3-2008

PRECONDITIONED CONJUGATE GRADIENT ALGORITHM

Elena Caraba

PRECONDITIONED CONJUGATE GRADIENT ALGORITHM

by

Elena Caraba

Undergraduate honors thesis under the direction of

Dr. Susanne Brenner and Dr. Li-Yeng Sung

Department of Mathematics

Submitted to LSU Honors College in partial fulfillment

of the Upper Division Honors Program

March 2008

Louisiana State University

& Agricultural and Mechanical College

Baton Rouge, Louisiana

ABSTRACT

PRECONDITIONED CONJUGATE GRADIENT ALGORITHM

Elena Caraba

Department of Mathematics

Bachelor of Science

The preconditioned conjugate gradient (PCG) method is one of the most effective tools for solving large sparse symmetric positive-definite systems. In this project we derive error estimates for the PCG method and we apply the PCG method to a discrete problem resulting from a weakly over-penalized interior penalty method for the Laplace equation on a planar domain.

ACKNOWLEDGMENTS

First and foremost, I thank Professor Susanne Brenner and Professor Li-Yeng Sung for suggesting a good research problem to explore and for all the good advice they gave me during the preparation of this thesis.

I also thank Professor Gabrielle Allen, Professor Paul Saylor and Professor Ed Seidel for always believing in me, guiding me throughout my years as an undergraduate student and offering me the opportunity to do research in the Center for Computation & Technology.

I thank once again Professor Paul Saylor, as well as Doctor Erik Schnetter, Professor Eldad Haber, Julianne Chung, Irina Craciun and Zachary Smith for useful discussions and advices.

I thank the Honors College, the Center for Computation & Technology and Lawrence Smolinsky for their generous support with the preparation of the thesis and with the participation to the National Conference for Undergraduate Research from Salisbury, Maryland in April 2008.

Last but not least, my gratitude goes to my family for being my first supporters and always giving me the best they could.

Contents

List of Figures

Chapter 1

Introduction

1.1 Mathematical Background

Often, in real life applications, the systems of equations that solve particular science problems, cannot be solved quickly with direct methods; instead, sequences of approximating solutions are needed to reach the exact solution. And most of the time these solutions are found using iterative methods. [1]

Before 1952, two great Mathematicians, Magnus R. Hestenes and Eduard Stiefel, were working independently on developing an effective method for solving a system of n simultaneously equations in n unknowns, especially when n is large. For effectiveness, they were looking for a method that is simple, requiring minimum storage space; rapid convergent for a number of infinte steps; stable with respect to round-off errors; and at each step of the method, one should have some information about the solution and the current estimate should be better than the previous one. Unfortunately, all these criteria are hard to meet and in real life, there is no best method that can solve all types of problems.

In 1952, Hestenes and Stiefel, joining their research results [8], discovered one of the most powerful iteration methods for rapidly solving large linear systems of

equations with symmetric positive definite coefficient matrices: the conjugate gradient method or simply, the CG algorithm.

When CG is applied to $Ax = b$, where matrix A has the properties mentioned above, the solution is reached in at most n steps, where n is the dimension of the matrix of coefficients; a maximum of original data is preserved; the method is simple to code and requires little storage space; each step gives an estimate better than the previous one; and the method can be started anew at any step of the iterative method.

The CG algorithm is a technique that does not require any computation of the eigenvalues of the system, as other methods do, and can therefore be used as a more direct method for finding the solution of $Ax = b$. [6, 7]

The purpose of this thesis is to present the conjugate gradient algorithm and the derivation of its formulas; to make an analysis of its rate of convergence ; to present a technique for speeding up the iterative process, known as preconditioning; and to present a numerical experiment based on a ill-conditioned problem.

1.2 Terminology

In this section we will define some of the main and most frequent notions used throughout this thesis.

- A *Krylov subspace of dimension n* is formed by a linear combination of the vectors $b, Ab, \ldots, A^{n-1}b$, where A is a matrix and b is a vector.
- A matrix A is *symmetric* if it is equal to its transpose: $A = A^T$.
- A matrix $A \in \mathbb{R}^{nxn}$ is *positive-definite* if $x^T Ax > 0$, for all non-zero vectors $x \in \mathbb{R}^n$.
- The *A-norm* of a vector x is defined as: $\|x\|_A = \sqrt{x^T Ax}$.

- If A is an nxn matrix, then its *condition number* with respect to a matrix norm $\|.\|$ is: $\kappa(A) = \|A\|\|A^{-1}\|$.

- A matrix P is called a *preconditioner* of another matrix A, if the condition number of the matrix $P^{-1}A$ is smaller than the condition number of A.

- A matrix is called *sparse* if it is primarily composed of zeros.

- We say two vectors p_{k-1} and p_k are A-*conjugate* or conjugate with respect to A if: $p_{k-1}^T A p_k = 0$

- The set $x_0 + < p_0, p_1, \ldots, p_{k-1} >= \{y : y - x_0 \in < p_0, p_1, \ldots, p_{k-1} >\}$ is called the k^{th} *dimensional affine space* through the point x_0.

- A matrix A is symmetric if and only if $v^T A w = w^T A v$, where w, v are two row vectors.

Chapter 2

The Conjugate Gradient Method

A Krylov subspace iterative method, the conjugate gradient method, or simply CG, is one of the most important algorithms in solving large and sparse symmetric positive definite (SPD) matrices.

Throughout this chapter we will discuss some of the benefits and disadvantages of the method. One of the main benefits is that it reaches the solution in at most n steps if the system we are trying to solve has n linear equations with n unknowns; moreover, the method is easy to code in Matlab and doesn't request a lot of memory. The rate of convergence of the method is in general good, but for ill-conditioned matrices becomes problematic; in the following chapter we will show how CG can be improved for this kind of matrices.

The CG algorithm, applied to the system $Ax = b$, starts with an initial guess of the solution x_0, with an initial residual r_0, and with an initial search direction that is equal to the initial residual: $p_0 = r_0$.

The idea behind the conjugate gradient method is that the residual $r_k = b - Ax_k$ is orthogonal to the Krylov subspace generated by b, and therefore each residual is perpendicular to all the previous residuals. The residual is computed at each step.

The solution at the next step is found using a search direction that is only a linear

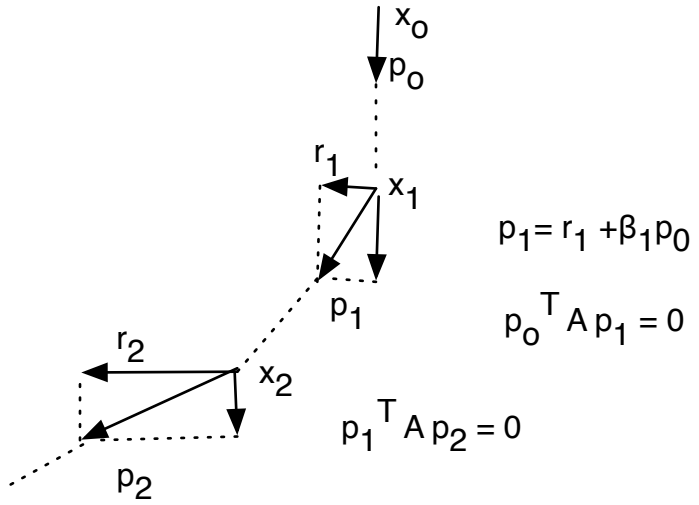

Figure 2.1: Searching Direction of the CG Algorithm.

combination of the previous search directions, which for x_1 is just a combination between the previous and the current residual.

Then, the solution at step k, x_k, is just the previous iterate plus a constant times the last search direction; the immediate benefit of the search directions is that there is no need to store the previous search directions. And using the orthogonality of the residuals to these previous search directions, the search is linearly independent of the previous directions. And for the solution at the next step, a new search direction is computed, as well as a new residual and new constants. The role of the constants are to give an optimal approximate solution. [1, 11]

A more visual explanation of how the CG algorithm finds the approximate solution to the exact solution is given in Fig. 2.1.

The iterative formulas of CG [13] are given below:

- approximate solution: $x_k = x_{k-1} + \alpha_k p_{k-1}$

- residual: $r_k = r_{k-1} - \alpha_k A p_{k-1}$

- search direction: $p_k = r_k + \beta_k p_{k-1}$

- improvement at step k: $\beta_k = \frac{r_k^T r_k}{r_{k-1}^T r_{k-1}}$

- step length: $\alpha_k = \frac{r_{k-1}^T r_{k-1}}{p_{k-1}^T A p_{k-1}}$

2.1 Deriving the Conjugate Gradient Algorithm

In this section we will derive the formulas of the conjugate gradient algorithm starting by minimizing the A-norm of the error at step k, $\|e_k\|_A$ [13]. The error at step k is defined as $e_k = x_* - x_k$, where $x_* = A^{-1}b$ is the exact solution and x_k is the solution at step k.

Using the definition of the A-norm of a vector given in Section 1.2, we define the A-norm of the error at step k to be: $\|e_k\|_A = \sqrt{e_k^T A e_k}$ and we introduce a new function $f(\alpha_k) = \|e_k\|_A^2$.

The error at step k defined as $e_k = x_* - x_k$ is also equivalent to $e_k = e_{k-1} - \alpha_k p_{k-1}$, if we consider that the error at step $k-1$ is $e_{k-1} = x_* - x_{k-1}$. Using this equivalence for the error, $f(\alpha_k)$ becomes:

$$\begin{aligned} f(\alpha_k) &= (e_{k-1} - \alpha_k p_{k-1})^T A (e_{k-1} - \alpha_k p_{k-1}) \\ &= (e_{k-1}^T - \alpha_k p_{k-1}^T) A (e_{k-1} - \alpha_k p_{k-1}) \\ &= e_{k-1}^T A e_{k-1} - \alpha_k e_{k-1}^T A p_{k-1} - \alpha_k p_{k-1}^T A e_{k-1} + \alpha_k^2 p_{k-1}^T A p_{k-1} \end{aligned}$$

In order to determine whether the function has a minimum or not at α_k, we need to calculate the first and second derivatives of the function, and since A is symmetric: $e_{k-1}^T A p_{k-1} = p_{k-1}^T A e_{k-1}$, we get:

$f'(\alpha_k) = 2\alpha_k p_{k-1}^T A e_{k-1} - 2p_{k-1}^T A e_{k-1}$

$f''(\alpha_k) = 2p_{k-1}^T A p_{k-1}$

A being SPD, implies $f''(\alpha_k) > 0$ and therefore, the function $f(\alpha_k)$ has a minimum at α_k.

$\Rightarrow f'(\alpha_k) = 0$

$\Rightarrow \alpha_k p_{k-1}^T A p_{k-1} = p_{k-1}^T A e_{k-1} \Rightarrow \boxed{\alpha_k = \frac{p_{k-1}^T A e_{k-1}}{p_{k-1}^T A p_{k-1}}}$

Later (in Section 2.4), we will show that $\alpha_k = \frac{r_{k-1}^T r_{k-1}}{p_{k-1}^T A p_{k-1}}$

Next, we deduce the residual at the k^{th} step: $r_k = b - Ax_k$

We know that $\boxed{x_k = x_{k-1} + \alpha_k p_{k-1}}$, so r_k becomes:

$r_k = b - A(x_{k-1} + \alpha_k p_{k-1})$, or

$r_k = b - Ax_{k-1} - \alpha_k A p_{k-1}$

$\Rightarrow \boxed{r_k = r_{k-1} - \alpha_k A p_{k-1}}$

The search direction p_k, is a linear combination of the previous search direction p_{k-1}, and the error at the current step, r_k:

$p_k = \gamma_k r_k + \beta_k p_{k-1}$, but $\gamma_k = 1$ because of normalization

Therefore, $\boxed{p_k = r_k + \beta_k p_{k-1}}$

We say p_k and p_{k-1} are conjugate with respect to A, if $p_{k-1}^T A p_k = 0$. If we replace here p_k with the formula we found for it, this relation is transformed into:

$p_{k-1}^T A(r_k + \beta_k p_{k-1}) = 0$

After we multiply out, it becomes $p_{k-1}^T A r_k + \beta_k p_{k-1}^T A p_{k-1} = 0$, and solving for β_k:

$\boxed{\beta_k = -\frac{p_{k-1}^T A r_k}{p_{k-1}^T A p_{k-1}}}$

Later, in Section 2.4, we will find a new formula: $\beta_k = \frac{r_k^T r_k}{r_{k-1}^T r_{k-1}}$.

2.2 An Alternative Approach to Solving $Ax = b$: A Minimization Problem

Solving $Ax = b$ is equivalent to finding x such that the quadratic function $F(x) = \frac{1}{2}x^TAx - b^Tx$ is as small as possible. And this is true if $F(x) \geq F(\hat{x})$, where $\hat{x}$ is the exact solution for $A\hat{x} = b$.

$$\begin{cases} F(x) = \frac{1}{2}x^TAx - b^Tx \\ F(\hat{x}) = \frac{1}{2}\hat{x}^TA\hat{x} - b^T\hat{x} \end{cases}, \quad \text{but} \quad b^T = \hat{x}^TA \tag{2.1}$$

$$\begin{cases} F(x) = \frac{1}{2}x^TAx - \hat{x}^TAx \\ F(\hat{x}) = \frac{1}{2}\hat{x}^TA\hat{x} - \hat{x}^TA\hat{x} \end{cases} \Rightarrow \begin{cases} F(x) = \frac{1}{2}x^TAx - \hat{x}^TAx \\ F(\hat{x}) = -\frac{1}{2}\hat{x}^TA\hat{x} \end{cases}$$

Subtracting the two equations of the system:

$\Rightarrow F(x) - F(\hat{x}) = \frac{1}{2}x^TAx - \frac{1}{2}\hat{x}^TAx - \frac{1}{2}\hat{x}^TAx + \frac{1}{2}\hat{x}^TA\hat{x}$

$\Rightarrow F(x) - F(\hat{x}) = \frac{1}{2}(x - \hat{x})^TA(x - \hat{x})$

$\Rightarrow F(x) - F(\hat{x}) \geq 0 \Leftrightarrow (x - \hat{x})^TA(x - \hat{x}) \geq 0$, which is true.

Therefore, solving $Ax = b$ is equivalent to finding an x such that $F(x)$ is as small as possible.

For all y from $x_0 + < p_0, p_1, \ldots, p_{k-1} >$, a k^{th} dimensional affine space the following holds:

$$F(x_k) = \min F(y) \tag{2.2}$$

Proof.

Let c_i and d_i be some scalars. Using the proof from Subsection 2.5.1 and the fact that y is in a k^{th} dimensional affine space generated by the vectors p_i we obtain:

$$\left.\begin{array}{l} x_k = x_0 + \sum_{i=1}^{k-1} c_ip_i \\ y = x_0 + \sum_{i=1}^{k-1} d_ip_i \end{array}\right\} \Rightarrow y - x_k = \sum_{i=1}^{k-1} \underbrace{(d_i - c_i)}_{\text{some scalar}} p_i = p \in < p_0, \ldots, p_{k-1} >, \text{for some p}$$

Therefore, $y = x_k + p$; and making this substitution for y in the quadratic function $F(y)$ it follows that:

$$\begin{aligned} F(y) &= \frac{1}{2}(x_k + p)^T A(x_k + p) - b^T(x_k + p) \\ &= (\frac{1}{2}x_k^T A x_k - b^T x_k) + p^T A x_k - b^T p + \frac{1}{2}p^T A p \\ &= F(x_k) - p^T(\underbrace{Ax_k - b}_{r_k}) + \frac{1}{2}\underbrace{p^T A p}_{>0} \\ &= F(x_k) - p^T r_k + \frac{1}{2}p^T A p \end{aligned}$$

Since $p^T r_k = 0$ because $p \in < p_0, \ldots, p_{k-1} > = < r_0, \ldots, r_{k-1} >$

$\Rightarrow F(y) = F(x_k) + \frac{1}{2}p^T A p$

$\Rightarrow F(x_k) \leq F(y)$

$\Rightarrow F(x_k) = \min F(y)$, $\forall y \in < p_0, \ldots, p_{k-1} > + x_0$

2.3 Applying the CG Algorithm to a 2 x 2 SPD matrix

In this section we will solve the system $Ax = b$ using the conjugate gradient method, for A a 2x2 symmetric positive definite matrix, and will show that the exact solution $\hat{x}$ is reached in two steps.

Consider the quadratic function defined on the previous section:

$$F(x) = \frac{1}{2}x^T A x - b^T x$$

Then, for the symmetric matrix A, the gradient of this function is nothing else but $Ax - b$:

$$F'(x) = Ax - b$$

Therefore, solving $Ax = b$ is equivalent to finding the x that minimizes $F(x)$.The solution of $Ax = b$ is a critical point of $F(x)$. From Multivariable Calculus one knows that the gradient, a vector field, always points in the direction of the greatest

increase of a function. And therefore, for every x, the gradient points in the direction of the steepest increase of $F(x)$. Thus, the direction of the steepest decrease is in the direction opposite $F'(x)$. [11]

Let x_1 and x_2 be the solutions returned by CG at the first and second iteration, respectively; then the residual after the first iteration is $r_1 = A\hat{x} - Ax_1$.

As we have previously mentioned, the idea behind the conjugate gradients method is to generate a series of approximating solutions until we reach the one closest to the exact solution. The gradient vector will always point in the direction of the steepest increase of the quadratic function, and for reaching the exact solution, the previous residual (different than zero and called r_1 in the 2x2 case) will be orthogonal to the search direction. More, the quadratic function $F(x)$ we showed in Section 2.2 to be minimized by the solution of CG, will reach the minimum where the projection of the gradient vector on the search line is zero. [1, 11]

In Fig. 2.2 we illlustrate these steps by showing a contour plot with the gradient vectors pointing in the direction of the steepest increase; the first step x_1 is taken along the initial search line p_0 will lead to x_1; this x_1 will hit the exact solution at the next step. Note: the error $e_1 = \hat{x} - x_1$ will be A-orthogonal to p_0.

We apply the CG method to the 2x2 SPD matrix by first checking that the residual after the first iteration ($r_1 = A\hat{x} - Ax_1$) is orthogonal to the initial residual (r_0): $r_1^T r_0 = 0$. Since $r_0 = p_0$ and $r_1 = b - Ax_1 = A\hat{x} - Ax_1$, saying residuals are orthogonal is the same as $(\hat{x} - x_1)^T A p_0 = 0$, which is nothing else than $e_1^T A p_0 = 0$.

The gradient at the bottom-most point of the contour plot being orthogonal to the gradient of the previous step, implies that the best approximation to the exact solution has been reached.

$\Rightarrow$ For the 2x2 matrix, the solution is reached in two steps.

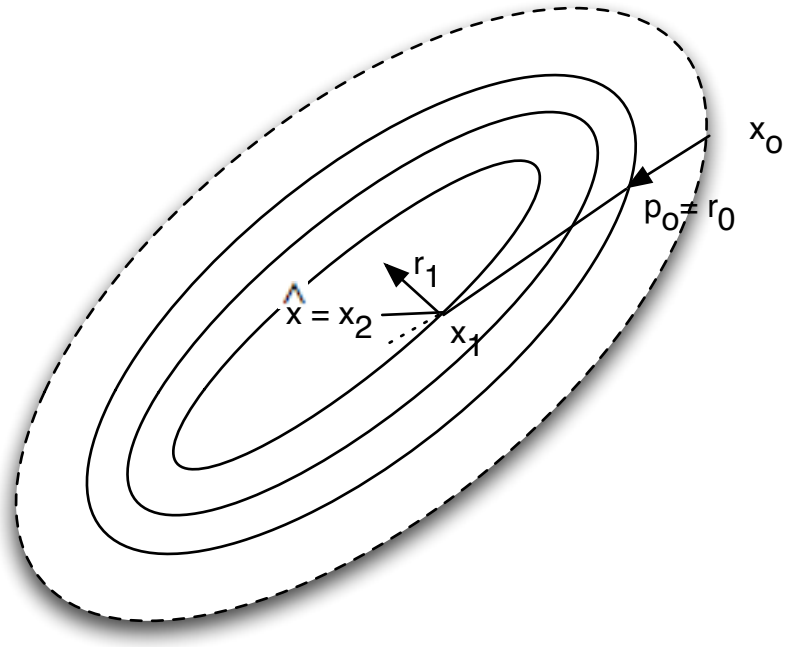

Figure 2.2: CG Algorithm for a 2x2 matrix.

2.4 CG Reaches the Exact Solution in n Steps

One of the main properties of the CG algorithm that distinguishes it from other iterative methods is the fact that it reaches the exact solution of $Ax = b$ in at most n steps for any initial guess x_0 and A is an nxn symmetric positive definite matrix. In this section we will prove this property and we will so with the help of mathematical induction.

We assume that CG algorithm can be carried out to k steps ($n > k$).

Prove:

- $< r_0, r_1, \ldots, r_k >=< p_0, p_1, \ldots, p_k >$
  - where $< r_0, r_1, \ldots, r_k >= \{x \in \mathbb{R}^n : x = \sum_{j=1}^{k} c_j r_j, c \text{ is a constant}\}$
  - and $< p_0, p_1, \ldots, p_k >= \{x \in \mathbb{R}^n : x = \sum_{j=1}^{k} c_j p_j, c \text{ is a constant}\}$

Assume that $p_0 \neq 0, p_1 \neq 0, \ldots, p_k \neq 0$, prove:

- $r_l^T r_j = 0, 0 \leq j < l \leq k$ (residual vectors are orthogonal to one another)
- $p_l^T A p_j = 0, 0 \leq j < l \leq k$ (direction vectors are conjugate to one another)

Proof

• First we prove by mathematical induction $< r_0, r_1, \ldots, r_k >=< p_0, p_1, \ldots, p_k >$

Step 1.

$r_0 = p_0$. True.

Step 2.

To prove

$$< r_0, r_1, \ldots, r_k >=< p_0, p_1, \ldots, p_k > \tag{2.3}$$

assume it is true up to k, and then show that

$$< r_0, r_1, \ldots, r_k, r_{k+1} >=< p_0, p_1, \ldots, p_k, p_{k+1} > \tag{2.4}$$

From the formula found for the search direction in the derivation of the CG algorithm, the residual at the $(k+1)^{th}$ step is also:

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

which implies that the residual at this step is a linear combination of p_{k+1} and p_k.

If relation (2.3) holds true, then just adding one more term, r_{k+1}, to both subspaces, makes the following true as well:

$$< r_0, r_1, \ldots, r_k, r_{k+1} >=< p_0, p_1, \ldots, p_k, r_{k+1} > \tag{2.5}$$

But since we claimed that r_{k+1} is a linear combination of p_{k+1} and p_k, the relation (2.5) becomes equivalent to:

$$< r_0, r_1, \ldots, r_k, r_{k+1} >=< p_0, p_1, \ldots, p_k, p_{k+1} >,$$

which proves the current step of induction.

Therefore, $< r_0, r_1, \ldots, r_k >=< p_0, p_1, \ldots, p_k >$ and we can also conclude that

$$p_k = \gamma_k r_k + \gamma_{k-1} r_{k-1} + \ldots + \gamma_1 r_1 + \gamma_0 r_0$$

We know $p_k \neq 0$ unless $\gamma_i = 0$, with $i \in \{0, \ldots k\}$. So the vectors $r_0, \ldots, r_k$ are linearly independent. They span any basis of the subspace given by their set of vectors in that subspace. And the dimension of the subspace is the number of vectors in any of these bases. In conclusion,

$$dim < r_0, \ldots, r_k >= k+1$$

• Next, we have to prove that $r_l^T r_j = 0$ and $p_l^T A p_j = 0$, with $0 \leq j < l \leq k$. We proceed by assuming that the relationships are true for l and then we prove they also hold for $l+1$.

Then we need to show: $r_{l+1}^T r_j = 0$, $l+1 > j \geq 0$ and $p_{l+1}^T A p_j = 0$, $l \geq j \geq 0$.

We will use mathematical induction on l.

1) If $l = j$, show $r_{l+1}^T r_l = 0$

We start with the formula we found for the residual:

$$r_{l+1} = r_l - \alpha_{l+1} A p_l \Rightarrow r_{l+1}^T r_l = r_l^T r_l - \alpha_{l+1} p_l^T A r_l \tag{2.6}$$

If eq. (2.6) is zero, then $r_l^T r_l - \alpha_{l+1} p_l^T A r_l = 0$ and we find a new formula for the step length:

$$\alpha_{l+1} = \frac{r_l^T r_l}{p_l^T A r_l}$$

According to the previous formula we found for $\alpha_{k+1} = \dfrac{p_k^T r_k}{p_k^T A p_k}$, we need to ask if $p_l^T r_l = r_l^T r_l$.

We proceed with the formula of the search direction, take its transpose and multiply it by r_l:

$p_l = r_l + \beta_l p_{l-1}$

$p_l^T r_l = r_l^T r_l + \beta_l p_{l-1}^T r_l$, by (2.3) $p_{l-1}^T r_l = 0$

$\Rightarrow p_l^T r_l = r_l^T r_l$

We also need to ask if $p_l^T A r_l = p_l^T A p_l$. For this we proceed by replacing p_l with the formula we derived for the search line: $p_l = r_l + \beta_l p_{l-1}$.

$p_l^T A p_l = p_l^T A(r_l + \beta_l p_{l-1}) = p_l^T A r_l + \beta_l p_l^T A p_{l-1}$

From the hypotheses of induction, $p_l^T A p_{l-1} = 0$, because $l = j$.

Therefore, we have that $p_l^T A p_l = p_l^T A r_l$ and indeed $\alpha_{l+1} = \frac{r_l^T r_l}{p_l^T A r_l}$. This also implies that equation (2.6) is zero and $r_{l+1}^T r_l = 0$

Therefore, for $l = j$

$$r_{l+1}^T r_j = 0 \tag{2.7}$$

Remark: A new formula for the step length has been deduced:

$$\boxed{\alpha_k = \frac{r_{k-1}^T r_{-1}}{p_{k-1}^T A p_{k-1}}}$$

2) If $j < l$, need to show that $r_{l+1}^T r_j = 0$

We can show it by using the formula of the residual, taking its transpose and multiplying it by r_j:

$r_{l+1} = r_l - \alpha_{l+1} A p_l$

$r_{l+1}^T r_j = r_l^T r_j - \alpha_{l+1} p_l^T A r_j$

But by the hypotheses of the induction $r_l^T r_j = 0$.

$$\left.\begin{array}{l} r_j \in < p_0, \ldots, p_j > \Rightarrow A r_j \in < A p_0, \ldots, A p_j > \\ p_l \text{ is orthogonal to } < A p_0, \ldots, A p_j > \end{array}\right\} \Rightarrow \; p_l^T A r_j = 0$$

$$\Rightarrow r_{l+1}^T r_j = 0 \tag{2.8}$$

By equation (2.7) and equation (2.8) with induction on l

$\Rightarrow r_l^T r_j = 0$, indeed.

- We also have to prove that $p_{l+1}^T A p_j = 0$, for $l \geq j \geq 0$.

1) If $j = l$, we have to show that $p_{l+1}^T A p_l = 0$.

Again, we proceed with the formula of the search direction, taking its transpose and multiplying it by Ap_l from the right:

$p_{l+1} = r_{l+1} + \beta_{l+1} p_l$

$p_{l+1}^T A p_l = r_{l+1}^T A p_l + \beta_{l+1} p_l^T A p_l$

We know that the improvement at step $l+1$ is $\beta_{l+1} = -\frac{p_l A r_{l+1}^T}{p_l^T A p_l}$ and by the symmetry of A, $p_l A r_{l+1}^T = r_{l+1}^T A p_l$. Then:

$$\begin{aligned} p_{l+1}^T A p_l &= r_{l+1}^T A p_l - \frac{r_{l+1}^T A p_l}{p_l^T A p_l} p_l^T A p_l \\ &= r_{l+1}^T A p_l - r_{l+1}^T A p_l \\ &= 0 \end{aligned}$$

Hence,

$$p_{l+1}^T A p_l = 0 \tag{2.9}$$

2) If $j < l$, we need to show $p_{l+1}^T A p_j = 0$.

In the relationship we have to show we replace p_{l+1} with its formula we already know.

$p_{l+1}^T A p_j = (r_{l+1}^T + \beta_{l+1} p_l^T) A p_j = r_{l+1}^T A p_j + \beta_{l+1} \underbrace{p_l^T A p_j}_{0}$

$$p_{l+1}^T A p_j = r_{l+1}^T A p_j \tag{2.10}$$

Next, from the formula of the residual $r_{j+1} = r_j - \alpha_{j+1} A p_j$ and from the fact we previously proved about residuals that $r_l^T r_j = 0$ we get:

$0 = r_{l+1}^T r_{j+1} = \underbrace{r_{l+1}^T r_j}_{0} - \alpha_{j+1} r_{l+1}^T A p_j$

Or,

$$\left.\begin{array}{l} 0 = -\alpha_{j+1} r_{l+1}^T A p_j \\ \alpha_{j+1} = \frac{r_j^T r_j}{p_j^T A p_j} \neq 0 \end{array}\right\} \Rightarrow \; r_{l+1}^T A p_j = 0 \;\; (\dagger)$$

We replace (†) in equation (2.10) and we get:

$$p_{l+1}^T A p_j = 0 \tag{2.11}$$

From equation (2.9) and equation (2.11) and again by induction on l, we can conclude that

$$p_l^T A p_j = 0$$

Now, we can simplify the expression of the improvement at step k, β_k, found in Section 2.1. [11]

Taking the inner product of the residual r_i with $r_{j+1} = r_j - \alpha_{j+1} A p_j$,

$$\alpha_{j+1} r_i^T A p_j = r_i^T r_j - r_i^T r_{j+1}^T.$$

For $i = j + 1$, $r_i^T r_j^T = 0$ and

$$r_i^T A p_j = -\frac{1}{\alpha_i} r_i^T r_i$$

By the symmetry of A, $p_{k-1}^T A r_k = r_k^T A p_{k-1}$, and the formula from Section 2.1, becomes

$$\beta_k = -\frac{p_{k-1}^T A r_k}{p_{k-1}^T A p_{k-1}} = -\frac{r_k^T A p_{k-1}}{p_{k-1}^T A p_{k-1}} = -\frac{1}{p_{k-1}^T A p_{k-1}} \left(-\frac{1}{\alpha_k} r_k^T r_k \right)$$

And replacing $\alpha_k = \dfrac{p_{k-1}^T r_{k-1}}{p_{k-1}^T A p_{k-1}}$ in the above formula, $\boxed{\beta_k = \dfrac{r_k^T r_k}{r_{k-1}^T r_{k-1}}}$

2.5 Krylov Subspaces

2.5.1 The Krylov Subspace of the Approximate Solution

We will show now that if $r_0 \neq 0, \ldots, r_{k-1} \neq 0$, then $x_k - x_0 \in < p_0, p_1, \ldots, p_{k-1} >$

Proof (by induction)

Step 1.

$x_1 - x_0 = \alpha p_0$

$\Rightarrow x_1 = \alpha p_o + x_0$

Step 2.

Assume that $x_k - x_0 \in < p_0, p_1, \ldots, p_{k-1} >$ is true. Then it is also true that:

$x_k - x_0 = \alpha_1 p_0 + \alpha_2 p_1 + \ldots + \alpha_k p_{k-1}$

We will need to show that $x_{k+1} - x_0 = \alpha_1 p_0 + \ldots + \alpha_k p_{k-1} + \alpha_{k+1} p_k$, and we will do this by manipulating the formula of the approximate solution from the CG algorithm:

$x_{k+1} = x_k + \alpha_{k+1} p_k$

$x_{k+1} - x_0 = x_k + \alpha_{k+1} p_k - x_0$

$x_{k+1} - x_0 = (x_k - x_0) + \alpha_{k+1} p_k$

$x_{k+1} - x_0 = \alpha_1 p_0 + \alpha_2 p_1 \ldots + \alpha_k p_{k-1} + \alpha_{k+1} p_k$

$\Rightarrow x_{k+1} - x_0$ is a linear combination of $p_0, p_1, \ldots, p_k$

$\Rightarrow x_{k+1} - x_0 \in < p_0, p_1, \ldots, p_{k-1}, p_k >$

From Step 1 and Step 2 of induction we can claim $x_k - x_0 \in < p_0, p_1, \ldots, p_{k-1} >$.

2.5.2 The Krylov Subspace - A Subspace of Residuals

In this section we will prove that the Krylov subspace $< r_0, Ar_0, \ldots, A^{k-1}r_0 >$ is the same with the subspace generated by the residuals $< r_0, r_1, \ldots, r_{k-1} >$.

Proof by mathematical induction

Step 1)

$r_0 = r_0$ True!

Step 2)

Assume that the following is true

$$< r_0, Ar_0, \ldots, A^{k-1}r_0 >=< r_0, r_1, \ldots, r_{k-1} > \tag{2.12}$$

and then we need to show that the following also holds:

$< r_0, Ar_0, \ldots, A^{k-1}r_0, A^k r_0 >=< r_0, r_1, \ldots, r_{k-1}, r_k >$

Adding one more term, namely r_k, to the relation (2.12), we have:

$< r_0, r_1, \ldots, r_{k-1}, r_k >=< r_0, Ar_0, \ldots, A^{k-1}r_0, r_k >$

But we know that:

$r_k = r_{k-1} - \alpha_k A p_{k-1}$

Also, from Section 2.4 we know that:

$< r_0, r_1, \ldots, r_{k-1}, r_k >=< p_0, p_1, \ldots, p_{k-1}, p_k >$

which implies that the following also hold:

$p_k = \delta_0 r_0 + \delta_1 r_1 + \ldots + \delta_k r_k$

$p_{k-1} = \delta_0 r_0 + \delta_1 r_1 + \ldots + \delta_{k-1} r_{k-1}$

Replacing the latter equations in the formula of the residual we obtain:

$$r_k = r_{k-1} - \alpha_k A(\delta_0 r_0 + \delta_1 r_1 + \ldots + \delta_{k-1} r_{k-1}) \tag{2.13}$$

The equation (2.12) can also be written as:

$$\begin{aligned} r_0 &= r_0 \\ r_1 &= c_0 r_0 + c_1 A r_0 \\ r_2 &= c_0 r_0 + c_1 A r_0 + c_2 A^2 r_0 \\ &\vdots \\ r_{k-1} &= c_0 r_0 + c_1 A r_0 + c_2 A^2 r_0 + \ldots + c_{k-1} A^{k-1} r_0 \end{aligned}$$

Plugging the above formulas for $r_0, r_1, \ldots, r_{k-1}$ in the equation (2.13), and grouping the residuals together after changing the coefficients accordingly, we get a new r_k:

$r_k = c_0 r_0 + b_1 A r_0 + b_2 A^2 r_0 + \ldots + b_k A^k r_0,$

where $b_1, b_2, \ldots, b_k$ are the new coefficients.

Therefore, we can claim that r_k is a linear combination of $r_0, Ar_0, A^2 r_0, \ldots, A^k r_0$. This implies:

$< r_0, r_1, \ldots, r_{k-1} > \subseteq < r_0, Ar_0, \ldots, A^{k-1} r_0 >$

Part 2) In this part we will try to prove the reverse inclusion:

$< r_0, Ar_0, \ldots, A^{k-1} r_0 > \subseteq < r_0, r_1, \ldots, r_{k-1} >$

From relation (2.12), we can express the residuals in the following way:

$$\begin{aligned} r_0 &= r_0 \\ Ar_0 &= c_0 r_0 + c_1 r_1 \\ A^2 r_0 &= c_0 r_0 + c_1 r_1 + c_2 r_2 \\ &\vdots \\ A^{k-1} r_0 &= c_0 r_0 + c_1 r_1 + \ldots + c_{k-1} r_{k-1} \end{aligned}$$

Another fact we know, just by using the relation (2.12) is:

$< r_0, Ar_0, \ldots, A^{k-1} r_0, A^k r_0 > \supseteq < r_0, r_1, \ldots, r_{k-1}, A^k r_0 >$

Now we look to see what is $A^k r_0$:

$$\begin{aligned} A^k r_0 &= A(A^{k-1} r_0) \\ &= A(c_0 r_0 + c_1 r_1 + \ldots + c_{k-1} r_{k-1}) \\ &= c_0 A r_0 + c_1 A r_1 + \ldots + c_{k-1} A r_{k-1} \end{aligned}$$

Using the formula for the residual, we can find the following:

$$\begin{aligned} A r_1 &= A(r_0 - \lambda_1 A p_0) \\ &= A r_0 - \lambda_1 A^2 r_0 \\ &= (c_0 r_0 + c_1 r_1) - \lambda_1 (c_0 r_0 + c_1 r_1 + c_2 r_2) \end{aligned}$$

Grouping the residual at the same step together and introducing new coefficients βs, which are just a combination of the previous ones, we obtain:

$A r_1 = \beta_0 r_0 + \beta_1 r_1 + \beta_2 r_2$

In a similar way we find:

$A r_2 = \beta_0 r_0 + \beta_1 r_1 + \beta_2 r_2 + \beta_3 r_3$

And we proceed in this way until we find that

$A r_{k-1} = \beta_0 r_0 + \beta_1 r_1 + \ldots + \beta_{k-1} r_{k-1} + \beta_k r_k$

Now we can return to the equation of $A^k r_0$ and introduce the above formulas in:

$A^k r_0 = c_0(c_0 r_0 + c_1 r_1) + c_1(\beta_0 r_0 + \beta_1 r_1 + \beta_2 r_2) + \ldots + c_{k-1}(\beta_0 r_0 + \beta_1 r_1 + \ldots + \beta_{k-1} r_{k-1} + \beta_k r_k)$

Grouping the residuals at the same step together and introducing new coefficients γ the new $A^k r_0$ becomes:

$A^k r_0 = \gamma_0 r_0 + \gamma_1 r_1 + \ldots + \gamma_{k-1} r_{k-1} + \gamma_k r_k$

Therefore, we can claim that $< r_0, A r_0, \ldots, A^{k-1} r_0 > \subseteq < r_0, r_1, \ldots, r_{k-1} >$.

2.6 Minimizing the Error

2.6.1 Minimizing the Error at the k^{th} Step

In this section, we want to show that $\|e_k\|_A = \min\|v\|_A$, where $v \in < e_0, Ae_0, \ldots, A^k e_0 >$. We know $e_k = \hat{x} - x_k$, where $\hat{x}$ is the exact solution and x_k is the approximate solution at the k^{th} step. Then our problem reduces to showing that

$$\hat{x} - x_k \in e_0 + < Ae_0, \ldots, A^k e_0 >, k \geq 1$$

Using the information we have about x_k , we get:

$$\begin{aligned} e_k &= \hat{x} - x_k \\ &= \hat{x} - (x_{k-1} + \alpha_k p_{k-1}) \\ &= (\hat{x} - x_{k-1}) - \alpha_k p_{k-1} \\ &= e_{k-1} - \alpha_k p_{k-1} \end{aligned}$$

Using the information we have at the $(k-1)^{th}$ step from the $(k-2)^{th}$ the formula above for the error at the k^{th} step becomes:

$$e_k = e_{k-2} - \alpha_{k-1} p_{k-2} - \alpha_k p_{k-1}$$

Generalizing, we obtain:

$$e_k = e_0 - \alpha_1 p_0 - \alpha_2 p_1 - \ldots - \alpha_k p_{k-1}$$

And this is equivalent to $e_k \in e_0 + < p_0, p_1, \ldots, p_{k-1} >$, which is the same as $e_k \in e_0 + < r_0, r_1, \ldots, r_{k-1} >$ because of relationship (2.3), and the same as $e_k \in e_0 + < r_0, Ar_0, \ldots, A^{k-1} r_0 >$ because of relationship (2.12).
Therefore, we can claim that $e_k \in e_0 + < Ae_0, \ldots, A^k e_0 >$.
Further, we will use the relationship (2.2) proved in Section 2.2: $F(x_k) = \min F(x)$,

for $x \in x_0 + < p_0, p_1, \ldots, p_k >$.

Let $v \in e_0 + < p_0, p_1, \ldots, p_{k-1} >$ or $v \in \hat{x} - x_0 + < p_0, p_1, \ldots, p_{k-1} >$.

This implies $\hat{x} - v \in x_0 + < p_0, p_1, \ldots, p_{k-1} >$. Then, using the fact that: $F(y) = F(\hat{x}) + \frac{1}{2}\|\hat{x} - y\|_A^2$, we can claim:

$F(x_k) \leq F(\hat{x} - v) = F(\hat{x}) + \frac{1}{2}\|v\|_A^2$, or

$F(x_k) - F(\hat{x}) \leq \frac{1}{2}\|v\|_A^2$, but $\|e_k\|_A = F(x_k) - F(\hat{x})$.

Therefore, $\|e_k\|_A = \min\|v\|_A$.

2.6.2 An Useful Formula for the A-norm of a Vector

In this section we will express $\|v\|_A$ in terms of the eigenvalues $\lambda_1, \ldots, \lambda_k$ of the SPD matrix A and some coefficients $c_1, \ldots, c_k$.

Let $q_1, \ldots, q_k$ be orthonormal eigenvectors of A. Then the following are true:

$$Aq_i = \lambda_i q_i, \text{ where } \lambda_i \text{ are eigenvalues}$$

$$q_i^T q_j = \delta_{ij} = \begin{cases} 0 & \text{if } i \neq j \\ 1 & \text{if } i = j \end{cases}$$

Any given vector v can be written as a linear combination of the eigenvectors q_i, $v = \sum_{i=1}^{n} c_i q_i$, and below we will derive an useful formula for the A-norm of this vector.

$$\begin{aligned} \|v\|_A^2 &= v^T A v \\ &= (\sum_{i=1}^{n} c_i q_i^T) A (\sum_{i=1}^{n} c_i q_i) \end{aligned}$$

After multiplying out the parentheses and using the definition of eigenvectors and the fact that they are orthonormal, the square of the A-norm of v reduces to:

$\|v\|_A^2 = \sum_{i=1}^{n} c_i^2 \lambda_i$

And therefore, $\|v\|_A = \sqrt{\sum_{i=1}^{n} c_i^2 \lambda_i}$.

2.6.3 Convergence Theorem - Part 1

In this section we will show that the A-norm of the error at the k^{th} step is the minimum of the A-norm of $p(A)e_0$, where e_0 is the initial error and p(A) is the polynomial of the form $p(A) = I + c_1A + \ldots + c_kA^k$ chosen from the set $\tilde{P}_k$ of all polynomials of degree k such that $p(0) = 1$.

If we let $p(A)e_0 = v$, where $p(A)e_0 = Ie_0 + c_1Ae_0 + \ldots + c_kA^ke_0$, it is implied that $v \in < e_0, Ae_0, \ldots, A^ke_0 >$.

Therefore, $\|p(A)e_0\|_A = \|v\|_A$, and from Section (2.6.1) we know:

$\|e_k\|_A = \min\|v\|_A$, or

$$\|e_k\|_A = \min_{p\in\tilde{P}_k} \|p(A)e_0\|_A.$$

2.6.4 Convergence Theorem - Part 2

Continuing from the previous subsection, we will show now that $\|e_k\|_A = \min_{p\in\tilde{P}_k}(\sum_{i=1}^n p(\lambda_i)v_i)^{1/2}$, where $0 < \lambda_1, \ldots, 0 < \lambda_k$ are the eigenvalues of A, and $e_0 = \sum_{i=1}^n c_iv_i$, where v_i are orthonormal eigenvectors of A.

For $p(A) = I + d_1A + \ldots + d_kA^k$,

$$\begin{aligned} p(A)v_1 &= Iv_1 + d_1Av_1 + \ldots + d_kA^kv_1 \\ &= v_1 + d_1\lambda_1v_1 + \ldots + d_k\lambda_1^kv_1 \\ &= (\sum_{i=0}^{n} d_i\lambda_1^i)v_1 \\ &= p(\lambda_1)v_1 \end{aligned}$$

Then, $p(A)e_0 = \sum_{i=1}^n p(A)c_iv_i = \sum_{i=1}^n c_ip(\lambda_i)v_i$

Then, from Subsection 2.6.2: $\|p(A)e_0\|_A^2 = \sum_{i=1}^n c_i^2p_i^2(\lambda_i)v_i = 1$ and $\|e_0\|^2 = \sum_{i=1}^n c_i^2v_i$.

$$1 \leq \max_{1\leq i\leq n} |p(\lambda_i)|^2 \sum_{i=1}^{n} c_i^2\lambda_i = \max_{1\leq i\leq n} |p(\lambda_i)|^2\|e_0\|_A$$

$$\|p(A)e_0\|_A \leq (\max_{1\leq i\leq n} |p(\lambda_i)|)\|e_0\|_A$$

$$\|e_k\|_A = \min_{p\in\tilde{P}_k} \|p(A)e_0\|_A \leq \min_{p\in\tilde{P}_k} \max_{1\leq i\leq n} |p(\lambda_i)|\|e_0\|_A$$
$$\|e_k\|_A \leq \max_{1\leq i\leq n} |p(\lambda_i)|\|e_0\|_A, \text{for any } p \in \tilde{P}_k$$

2.7 Chebyshev Polynomials

Further in our analysis of the conjugate gradient method, we have to answer the following question: Given $0 < a < b$, how can one find a polynomial $p \in P_k, p(0) = 1$ such that $\max_{\lambda\in[a,b]} |p(\lambda)|$ is as small as possible?

Let A be SPD and have the eigenvalues $0 < \lambda_1 \leq \lambda_2 \leq \ldots \leq \lambda_k$. Also, we know the following relationships hold true for $p(0) = 1$ and $p \in P_k$, where e_k is the error at the k^{th} step and e_0 is the initial error:

$$\max_{1\leq i\leq n} |p(\lambda_i)| \leq \max_{\lambda\in[\lambda_0,\lambda_k]} |p(\lambda)|$$
$$\|e_k\|_A \leq \max_{\lambda\in[\lambda_0,\lambda_k]} |p(\lambda)|\|e_0\|_A$$

The next two subsections will show that the Chebyshev polynomials are the ones that will maximize $|p(\lambda)|$ and which satisfy the above relations.

2.7.1 General Form of a Chebyshev Polynomial

The Chebyshev polynomials we will use in our analysis have the following form:

$$\begin{cases} T_0(x) = 1 \\ T_1(x) = x \\ T_{n+1}(x) = 2xT_n(x) - T_{n-1}(x) \end{cases}$$

We claim that the following is true

$$T_n = 2^{n-1}x^n + \ldots + 1 \tag{2.14}$$

with $n \geq 1$ and T_n is a Chebyshev polynomial and we will prove it by mathematical induction.

Proof by Induction

Step 1

Need to verify if (2.14) holds true for n=1.

$T_1(x) = 2^{1-1}x = x$. True.

Step 2

Suppose the general form for Chebyshev polynomials holds for $T_0, T_1, \ldots, T_{k-1}, T_k$; then we need to show that it also holds for T_{k+1}.

We know that $T_k = 2^{k-1}x^k + \ldots$ and we want to show that $T_{k+1} = 2^k x^{k+1} + \ldots$

Using the definition of the Chebyshev polynomials, we can find the polynomial of degree $k+1$ by using what we know about the polynomials of degree k and $k-1$ are:

$T_{k+1}(x) = 2x(x^k + \ldots) - (2^{k-2}x^{k-1} + \ldots)$

After we multiply out, the new polynomial becomes:

$T_{k+1}(x) = 2^k x^{k+1} + \ldots$, which has the form we were looking for.

And from Step 1 and 2 we can conclude that $T_n = 2^{n-1}x^n + \ldots$, $n \geq 1$

2.7.2 The Chebyshev Monic Polynomial

In this part, we will show that $\tilde{T}_n(x) = \frac{1}{2^{n-1}}T_n(x) = x^n + \ldots$ (which is a monic Chebyshev polynomial of degree n) satisfies the following property:

$$\max_{-1\leq x\leq 1} |\tilde{T}_n(x)| \leq \max_{-1\leq x\leq 1} |p_n(x)| \tag{2.15}$$

for any monic polynomial of degree n of the form $p_n(x) = x^n + \ldots$ [9]

Proof

Suppose equation (2.15) is false.

And let $x_i = cos(\frac{i\pi}{n})$, for $0 \leq i \leq n$.

For x in the interval $[-1, 1]$, the Chebyshev polynomials have the following form:

$T_n(x) = \cos(n\cos^{-1}(x)), n \geq 0$ and the following is true $T_n(\cos(\frac{i\pi}{n})) = (-1)^i$.

So, $T_n(x_i) = (-1)^i$ and $\tilde{T}_n(x_i) = 2^{1-n}(-1)^i$. Multiplying both sides by $(-1)^i$ we obtain:

$$(-1)^i\tilde{T}_n(x_i) = (-1)^i((-1)^i 2^{1-n})$$

which reduces to:

$$(-1)^i \tilde{T}_n(x_i) = 2^{1-n} \tag{2.16}$$

If $T_n(x_i) = (-1)^i$, then $|T_n(x)| \leq 1$.

Also,

$$|\tilde{T}_n(x)| = \left|\frac{1}{2^{n-1}} T_n(x)\right| \leq \left|\frac{1}{2^{n-1}}\right| |T_n(x)| \leq \left|\frac{1}{2^{n-1}}\right| = \frac{1}{2^{n-1}}$$

Thus, $\max\limits_{-1\leq x\leq 1} |\tilde{T}_n(x)| = \frac{1}{2^{n-1}}$

And supposing equation (2.15) is false, implies the following holds true:

$$|p_n(x)| < 2^{1-n} \tag{2.17}$$

Hence, we can conclude$(-1)^i[\tilde{T}_n(x_i) - p(x_i)] > 0$.

This means there are $(n+1)$ oscillations in sign over $[-1, 1]$, implying $\tilde{T}_n(x_i) - p(x_i)$ has n roots.

Contradiction!!!

$\tilde{T}_n(x_i) - p_n(x_i)$ has degree at most n-1 because both polynomials are monic.

$$\Rightarrow \max_{-1\leq x\leq 1} |\tilde{T}_n(x_i)| \leq \max_{-1\leq x\leq 1} |p_n(x)|$$

2.8 The CG Rate of Convergence

When choosing an iterative method to solve a specific problem it is important to take into consideration the speed at which the method will reach the exact solution. In numerical analysis, this speed is called the rate of convergence and the fewer iterations are needed to reach the solution, the higher the rate of convergence and the better is the method.

An indicator of the difficulty in solving the system $Ax = b$ with the conjugate gradient method is the condition number. The condition number is defined as the product of the norm of A times the norm of A^{-1} [4]:

$$\kappa(A) = \|A\|\|A^{-1}\|$$

Assuming that we know only the 2-norm condition number κ of a symmetric positive definite matrix, $\kappa(A) = \|A\|_2\|A^{-1}\|_2$ we apply the CG to the matrix problem $Ax = b$ and we find that the A-norms of the errors satisfy the following relationship [11, 13]:

$$\frac{\|e_k\|_A}{\|e_0\|_A} \leq 2/\left[\left(\frac{\sqrt{\kappa}+1}{\sqrt{\kappa}-1}\right)^k + \left(\frac{\sqrt{\kappa}+1}{\sqrt{\kappa}-1}\right)^{-k}\right] \leq 2\left(\frac{\sqrt{\kappa}-1}{\sqrt{\kappa}+1}\right)^k \tag{2.18}$$

If x_k is the k^{th} output of the CG, the above relationship is equivalent to

$$\|x - x_k\|_A \leq 2\left(\frac{\sqrt{\kappa}-1}{\sqrt{\kappa}+1}\right)^k \|x - x_0\|_A \tag{2.19}$$

In Section 2.6.4 we showed that

$$\frac{\|e_k\|_A}{\|e_0\|_A} \leq \min_{p\in\tilde{P}_k, p(0)=1} \max_{1\leq i\leq n} |p(\lambda_i)|,$$

where λ_i were the eigenvalues of the symmetric positive definite matrix A. Using this fact, we can prove equation (2.18) by finding a polynomial $p \in \tilde{P}_n$ whose maximum value is the middle expression of (2.18).

The polynomial we choose is:

$$p(x) = \frac{T_n\left(\gamma - \frac{2x}{\lambda_{max}-\lambda_{min}}\right)}{T_n(\gamma)},$$

where $\gamma = \frac{\lambda_{max}+\lambda_{min}}{\lambda_{max}-\lambda_{min}}$, or using the condition number defined as the ratio of the maximum eigenvalue over the minimum eigenvalue: $\kappa = \frac{\lambda_{max}}{\lambda_{min}}$, $\gamma = \frac{\kappa+1}{\kappa-1}$; and T_n is the standard Chebyshev polynomial.

The argument of T_n in the numerator of $p(x)$ lies in $[-1, 1]$. To prove relation (2.18) it is sufficient to show that

$$T_n(\gamma) = T_n\left(\frac{\kappa+1}{\kappa-1}\right) = \frac{1}{2}\left[\left(\frac{\sqrt{\kappa}+1}{\sqrt{\kappa}-1}\right)^n + \left(\frac{\sqrt{\kappa}+1}{\sqrt{\kappa}-1}\right)^{-n}\right]$$

And we will do so by using the Chebyshev polynomial of the form $T_n(x) = \frac{1}{2}(z^n + z^{-n})$ with the following change of variable $x = \frac{1}{2}(z + z^{-1})$.

If we multiply both sides of the equation $\frac{\kappa+1}{\kappa-1} = \frac{1}{2}(z - z^{-1})$, by z and we obtain the quadratic equation:

$$\frac{1}{2}z^2 - \frac{\kappa+1}{\kappa-1}z + \frac{1}{2} = 0,$$

with the positive solution

$$z = \frac{\kappa+1}{\kappa-1} + \sqrt{(\frac{\kappa+1}{\kappa-1})^2 - 1},$$

or after more calculations

$$z = \frac{\sqrt{\kappa}+1}{\sqrt{\kappa}-1}.$$

If we replace the solution z in the Chebyshev polynomial $T_n(\gamma)$ we have the polynomial we were looking for:

$$T_n(\gamma) = \frac{1}{2}\left[\left(\frac{\sqrt{\kappa}+1}{\sqrt{\kappa}-1}\right)^n + \left(\frac{\sqrt{\kappa}+1}{\sqrt{\kappa}-1}\right)^{-n}\right]$$

Therefore, the relation (2.19) holds true and implies that the convergence in the A-norm will be very fast if the condition number is one.

We can conclude now that the conjugate gradient method will work very well on matrices that are well-conditioned, and which have the condition number close to one. This condition number plays an important role in iterative methods. The closer its value to one, the better the convergence of that method. In the next chapter we will approximate the system $Ax = b$ with a related system $\tilde{A}x = \tilde{b}$, where we will try to make $\tilde{A}$ close to the identity and therefore, make its condition number, $\kappa(\tilde{A})$, closer to 1 than the condition number of the initial matrix A, $\kappa(A)$. [1, 2, 6]

Chapter 3

The Preconditioned Conjugate Gradient

As we have seen in the previous chapter, the conjugate gradient method works very well on matrices that are well-conditioned (i.e. condition number is not too big); however, in real applications, most matrices are ill-conditioned (i.e. the condition number is large), reducing the efficiency of the algorithm.

In this chapter we will discuss how a different version of the conjugate gradient method can be used for ill-conditioned large sparse symmetric positive-definite systems.

3.1 The Concept of Preconditioning

Preconditioning is an important technique used to develop an efficient conjugate gradient method solver for challenging problems in scientific computing [2]. The technique comes in the picture when we try to solve a linear system with a very large condition number.

According to equation (2.19), the larger the condition number of a SPD matrix A is, the slower the conjugate gradient method will converge.

The idea behind preconditioning is using the CG on an equivalent system. Thus, instead of solving $Ax = b$ we solve a related problem $\tilde{A}\tilde{x} = \tilde{b}$, for which $\tilde{A}$ is chosen

such that its condition number is closer to one; in other words, $\tilde{A}$ is close to the identity.

3.2 Analysis of the Preconditioner

As we have already mentioned, the approach to preconditioning is using the conjugate gradient algorithm to solve a transformed system of the original one.
Let $\tilde{A}\tilde{x} = \tilde{b}$ be the transformed system of $Ax = b$. And we relate the two systems through the following relationships:

$$\tilde{A} = B^{1/2}A$$

$$\tilde{x} = B^{1/2}y$$

$$\tilde{b} = B^{1/2}b$$

where we picked $B^{1/2}$ to be a symmetric positive-definite matrix and $y = B^{-1/2}x$. B is called a preconditioner.

Making a change of notation, we let $C = B^{1/2}AB^{1/2}$. Then, instead of solving $Ax = b$, we will have to solve a related problem:

$$\begin{cases} Cy = B^{1/2}b \\ x = B^{1/2}y \end{cases} \tag{3.1}$$

Since $B^{1/2}$ and A are both SPD, C is also SPD. Hence, the conjugate gradient algorithm can be applied to the system (3.1).

In this conditions, the inequality (2.19) becomes:

$$\|y - y_k\|_C \leq 2\left(\frac{\sqrt{\kappa(C)} - 1}{\sqrt{\kappa(C)} + 1}\right)^k \|y - y_0\|_C \tag{3.2}$$

With $x_k = B^{1/2}y_k$, we find that the C-norm of the approximate solution, y_k, is

the same as the A-norm of the approximate solution, x_k:

$$\begin{aligned}\|y - y_k\|_C &= \sqrt{(y-y_k)^T C(y-y_k)} \\ &= \sqrt{(y-y_k)B^{1/2}AB^{1/2}(y-y_k)} \\ &= \sqrt{[B^{1/2}(y-y_k)]^T A[B^{1/2}(y-y_k)]} \\ &= \sqrt{(x-x_k)^T A(x-x_k)} \\ &= \|x - x_k\|_A\end{aligned}$$

Then equation (3.2) will become:

$$\|x - x_k\|_A \leq 2\left(\frac{\sqrt{\kappa(C)}-1}{\sqrt{\kappa(C)}+1}\right)^k \|x - x_0\|_A$$

The condition number of matrix C can also be calculated from its maximum and minimum eigenvalues, $\lambda_{max}(C)$ and $\lambda_{min}(C)$, respectively: $\kappa(C) = \frac{\lambda_{max}(C)}{\lambda_{min}(C)}$

And from the change of notation we made for C:

$$\kappa(C) = \frac{\lambda_{max}(C)}{\lambda_{min}(C)} = \frac{\lambda_{max}(B^{1/2}AB^{1/2})}{\lambda_{min}(B^{1/2}AB^{1/2})}$$

By matrix conjugation: $B^{1/2}CB^{-1/2} = B^{1/2}(B^{1/2}AB^{1/2})B^{-1/2} = BA$, we can conclude that matrices BA and C have the same eigenvalues; then the condition number of C can also be calculated as the ratio between the maximum and minimum eigenvalues of the matrix BA: $\kappa(C) = \frac{\lambda_{max}(BA)}{\lambda_{min}(BA)}$.

Then, for the condition number of matrix C from the system (3.1), to be closer to 1 than the condition number of matrix A from the system $Ax = b$, we only need to find a preconditioner $B^{1/2}$ for which the ratio $\frac{\lambda_{max}(BA)}{\lambda_{min}(BA)}$ will be the closest to one.

3.3 Naive Preconditioned Conjugate Gradient

In this section we will focus on deriving the algorithm for the new preconditioned conjugate gradient method discussed in the above section.

We will start by appying the CG algorithm to $B^{1/2}AB^{1/2}y = B^{1/2}b$ with initial guess $y_0 = B^{-1/2}x_0$ to find y_k, and then use this to compute $x_k = B^{1/2}y_k$, the best approximation to the exact solution, x.

Remark: In our calculations, $B^{1/2}$ or B^{-1} doesn't need to be computed.

The CG algorithm applied to the problem:

$$\begin{cases} Cy = B^{1/2}b \\ y_0 = B^{-1/2}x_0 \end{cases} \quad (\ddagger)$$

finds the solution with the following formulas:

$$solution : y_k = y_{k-1} + \tilde{\alpha}_k \tilde{p}_{k-1} \tag{3.3}$$

$$residual : \tilde{r}_k = \tilde{r}_{k-1} - \tilde{\alpha}_k C\tilde{p}_{k-1} \tag{3.4}$$

$$search\ direction : \tilde{p}_k = \tilde{r}_k + \tilde{\beta}_k \tilde{p}_{k-1} \tag{3.5}$$

$$improvement : \tilde{\beta}_k = \frac{\tilde{r}_k^T \tilde{r}_k}{\tilde{r}_{k-1}^T \tilde{r}_{k-1}} \tag{3.6}$$

$$step\ length : \tilde{\alpha}_k = \frac{\tilde{r}_{k-1}^T \tilde{r}_{k-1}}{\tilde{p}_{k-1}^T C\tilde{p}_{k-1}} \tag{3.7}$$

A relationship between the initial residual $\tilde{r}_0$ of $(\ddagger)$ system and the initial residual r_0 of $Ax = b$ both solved with the CG method, can be found in the following way:

$B^{1/2}b - Cy_0 = \tilde{r}_0$

- multiply both sides of $B^{1/2}b - B^{1/2}AB^{1/2}y_0 = \tilde{r}_0$ by $B^{-1/2}$ from the left to obtain $b - A(B^{1/2}y_0) = B^{-1/2}\tilde{r}_0$ or $b - Ax_0 = B^{-1/2}\tilde{r}_0$.

Thus, $r_0 = B^{-1/2}\tilde{r}_0$.

Generalizing, we obtain $r_k = B^{-1/2}\tilde{r}_k$. $(\star)$

In all the computations to follow, we will substitute $B^{1/2}\tilde{p}_k$ with p_k.

If we replace y_k with $B^{-1/2}x_k$ in equation (3.3), then the equation becomes:

$B^{-1/2}x_k = B^{-1/2}x_{k-1} + \tilde{\alpha}_k \tilde{p}_{k-1}$

Multiplying both sides by $B^{1/2}$ from the left, equation (3.3) is further transformed

into:

$x_k = x_{k-1} + \tilde{\alpha}_k B^{1/2}\tilde{p}_{k-1}$, or $\boxed{x_k = x_{k-1} + \tilde{\alpha}_k p_{k-1}}$

Also, the equation (3.4) will become:

$B^{1/2}r_k = B^{1/2}r_{k-1} - \tilde{\alpha}_k C\tilde{p}_{k-1}$, or

$B^{1/2}r_k = B^{1/2}r_{k-1} - \tilde{\alpha}_k B^{1/2}AB^{1/2}\tilde{p}_{k-1}$

We multiply both sides from the left by $B^{-1/2}$ and get that $r_k = r_{k-1} - \tilde{\alpha}_k AB^{1/2}\tilde{p}_{k-1}$,

or $\boxed{r_k = r_{k-1} - \tilde{\alpha}_k Ap_{k-1}}$

Next, we multiply equation (3.5) by $B^{1/2}$ from the left: $B^{1/2}\tilde{p}_k = B^{1/2}\tilde{r}_k + \tilde{\beta}_k B^{1/2}\tilde{p}_{k-1}$,

to become:

$p_k = B^{1/2}\tilde{r}_k + \tilde{\beta}_k p_{k-1}$, or

$p_k = B^{1/2}(B^{1/2}r_k) + \tilde{\beta}_k p_{k-1}$, or $\boxed{p_k = Br_k + \tilde{\beta}_k p_{k-1}}$

Using the relationship found between the residuals of the two systems, eq. $(\star)$, we also find a new formula for the improvement $\tilde{\beta}_k$:

$$\tilde{\beta}_k = \frac{(B^{1/2}r_k)^T(B^{1/2}r_k)}{(B^{1/2}r_{k-1})^T(B^{1/2}r_{k-1})} = \frac{r_k^T B^{1/2}B^{1/2}r_k}{r_{k-1}^T B^{1/2}B^{1/2}r_{k-1}}$$

or $\boxed{\tilde{\beta}_k = \dfrac{r_k^T Br_k}{r_{k-1}^T Br_{k-1}}}$

Similarly, the equation (3.7) becomes: $\boxed{\tilde{\alpha}_k = \dfrac{r_{k-1}^T Br_{k-1}}{p_{k-1}^T Ap_{k-1}}}$

Consequently, the new conjugate gradient algorithm that is preconditioned becomes:

- approximate solution: $x_k = x_{k-1} + \tilde{\alpha}_k p_{k-1}$

- residual: $r_k = r_{k-1} - \tilde{\alpha}_k Ap_{k-1}$

- search direction: $p_k = Br_k + \tilde{\beta}_k p_{k-1}$

- improvement at step k: $\tilde{\beta}_k = \dfrac{r_k^T Br_k}{r_{k-1}^T Br_{k-1}}$

- step length: $\tilde{\alpha}_k = \dfrac{r_{k-1}^T Br_{k-1}}{p_{k-1}^T Ap_{k-1}}$

Chapter 4

Numerical Experiments

In this chapter we will apply the two algorithms derived in Chapter 2 and Chapter 3 to a symmetric positive definite matrix A and compare the convergence behavior of the two methods.

We will illustrate this behavior on a problem of size $n = 1536$, arised from a new discontinuous Galerkin method for elliptic problems.

All computational results and experiments presented in this work were done using Matlab 7.5.0.

The data was generated from the discrete problem of the weakly over-penalized symmetric interior penalty method, discussed in [3]. For preconditioning we used two matrices: a simple sparse preconditioner (P_1), generated by taking the inverse of the diagonal entries of the matrix to be preconditioned, and a simple block-diagonal preconditioner (P_2), also presented in [3].

On this data, we used the Matlab codes from Table 4.2 and Table 4.3, to compute an approximate solution vector x to the exact solution of the system. The program stops when either a certain tolerance is reached by the error (in our program 10^-6) or the number of iterations is higher than the maximum of 100 and $\sqrt{n}$.

Then, we made a comparison between the two methods and present the results in

Method	Iterations	κ_2
CG	78	$5.8920e+04$
P_1CG	67	$3.4079e+04$
P_2CG	55	$0.0472e+04$

Table 4.1: Numerical results for the conjugate gradient and preconditioned conjugate gradient methods

Table 4.1. In our analysis, we considered the condition number of the original system, $Ax = b$, and the condition number of the preconditioned system: $B^{1/2}AB^{1/2}x = B^{1/2}b$.

As we have expected, the preconditioned conjugate gradient method performed better than the conjugate gradient method. Both preconditioners took less number of iterations and their condition number were less than of than that of the CG algorithm.

The better rate of convergence of the preconditioned system versus the original system is confirmed: the condition number for the system solved with CG was $5.8920e+04$, while for the system solved with PCG were $3.4079e+04$ for the first preconditioner P_1 and $0.0472e+04$ for the second preconditioner P_2.

Comparing the two preconditioners, we observe that carefully choosing a preconditioner with a lower condition number, the convergence of the system is improved, as well as the number of iterations required to compute the solution.

```
function [x,iter,r]=CG(A, b)
n = length(b);
x = zeros(n,1);
maxiters = max(100,sqrt(n));
r = b;
M = r'*r;
p = r;
iter=0;
while norm(r)/norm(b) > 1e-6 &&  iter < maxiters
    iter=iter+1
    alpha = M / (p'*A*p);
    x = x + alpha * p;
    r = r - alpha * A*p;
    Mold = M;
    M = r'*r;
    beta = M / Mold;
    p = r + beta * p;
end;
```

Table 4.2: Conjugate Gradient Algorithm in Matlab

```
function [x,iter,r]=PCG(A, b,B)
n = length(b);
x = zeros(n,1);
iter =0;
maxiters = max(100,sqrt(n));
r = b;
M = r'*B*r;
p = B*r;
while norm(r)/norm(b) > 1e-6 &&  iter < maxiters
    iter=iter+1;
    alpha = M / (p'*A*p);
    x = x + alpha * p;
    r = r - alpha * A*p;
    Mold = M;
    M = r'*B*r;
    beta = M / Mold;
    p = B*r + beta * p;
end;
```

Table 4.3: Preconditioned Conjugate Gradient Algorithm in Matlab

Chapter 5

Conclusions

Iterative methods are useful tools in solving large systems of linear equations, and therefore choosing the right method to solve your problem is important. The conjugate gradient method is one very useful strategy to solve a symmetric positive definite system, especially for sparse matrices. A good method requires fast convergence, simplicity of the algorithm, stability, little storage memory, and not lastly, a good estimate of the solution. A preconditioned conjugate gradient method satisfies all these requirements and it is therefore an effective iterative method.

Bibliography

[1] S. Ashby, T. Barth, T. Mantueffel and P. Saylor, *A Tutorial on Iterative Methods for Linear Algebraic Systems*, University of Illinois, 1996.

[2] M. Benzi, *Preconditioning Techniques for Large Linear Systems: A Survey*, Journal of Computational Physics, 182 (2002), pp. 418-477

[3] S. C. Brenner, L. Owens and L. Y. Sung, *A weakly over-penalized symmetric interior penalty method*, 2008, submitted.

[4] S. C. Brenner and L. Y. Sung, Math 526: Numerical Linear Algebra, Lecture Notes, Department of Mathematics of South Carolina, SC, 29208

[5] G. H. Golub and D. P. O'Leary, *Some History of the Conjugate Gradient and Lanczos Algorithms: 1948-1976*, SIAM Review, Vol. 31, No. 1, pp. 50-102, March 1989.

[6] G. H. Golub and C. F Van Loan, *Matrix computations*, 3rd ed., The Johns Hopkins University Press, 1996.

[7] W. H. Hager, *Applied Numerical Linear Algebra*, Prentice Hall, New Jersey, 1988, pp. 319-353

[8] M. R. Hestenes and E. Stiefel, *Methods of Conjugate Gradients for Solving Linear Systems*, Journal of Research of the National Bureau of Standards, Vol. 49, No. 6, December 1952.

[9] D. Kincaid and W. Cheney, *Numerical Mathematics and Computing*, Thomson Learning, August 2003.

[10] W. J. Kammerer and M. Z. Nashed *On the Convergence of the Conjugate Gradient Method for Singular Linear Operator Equations*, SIAM Journal on Numerical Analysis, Vol. 9, No. 1. (Mar., 1972), pp. 165-181 .

[11] J. R. Shewchuk, *An Introduction to the Conjugate Gradient Method Without the Agonizing Pain*, August, 1994.

[12] A. van der Sluis and H.A. van der Vorst, *The Rate of Convergence of Conjugate Gradients*, Numerische Mathematik, Springer Berlin/Heidelberg, Vol. 48, No. 5, September, 1986.

[13] L. N. Trefethen and D. Bau, *Numerical Linear Algebra*, SIAM, Philadelphia, 1997, pp. 89-97; 293-303, 313-321.