Derivation of the Conjugate Gradient Method

Olof Runborg

June 1, 2011

1 Goal

We want to find \( x_k \in K_k(b, A) \) which minimizes \( \|Ax_k - b\|_{A^{-1}} \) when \( A \) is a symmetric positive definite matrix. Normally one would need to first compute a basis for \( K_k(b, A) \) and then do a least squares fit to find the solution. However, for symmetric positive definite matrices we can do it iteratively with the conjugate gradient method.

1.1 Ansatz for the method

We let

\[
x_{k+1} = x_k + \alpha_{k+1}p_{k+1}, \quad x_0 = 0, \quad p_1 = b,
\]

with \( \alpha_k \) and search directions \( p_k \) to be determined such that the iterates \( x_k \) belong to \( K_k(b, A) \) and minimize \( \|Ax - b\|_{A^{-1}} \). Moreover, we want to be able to compute \( \alpha_k \) and \( p_k \) cheaply from quantities available from the previous iteration.

By simple induction

\[
x_k = \alpha_1 p_1 + \cdots + \alpha_k p_k =: S_k a_k, \quad k \geq 1,
\]

where

\[
S_k = [p_1 \ p_2 \cdots p_k], \quad a_k = [\alpha_1 \ \alpha_2 \cdots \ \alpha_k]^T.
\]

We also define the residuals

\[
r_k = Ax_k - b,
\]

and note that after multiplying (1) by \( A \) and subtracting \( b \) we get

\[
r_{k+1} = r_k + \alpha_{k+1}Ap_{k+1}, \quad r_0 = -b.
\]

The method stops if \( r_k = 0 \) for some \( k \). We say that the method fails if \( p_k = 0 \) for some \( k \). We will assume for now that the method has not failed before iteration \( k \), i.e. that \( p_{\ell} \neq 0 \) for \( \ell \leq k \). At the end we will then show that this implies that it will not fail in iteration \( k + 1 \). Since \( p_1 = b \neq 0 \) it follows by induction that the method will never fail.
1.2 Choice of $p_k$-directions

Since we look for $x_k \in \mathcal{K}_k(b, A)$ and $x_k = S_k a_k$ we want to pick $p_k$ such that $\text{range}(S_k) = \mathcal{K}_k(b, A)$. We will show here that this follows if we take

$$p_{k+1} = r_k - AS_k z_k,$$

where $z_k$ is chosen such that it minimizes $||p_{k+1}||_2$. It is thus the least squares solution given by the normal equations

$$S_k^T A^T S_k z_k = S_k^T A r_k.$$

Note that in the steepest descent method we take just $p_{k+1} = r_k$. The subtraction of $AS_k z_k$ makes the search much more efficient. The construction has several interesting consequences.

1. The $p_k$-directions are $A$-conjugate (or just conjugate), which means that

$$p_k^T A p_\ell = 0, \quad k \neq \ell.$$

This follows from the fact that $S_k^T A^T p_{k+1} = S_k^T A^T (r_k - AS_k z_k) = 0$ for all $k$. It also implies that

$$S_k^T A S_k = D_k, \quad D_k \text{ is diagonal with elements } d_\ell = p_\ell^T A p_\ell.$$

2. If the method has not failed before iteration $k$ then

$$\text{range}(S_k) = \text{span}\{p_1, \ldots, p_k\} = \text{span}\{r_0, \ldots, r_{k-1}\} = \mathcal{K}_k(b, A).$$

Moreover, $\{p_j\}_{j=1}^{k}$ and $\{r_j\}_{j=0}^{k-1}$ are linearly independent.

By (2) we have $A p_k \in \text{span}\{r_k, r_{k-1}\}$ and therefore

$$\text{range}(AS_k) = \text{span}\{A p_1, \ldots, A p_k\} \subset \text{span}\{r_0, \ldots, r_k\}.$$

Hence, from (3), we get that $p_k \in \text{span}\{r_0, \ldots, r_{k-1}\}$ for all $k$, and then

$$\text{span}\{p_1, \ldots, p_k\} \subset \text{span}\{r_0, \ldots, r_{k-1}\}.$$

Since $A$ is symmetric positive definite, $D_k$ has only positive entries, unless $p_\ell = 0$ for some $\ell \leq k$. This means that if the method has not failed before iteration $k$, the directions $p_1, \ldots, p_k$ are linearly independent by (6). (Otherwise there would be a vector $x \neq 0$ such that $S_k x = 0$ and then $D_k x = 0$, a contradiction.) Hence, the left hand side space in (7) has dimension $k$, which is therefore also the dimension of the right hand side. Then $r_0, \ldots, r_{k-1}$ are linearly independent and the spaces must be equal.

Finally, we can show that the space is in fact equal to the Krylov space $\mathcal{K}_k(b, A)$. We use induction. Clearly $\text{span}\{r_0\} = \mathcal{K}_0(b, A)$. Suppose $\text{span}\{p_1, \ldots, p_{k-1}\} = \mathcal{K}_{k-1}(b, A)$. Then, for $\ell \leq k - 1$ we have $r_{\ell-1} \in \mathcal{K}_{k-1}(b, A)$ and by (2)

$$r_\ell \in \text{span}\{r_{\ell-1}, A p_\ell\} \subset \text{span}\{\mathcal{K}_{k-1}(b, A), A b, \ldots, A^{k-1} b\} \subset \mathcal{K}_k(b, A).$$

But since the residuals $\{r_\ell\}$ are linearly independent we must have equality, $\text{span}\{r_0, \ldots, r_{k-1}\} = \mathcal{K}_k(b, A)$. 

2
1.3 Simplified minimization problem

Since \( \{p_j\} \) span the Krylov space \( K_k(b, A) \) we can simplify the minimization problem

\[
\min_{x_k \in K_k(b, A)} \|Ax_k - b\|_{A^{-1}} \Rightarrow \min_{a_k \in \mathbb{R}^k} \|AS_k a_k - b\|_{A^{-1}}. 
\]

The solution to this weighted least squares problem is given by minimizing

\[
(AS_k a - b)^T A^{-1} (S_k a - b) = a^T S_k^T AS_k a - 2a^T S_k^T b + b^T A^{-1} b,
\]

over \( a \), which has the solution \( a_k \) satisfying

\[
S_k^T AS_k a_k = S_k^T b \Rightarrow D_k a_k = S_k^T b.
\]

Hence, \( a_k = p_k^T b / p_k^T A p_k \). By also using (5) we get

\[
p_k^T b = p_k^T (Ax_k - r_{k-1}) = p_k^T (AS_{k-1} a_{k-1} - r_{k-1}) = -p_k^T r_{k-1},
\]

which shows that

\[
\alpha_k = \frac{-p_k^T r_{k-1}}{p_k^T A p_k}.
\]

We also have

\[
S_k^T r_k = S_k^T (Ax_k - b) = 0 \Rightarrow p_k^T r_k = 0, \quad \ell \leq k.
\]

This implies that

\[
r_k \perp K_k(b, A),
\]

and, by induction,

\[
r_k \perp r_{\ell}, \quad k \neq \ell.
\]

Hence

1. The residual at iteration \( k \) is orthogonal to the search space \( K_k(b, A) \), cf. Galerkin methods,

2. The previously computed residuals form an orthogonal basis for \( K_k(b, A) \).

Consequently, if \( p_k^T r_{k-1} = 0 \) then \( p_k \in K_{k-1}(b, A) \) which is impossible since \( \{p_k\} \) are linearly independent. Therefore \( \alpha_k \neq 0 \).

1.4 Simplified computation of \( p_{k+1} \)

Finding \( p_{k+1} \) involves solving the least squares problem (4), where \( z_k \) depends on all previous \( p_k \)-directions via the \( S_k \) matrix. We will now show that this can be simplified since \( p_{k+1} \in \text{span}\{r_\ell, p_\ell\} \):

\[
z_k = \gamma_k \begin{pmatrix} w_k \\ \alpha_k \end{pmatrix}, \quad w_k \in \mathbb{R}^{k-1},
\]

3
and with $\alpha_k \neq 0$ as defined above. Then
\[
p_{k+1} = r_k - \gamma_k AS_{k-1} w_k - \gamma_k \alpha_k A p_k
= r_k - \gamma_k AS_{k-1} w_k - \gamma_k (r_k - r_{k-1})
= (1 - \gamma_k) r_k + \gamma_k (r_{k-1} - AS_{k-1} w_k).
\]
We note that $r_k \perp (AS_{k-1} w_k - r_{k-1})$ for any $w_k$, since range($AS_{k-1}$) $\subset K_k(b, A)$. Therefore,
\[
||p_{k+1}||^2 = (1 - \gamma_k)^2 ||r_k||^2 + \gamma_k^2 ||r_{k-1} - AS_{k-1} w_k||^2.
\tag{8}
\]
In order to minimize this over $z_k$, i.e. over $\gamma_k$ and $w_k$, we see that $w_k$ must be the minimizer of $||r_{k-1} - AS_{k-1} w_k||$. Hence, $w_k = z_{k-1}$ and $r_{k-1} - AS_{k-1} w_k = p_k$. This shows the claim and we have
\[
p_{k+1} = (1 - \gamma_k) r_k + \gamma_k p_k = (1 - \gamma_k) (r_k + \beta_k p_k), \quad \beta_k = \frac{\gamma_k}{1 - \gamma_k}.
\]
From the fact that $p_{k+1}^T A p_k = 0$ by (5) we get an expression for $\beta_k$,
\[
\beta_k = -\frac{r_k^T A p_k}{p_k^T A p_k}.
\]
It is also possible to derive an expression for $\gamma_k$. By minimizing (8) we get
\[
\gamma_k = \frac{||r_k||^2}{||r_k||^2 + ||p_k||^2}.
\]
However, as we will see below, $\gamma_k$ is actually not needed in the final optimized algorithm.

### 1.5 Non-failure

We have left to show that the method does not fail. Recall that in Section 1.1 we assumed that $p_\ell \neq 0$ and $r_\ell \neq 0$ for $0 \leq \ell \leq k$. The results we have shown so far therefore only holds upto $\ell = k$. We can now finish the induction argument by noting that whenever $p_k \neq 0$ and $r_k \neq 0$ we must have $p_{k+1} \neq 0$ by (8). The results in the previous sections therefore hold for all $\ell$, until $r_\ell = 0$ and the method stops.

### 1.6 Basic algorithm

We summarize the basic algorithm that follows from the considerations above:

1. Given $r_k, p_k, x_k$.
2. Compute $\beta_k = -\frac{r_k^T A p_k}{p_k^T A p_k}, \quad \gamma_k = \frac{||r_k||^2}{||r_k||^2 + ||p_k||^2}$. 

4
3. Update $p_{k+1} = (1 - \gamma_k)(r_k + \beta_k p_k)$.

4. Compute $\alpha_{k+1} = \frac{p_{k+1}^T r_k}{p_{k+1}^T A p_{k+1}}$

5. Update $x_{k+1} = x_k + \alpha_{k+1} p_{k+1}$

6. Update $r_{k+1} = r_k + \alpha_{k+1} A p_{k+1}$

7. Iterate.

This can be implemented such that only one matrix vector multiply is used per iteration. We will see this more clearly in the next section.

1.7 Optimized algorithm

We make a few observations which simplify the algorithm. First, if we define

$$p_k = (1 - \gamma_k)\hat{p}_k,$$

$$\hat{\alpha}_k = \frac{\hat{p}_k^T b}{\hat{p}_k^T A \hat{p}_k},$$

$$\hat{\beta}_k = -\frac{\hat{p}_k^T A p_k}{\hat{p}_k^T A \hat{p}_k},$$

then

$$\hat{\alpha}_k \hat{p}_k = \alpha_k p_k, \quad \hat{\beta}_k \hat{p}_k = \beta_k p_k.$$  

We can therefore use $(\hat{p}_k, \hat{\alpha}_k, \hat{\beta}_k)$ instead of $(p_k, \alpha_k, \beta_k)$ in the algorithm, avoiding $\gamma_k$ entirely. Moreover,

$$r_k^T r_k = r_k^T (r_{k-1} + \alpha_k A \hat{p}_k) = \alpha_k r_k^T A \hat{p}_k,$$

and

$$\hat{p}_k^T r_{k-1} = (r_{k-1} + \beta_{k-1} \hat{p}_{k-1})^T r_{k-1} = r_{k-1}^T r_{k-1}.$$

Hence,

$$\frac{r_k^T r_k}{r_{k-1}^T r_{k-1}} = \frac{\hat{\alpha}_k r_k^T A \hat{p}_k}{r_k^T A \hat{p}_k} = -\frac{\hat{\beta}_k}{\hat{p}_k^T A \hat{p}_k} = \hat{\beta}_k.$$

This leads to an algorithm with one matrix vector multiply, two inner products and three saxpy per iteration:

1. Given $r_k, \hat{p}_k, x_k$ and $s_{k-1} = r_{k-1}^T r_{k-1}$

2. Compute $s_k = r_k^T r_k$  \hspace{1cm} (inner product)

3. Compute $\hat{\beta}_k = \frac{s_k}{s_{k-1}}$  \hspace{1cm} (scalar)

4. Update $\hat{p}_{k+1} = r_k + \hat{\beta}_k \hat{p}_k$  \hspace{1cm} (saxpy)

5. Compute $q_{k+1} = A \hat{p}_{k+1}$  \hspace{1cm} (matrix vector multiply)

6. Compute $\hat{\alpha}_{k+1} = \frac{s_k}{\hat{p}_{k+1}^T q_{k+1}}$  \hspace{1cm} (scalar, inner product)

7. Update $x_{k+1} = x_k + \hat{\alpha}_{k+1} \hat{p}_{k+1}$  \hspace{1cm} (saxpy)

8. Update $r_{k+1} = r_k + \hat{\alpha}_{k+1} q_{k+1}$  \hspace{1cm} (saxpy)

9. Iterate.