^T P^T r_k \\ &= x_k + \text{proj}_{d_k}^A r_k. \end{aligned}$$

The r_k's are also residuals in the untransformed space

$$\begin{aligned} P^T r_k &= \overline{r}_k \\ &= -P^T(AP\bar{x}_k + \nabla\phi(x_0)) \\ &= -P^T(\nabla\phi(x_0 + P\bar{x}_k)). \end{aligned}$$

The d_k's and r_k's have some of the same properties as they would in the standard CG method. The d_k's are $A-$conjugate

$$d_k^T A d_i = \overline{d}_k^T P^T AP\overline{d}_i = 0, \quad \text{if } i < k.$$

The r_k's are orthogonal to previous search directions

$$\begin{aligned} r_k^T d_i &= r_k^T P\overline{d}_i \\ &= \overline{r}_k^T\overline{d}_i = 0, \quad \text{for } i < k. \end{aligned}$$

In contrast to standard CG properties, the r_k's are mutually PP^T-conjugate instead of simply orthogonal

$$r_k^T PP^T r_i = \overline{r}_k^T\overline{r}_i = 0, \quad \text{for } i < k. \tag{7}$$

To calculate the d_k's, left multiplying by P the update to $\overline{d}_k$

$$\begin{aligned} P\overline{d}_{k+1} &= P(I - \text{proj}_{\overline{d}_k}^{P^T AP} P^T AP)\overline{r}_{k+1} \\ &= (I - P\text{proj}_{\overline{d}_k}^{P^T AP} PA)P\overline{r}_{k+1} \\ &= (I - \text{proj}_{d_k}^A A)PP^T r_k. \end{aligned}$$

Thus by substituting line 4 in the Conjugate Gradients Algorithm 1.1 for

$$d_{k+1} = (I - \text{proj}_{d_k}^A A)PP^T r_k, \tag{8}$$

results in the *PCG* method (the projected conjugate gradients method) in Algorithm 1.2. The stopping criteria is now based on a relative decrease of

Input: $PP^T \in \mathbb{R}^{n\times n}, r_0 = p_0 = PP^T\nabla\phi(x_0), k = 0$ and tolerance $\epsilon \in \mathbb{R}_+$.

repeat

  $r_{k+1} = (I - A\text{proj}_{d_k}^A)r_k$

  $x_{k+1} = x_k + \text{proj}_{d_k}^A r_k$

  $d_{k+1} = (I - \text{proj}_{d_k}^A A)PP^T r_{k+1}$

  $k = k + 1$

until $\|r_k\|_{PP^T} \geq \epsilon\|r_0\|_{PP^T}$

Output: x_ρ.

Algorithm 1.2: The PCG method: Projected Conjugate Gradients

the projected residual. It would not do to use the residual as a stopping criteria, as there is no longer any guarantee of $\|r_k\|$ converging to zero!

As a single equation

$$d_{k+1} = (I - \text{proj}_{d_k}^A A)PP^T(I - A\text{proj}_{d_k}^A)r_k$$

which is remarkably similar to the BFGS search direction [2], where PP^T is a given previous approximation to a Hessian matrix, then the next BFGS search direction is

$$d_{k+1}^{BFGS} = -\text{proj}_{d_k}^A r_k + (I - \text{proj}_{d_k}^A A)PP^T(I - A\text{proj}_{d_k}^A)r_k.$$

In the case that $S = \mathbb{R}^n$, then P is a change of coordinates and is chosen to precondition the system. When Z is a proper subspace, P acts as a submersion. In either case, PP^T can be chosen to improve the spectral properties of P^TAP because $\lambda(P^TAP) \cup \{0\} = \lambda(PP^TA)$, where $\lambda(A)$ denotes the set of eigenvalues of A. The number of iteration required by the Conjugate Gradient Algorithm 1.1 to reach the exact solution is equal to, at most, the number of distinct eigenvalues in $\lambda(P^TAP)$. Thus ideally we would like PP^T to be an approximate inverse of A so that PP^TA has eigenvalues concentrated around one, and PP^T is referred to a the preconditioning matrix. Often the notation is $M^{-1} = PP^T$, emphasizing that it is some sort of estimate of the inverse.

When Z is a proper subspace, slightly abusing notation, let $Z \in \mathbb{R}^{n\times q}$ be a matrix who's columns form a basis of the subspace Z. As P is a linear submersion, it must be of the form $P = ZH$ where $H \in \mathbb{R}^{q\times q}$ is nonsingular. As Z^TAZ is also nonsingular

$$\lambda(P^TAP) = \lambda(H^TZ^TAZH) = \lambda(HH^TZ^TAZ).$$

Thus HH^T should be chosen to act as a preconditioner of the matrix Z^TAZ. For this special matrix, precondtioners of the form $HH^T = (Z^TGZ)^{-1} \approx (Z^TAZ)^{-1}$ have shown to be successful [**Keller2000**], where $G \in \mathbb{R}^{n\times n}$. In this case $PP^T = Z(Z^TGZ)^{-1}Z = \text{proj}_Z^G$ which is similar to an oblique projection, in that $PP^T(GZ) = Z$. Note that in this development, nor A nor G need be positive definite, but instead, only Z^TAZ and Z^TGZ need be positive definite.

1.1.1 Linear constrained case

Let $x_0 \oplus Z = \{x \,|\, Bx = d\}$. The initial point x_0 is feasible point: $Bx_0 = d$. The objective is to minimize $\phi(x)$ restricted to $x_0 \oplus \text{Ker}(B)$.

Note that to execute Algorithm 1.2, we do not need the whole matrix PP^T, but instead, we only need to know how to apply PP^T to the residual vectors r_k. As shown by Gould, Hribar and Nocedal [3], when the affine subspace is defined by a linear constraint $Bx = d$, we can implicitly apply $PP^T = \text{proj}_Z^G$ without calculating $Z = \text{Ker}(B)$. In fact, as $PP^Tr = PP^TGG^{-1}r$, which is a projection of $G^{-1}r$ given by

$$\begin{aligned} PP^Tr &= \arg\min_{x\in Z} \frac{1}{2}\|x - G^{-1}r\|_G^2 \\ &= \arg\min_{Bx=0} \frac{1}{2}x^TGx - x^Tr. \end{aligned}$$

To solve the above, we form the associated Lagrangian

$$L(x,\mu) = \frac{1}{2}x^TGx - x^Tr + x^TB^T\mu,$$

where $\mu \in \mathbb{R}^m$. Differentiating in μ and x

$$\begin{bmatrix} G & B^T \\ B & 0 \end{bmatrix} \begin{bmatrix} x \\ \mu \end{bmatrix} = \begin{bmatrix} r \\ 0 \end{bmatrix}. \tag{9}$$

This system, also known as a KKT system, can be solved in a number of efficient ways so long as G is positive definite and by factorizing B, see [5]. Of course, one could model our original problem (6) as a KKT system. But if A is not positive definite, but only $Z^TAZ \succ 0$, then solving the KKT system is expensive. In this case, using the Projected CG Algorithm 1.2 where PP^Tr_{k+1} is calculated by solving (9) becomes worthwhile.

1.2 Restarting with remembering preconditioners

In certain circumstances, it can be advantageous to stop the CG iterations and restart on a given point $\bar{x}_0$. In other words, apply the standard CG method to

$$A(x - \bar{x}_0) = b \Leftrightarrow Ax = b + A\bar{x}_0 =: \bar{b}.$$

This can switch the search to a different set of Krylov subspaces. We have used this property in an article "Action constrained quasi-Newton methods". Let $d_0 \dots d_p$ and $r_0, \dots, r_p$ be a set of conjugate and residual directions. Suppose we have a point $\bar{x}_0 \in \mathbb{R}^n$ with a gradient $\bar{r}_0 = \nabla\phi(\bar{x}_0)$ such that $\bar{r}_0 \in \mathcal{K}_{p+1}^{\perp}$ =span$\{r_0, \dots, r_p\}$. Furthermore, suppose we have a symmetric preconditioner M^{-1} that "remembers" all previous conjugate directions

$$d_j^T A M^{-1} = d_j^T, \quad \text{for } j = 1, \dots, p. \tag{10}$$

In quasi-Newton methods, this property is referred to the quadratic hereditary property and is satisfied by all metric matrices of the Broyden family after p iterations [1] on convex quadratic functions. By then starting the PCG iterations with $\bar{r}_0$ and $\bar{d}_0 = M^{-1}\bar{r}_0$ we generate vectors $\bar{d}_0, \dots, \bar{d}_q$ such that $d_0 \dots d_{p-1}, \bar{d}_0, \dots, \bar{d}_q$ are a conjugate set.

proof: The proof is by induction on $d_j^T A\bar{d}_i = 0$ and $d_j^T \bar{r}_i = 0$ for $1 \le j \le p$ and $1 \le i \le q$. For $i = 0$, the spectral properties of M^{-1} and $\bar{r}_0 \in \mathcal{K}_{p+1}^{\perp}$ guarantee that

$$\begin{aligned} d_j^T A\bar{d}_0 &= d_j^T A M^{-1}\bar{r}_0 \\ &= d_j^T \bar{r}_0 = 0, \quad \text{for } j = 1, \dots, p. \end{aligned}$$

Supposing the induction hypothesis is true for i and all $0 \le j \le p$, using (5) to calculate the next residual $\bar{r}_{i+1}$, then by induction

$$\begin{aligned} d_j^T \bar{r}_{i+1} &= d_j^T (I - A\text{proj}_{\bar{d}_i}^A)\bar{r}_i \\ &= d_j^T \bar{r}_i \quad \big(\textbf{by induction } d_j^T A\bar{d}_i = 0\big) \\ &= 0. \quad \big(\textbf{by induction } d_j^T \bar{r}_i = 0\big) \end{aligned}$$

Using (8) to substitute $\bar{d}_{i+1}$

$$\begin{aligned} d_j^T A\bar{d}_{i+1} &= d_j^T A(I - \text{proj}_{\bar{d}_i}^A A)M^{-1}\bar{r}_{i+1} \\ &= d_j^T AM^{-1}\bar{r}_{i+1} \quad \big(\textbf{by induction } d_j^T A\bar{d}_i = 0\big) \\ &= d_j^T \bar{r}_{i+1} \quad \textbf{by } (10) \\ &= 0. \quad \square \end{aligned}$$

The conjugate directions that result from using a remembering preconditioner are not necessary the same as those generated by the CG method. Though they still serve as search directions that can be used to iteratively minimize $\phi(x)$ as in Algorithm 1.3. On iteration k, the subroutine `update_remember` on line 7 updates the preconditioner M^{-1} so (10) holds for all search directions $d_i, i = 1, \ldots k$.

Input: $M^{-1} \in \mathbb{R}^{n\times n}, r_0 = p_0 = M^{-1}\nabla\phi(x_0), k = 0, m = 1, M \in \mathbb{N}$, $\epsilon \in \mathbb{R}_+, \mathcal{D} = \{\emptyset\}, \mathcal{AD} = \{\emptyset\}$

1 **repeat**
2 $\quad \alpha_k = \dfrac{\langle r_k, r_k\rangle_{M^{-1}}}{\langle d_k, d_k\rangle_A}$
3 $\quad r_{k+1} = r_k - \alpha_k A d_k$
4 $\quad x_{k+1} = x_k + \alpha_k d_k$
5 $\quad \mathcal{D} \leftarrow d_{k+1}$, $\mathcal{AD} \leftarrow Ad_{k+1}$
6 $\quad$ **if** $m = M$ **then**
7 $\quad\quad M^{-1} =$`update_remember`$(M^{-1}, \mathcal{D}, \mathcal{AD})$
8 $\quad\quad d_{k+1} = M^{-1}r_{k+1}$
9 $\quad\quad m = 0$, $\mathcal{D} = \{\emptyset\}$, $\mathcal{AD} = \{\emptyset\}$
10 $\quad$ **else**
11 $\quad\quad d_{k+1} = M^{-1}r_{k+1} + \dfrac{\langle r_{k+1}, r_{k+1}\rangle_{M^{-1}}}{\langle r_k, r_k\rangle_{M^{-1}}} d_k$
12 $\quad$ **end**
13 $\quad k = k + 1, m = m + 1$
14 **until** $\|r_k\| \geq \epsilon\|r_0\|$

Output: x_{p-1}.

Algorithm 1.3: Restarting Conjugate Gradients method

2 Implementing

We focus on implementing the PCG Algorithm 1.2 as this covers all cases, in that, with $PP^T = I$ it is the standard CG method, with $PP^T = M^{-1}$, where M^{-1} is an approximate inverse, it is the Preconditioned CG method.

First we say goodbye to our projection notation, for $v \in \mathbb{R}^n$

$$\text{proj}_{d_k}^A v = \frac{d_k^T v}{\langle d_k, d_k \rangle_A} d_k.$$

To further simplify computations, we need two identities. For the first, recall that $r_k \in \mathcal{K}_{k-1}^{\perp}$ thus using (8)

$$\begin{aligned} \langle r_k, d_k \rangle &= \left\langle r_k, (I - \text{proj}_{d_{k-1}}^A A) P P^T r_k \right\rangle \\ &= \langle r_k, r_k \rangle_{PP^T} - \underbrace{\langle r_k, d_{k-1} \rangle}_{=0} \frac{d_{k-1}^T P P^T r_k}{\langle d_{k-1} d_{k-1} \rangle_A} \\ &= \langle r_k, r_k \rangle_{PP^T}. \end{aligned} \tag{11}$$

This identity applied to the r_k update (5) and x_k update (3) results in lines 3 and 4 of the implemented PCG method in Algorithm 2.1.

The second identity relies rearranging (5) to see that

$$\begin{aligned} r_{k+1} &= r_k - \frac{\langle r_k, d_k \rangle}{\langle d_k, d_k \rangle_A} A d_k \\ &= r_k - \frac{\langle r_k, r_k \rangle_{PP^T}}{\langle d_k, d_k \rangle_A} A d_k. \end{aligned} \tag{12}$$

Equivalently

$$A d_k = \frac{r_k - r_{k+1}}{\alpha_k},$$

where $\alpha_k = \frac{\langle r_k, r_k \rangle_{PP^T}}{\langle d_k, d_k \rangle_A}$. Using this to substitute $A d_k$ and recalling that $\langle r_k, r_{k+1} \rangle_{PP^T} = 0$ (see (7)), the second identity is

$$\begin{aligned} \frac{\langle r_{k+1}, A d_k \rangle_{PP^T}}{\langle d_k, d_k \rangle_A} &= -\frac{\langle r_{k+1}, r_{k+1} \rangle_{PP^T}}{\langle d_k, d_k \rangle_A} \left(\frac{\langle d_k, d_k \rangle_A}{\langle r_k, r_k \rangle_{PP^T}} \right) \\ &= -\frac{\langle r_{k+1}, r_{k+1} \rangle_{PP^T}}{\langle r_k, r_k \rangle_{PP^T}}. \end{aligned} \tag{13}$$

This identity

$$\begin{aligned} d_{k+1} &= (I - \text{proj}^A_{d_k} A) PP^T r_k \\ &= PP^T r_k + \frac{\langle r_{k+1}, r_{k+1} \rangle_{PP^T}}{\langle r_k, r_k \rangle_{PP^T}} d_k. \end{aligned} \tag{14}$$

The search directions are updated using the above on line (6) of Algorithm (2.1).

Input: $PP^T \in \mathbb{R}^{n \times n}, r_0 = p_0 = PP^T \nabla \phi(x_0), k = 0$ and tolerance $\epsilon \in \mathbb{R}_+$.

1 **repeat**

2 $\quad \alpha_k = \dfrac{\langle r_k, r_k \rangle_{PP^T}}{\langle d_k, d_k \rangle_A}$

3 $\quad r_{k+1} = r_k - \alpha_k A d_k$

4 $\quad x_{k+1} = x_k + \alpha_k d_k$

5 $\quad d_{k+1} = PP^T r_{k+1} + \dfrac{\langle r_{k+1}, r_{k+1} \rangle_{PP^T}}{\langle r_k, r_k \rangle_{PP^T}} d_k$

6 $\quad k = k + 1$

7 **until** $\|r_k\|_{PP^T} \geq \epsilon \|r_0\|_{PP^T}$

Output: x_{p-1}.

Algorithm 2.1: Implemented Projected Conjugate Gradients

References

[1] C. Broyden. "Quasi-Newton methods and their application to function minimisation". In: *Mathematics of Computation* (1967), pp. 368–381. URL: `http://www.jstor.org/stable/2003239`.

[2] B. R. Fletcher and M. J. D. Powell. "A rapidly convergent descent method for minimization". In: *The Computer Journal* 6.2 (1963), pp. 163–168.

[3] N. I. M. Gould and M. E. Hribar. "On the Solution of Equality Constrained Quadratic Programming Problems Arising in Optimization". In: (2000).

[4] M. R. Hestenes and E. Stiefel. "Methods of Conjugate Gradients for Solving Linear Systems". In: *Journal of research of the National Bureau of Standards* 49.6 (1952).

[5] J Nocedal and S. J. Wright. *Numerical Optimization*. Ed. by P. Glynn and S. M. Robinson. Vol. 43. Springer Series in Operations Research 2. Springer, 1999. Chap. 5, pp. 164–75. ISBN: 0387987932. DOI: 10.1002/lsm.21040. URL: http://www.ncbi.nlm.nih.gov/pubmed/21643320.

[6] J. R. Shewchuk. *An Introduction to the Conjugate Gradient Method Without the Agonizing Pain*. Tech. rep. School of Computer Science Carnegie Mellon University, 1994.