TMA4220: Numerical solution of partial differential equations by element methods

The Conjugate Gradient Method for Finite Element Poisson Discretizations

February 17, 2003

Einar M. Rønquist
Department of Mathematical Sciences
NTNU, N-7491 Trondheim, Norway

1 The finite element method

As a model problem, consider the Poisson problem in a domain Ω with homogeneous Dirichlet boundary conditions,

$$-\nabla^2 u = f \quad in\ \Omega, \tag{1}$$

$$u = 0 \quad on\ \partial\Omega. \tag{2}$$

The corresponding weak formulation is: Find $u \in X = H_0^1(\Omega)$ such that

$$a(u,v) = l(v) \qquad \forall v \in X. \tag{3}$$

Here, $a(w,v)$ is the symmetric, positive definite bilinear form

$$a(w,v) = \int_\Omega \nabla u \cdot \nabla v \, d\Omega \ , \tag{4}$$

and $l(v)$ is the bounded linear form

$$l(v) = \int_\Omega f\, v \, d\Omega \ . \tag{5}$$

The Poisson problem can also be stated in terms of the minimization problem

$$u = \arg\min_{w \in X} \ J(w) = \frac{1}{2} a(w,w) - l(w) \ . \tag{6}$$

Since $a(\cdot,\cdot)$ is symmetric, positive definite, the quadratic functional $J(w)$ has a minimum; this minimum can be found by requiring that the first variation of $J(w)$ vanish, which results in the weak statement (3).

The finite element formulation of the Poisson problem consists of searching for an approximate solution u_h in a finite-dimensional (conforming) subspace X_h of X, i.e., $u_h \in X_h \subset X$. The space X_h can be defined in terms of linear finite elements, or in terms of higher-order finite elements or spectral elements; the total number of degrees-of-freedom is assumed to be N, i.e., $dim(X_h) = N$.

The discrete problem can then be stated as: Find $u_h \in X_h$ such that

$$a(u_h, v) = l(v) \qquad \forall v \in X_h \ , \tag{7}$$

while the corresponding minimization problem can be stated as

$$u_h = \arg\min_{w \in X_h} \ J(w) = \frac{1}{2} a(w,w) - l(w) \ . \tag{8}$$

Choosing a nodal basis for X_h, i.e., setting

$$X_h = span\{\phi_1, \phi_2, ...\phi_N\} \ , \tag{9}$$

$$u_h(x) = \sum_{j=1}^{N} u_j \phi_j(x) \ , \quad \phi_j(x_i) = \delta_{ij} \ , \tag{10}$$

we arrive at a set of algebraic equations of the form

$$\underline{A}\,\underline{u} = \underline{F} \ , \tag{11}$$

where

$$A_{ij} = a(\phi_i, \phi_j) \ , \tag{12}$$
$$F_i = l(\phi_i) \ , \tag{13}$$

and $\underline{u}^T = [u_1, u_2, ..., u_N]^T$ are the nodal values of u_h. Here, $\underline{A}$ is a symmetric, positive definite matrix of dimension N.

Finally, by using the chosen nodal basis for X_h, we also note that the algebraic system of equations, (11), corresponds to the minimization problem

$$\underline{u} = \arg \min_{\underline{w} \in R^N} J^N(\underline{w}) = \frac{1}{2}\underline{w}^T \underline{A}\,\underline{w} - \underline{w}^T \underline{F}. \tag{14}$$

In particular, (11) corresponds to setting the first variation of $J^N(\underline{w})$ equal to zero. Note that $J^N(\underline{w}) : R^N \rightarrow R$ is the same as $J(w_h) : X_h \rightarrow R$ since $J^N(\underline{w}) \equiv J(\sum_{j=1}^N w_j \phi_j)$; the only difference here is that J^N expresses the quadratic functional in terms of the basis coefficients, while J expresses it in terms of functions in X_h.

Once we have computed $\underline{u}$, we obtain u_h by using the chosen nodal basis (10).

2 The conjugate gradient method

2.1 Preliminaries

We will now explain how the conjugate gradient method works in the context of solving (11). Consider a subspace K^n of R^N spanned by a set of n linearly independent vectors $\underline{p}_1, \underline{p}_2, ..., \underline{p}_n$, that is

$$K^n = span\{\underline{p}_1, \underline{p}_2, ..., \underline{p}_n\} \quad . \tag{15}$$

Define the $N \times n$ matrix $\underline{P}$ as

$$\underline{P} = [\underline{p}_1, \underline{p}_2, ..., \underline{p}_n] \quad . \tag{16}$$

With this notation, we can write any element $\underline{w} \in K^n$ as

$$\underline{w} = \underline{P}\,\underline{z} \tag{17}$$

for some $\underline{z} \in R^n$ since the columns of $\underline{P}$ represent a basis for K^n.

We know that the solution of our system of equations $\underline{A}\,\underline{u} = \underline{F}$ can be found my minimizing the quadratic functional $J^N(\underline{w})$ over all $\underline{w} \in R^N$. However, if we limit our minimization to elements in $K^n \subset R^N$, this corresponds to minimizing

$J^N(\underline{w})$ only over elements $\underline{w}$ that can be expressed in the form (17). Hence, the problem: Find $\underline{u}_n \in K^n$ such that

$$\underline{u}_n = \arg \min_{\underline{w} \in K^n} J^N(\underline{w}) \tag{18}$$

can be restated as: Find $\underline{u}_n = \underline{P}\,\underline{a} \in K^n$ where

$$\underline{a} = \arg \min_{\underline{z} \in R^n} J^N(\underline{w}(\underline{z})) \tag{19}$$

and

$$J^N(\underline{w}(\underline{z})) \equiv J^n(\underline{z}) = \frac{1}{2}\underline{z}^T \underline{P}^T \underline{A}\,\underline{P}\,\underline{z} - \underline{z}^T \underline{P}^T \underline{F} \tag{20}$$

Similar to the original problem (14), the solution to (19)-(20) is found by requiring that the first variation of $J^n(\underline{z})$ is equal to zero, that is,

$$\underline{P}^T \underline{A}\,\underline{P}\,\underline{a} = \underline{P}^T \underline{F} \quad . \tag{21}$$

In order to proceed, let us now further assume that the vectors $\{\underline{p}_i\}$ are $\underline{A}$-orthogonal (or conjugate) in the sense that

$$\begin{aligned} \underline{p}_i^T \underline{A}\,\underline{p}_j &= 0 \ , \ i \neq j \, , \\ &= \pi_i \ , \ i = j \, . \end{aligned} \tag{22}$$

This $\underline{A}$-orthogonality (or conjugacy) can also be expressed succinctly as

$$\underline{P}^T \underline{A}\,\underline{P} = \underline{D} \equiv diag(\pi_1, \pi_2, ..., \pi_n) \ . \tag{23}$$

Note that $\underline{D}$ is a *diagonal*, symmetric, positive definite, $n \times n$ matrix.

By substituting (23) into (20), equation (19) can be expressed as

$$\underline{a} = \arg \min_{\underline{z} \in R^n} J^n(\underline{z}) = \frac{1}{2}\underline{z}^T \underline{D}\,\underline{z} - \underline{z}^T \underline{P}^T \underline{F}, \tag{24}$$

and (21) simply becomes

$$\underline{D}\,\underline{a} = \underline{P}^T \underline{F} \quad . \tag{25}$$

We now notice that, *unlike* the original problem, solving for $\underline{a}$ is trivial since $\underline{D}$ is diagonal; we simply get

$$\underline{a} = \underline{D}^{-1} \underline{P}^T \underline{F} = \underline{D}^{-1} \underline{P}^T \underline{A}\,\underline{u} \quad , \tag{26}$$

where we have replaced $\underline{F}$ with $\underline{A}\,\underline{u}$. Componentwise, we get

$$a_i = \frac{\underline{p}_i^T \underline{A}\,\underline{u}}{\underline{p}_i^T \underline{A}\,\underline{p}_i} = \frac{\gamma_i}{\pi_i} \quad , i = 1, ..., n \ , \tag{27}$$

where we have used (22) and defined

$$\gamma_i \equiv \underline{p}_i^T \underline{A}\,\underline{u} \ . \tag{28}$$

Having found $\underline{a}$ in (19), the corresponding solution $\underline{u}_n \in K^n$ is

$$\underline{u}_n = \underline{P}\,\underline{a} = \sum_{i=1}^{n} a_i \underline{p}_i \quad . \tag{29}$$

In the particular case when $n = N$, the basis vectors in K^n span R^N, and the minimization problem (14) and (18) coincide; the exact basis coefficients $\underline{u}$ can thus be expressed as

$$\underline{u} = \underline{u}_N = \sum_{i=1}^{N} a_i \underline{p}_i \,. \tag{30}$$

Since the A-norm of $\underline{u}$ is given by $\| \underline{u} \|_A^2 = \underline{u}^T \underline{A}\, \underline{u}$ $(= a(u_h, u_h))$, we get

$$\| \underline{u} \|_A^2 = \underline{u}^T \underline{A}\, \underline{u} = \sum_{i=1}^{N} \frac{\gamma_i^2}{\pi_i} \,. \tag{31}$$

The difference between the exact solution $\underline{u} \in R^N$ and $\underline{u}_n \in K^n$ is

$$\underline{e}_n \equiv \underline{u} - \underline{u}_n = \sum_{i=1}^{N} a_i \underline{p}_i - \sum_{i=1}^{n} a_i \underline{p}_i = \sum_{i=n+1}^{N} a_i \underline{p}_i . \tag{32}$$

We have thus shown that the error $\underline{u} - \underline{u}_n$ is in $span\{\underline{p}_{n+1}, ..., \underline{p}_N\}$. Using the $\underline{A}$-orthogonality (22) of the basis vectors, we immediately obtain

$$\| \underline{u} - \underline{u}_n \|_A^2 = \sum_{i=n+1}^{N} a_i^2 \pi_i = \sum_{i=n+1}^{N} \frac{\gamma_i^2}{\pi_i} \tag{33}$$

and

$$(\underline{u} - \underline{u}_n)^T \underline{A}\, \underline{v} = 0 \qquad \forall \underline{v} \in K^n \,. \tag{34}$$

Hence, $\underline{u}_n$ represents an $\underline{A}$-orthogonal projection of the exact solution $\underline{u} \in R^N$ onto the subspace K^n, and the error is $\underline{A}$-orthogonal to this space. It also follows that the error

$$\| \underline{u} - \underline{u}_n \|_A \leq \| \underline{u} - \underline{v} \|_A \qquad \forall \underline{v} \in K^n . \tag{35}$$

Note that the coefficients $\underline{a}$ can simply be regarded as generalized Fourier coefficients; in particular, (31) is the analogue of Parseval's formula for functions.

We now make some comments on the orthogonality condition (34) and the optimality (projection) condition (35). As long as the basis vectors $\{\underline{p}_i\}$ are linearly independent, K^n will be a larger space than K^{n-1}, and the error $\| \underline{e}_n \|_A$ will decrease as we increase $dim(K^n) = n$. Note that we have tacitly assumed that we can easily construct an expanding set of $\underline{A}$-orthogonal basis vectors; the conjugate gradient method is able to meet this objective. Armed with a set of conjugate search directions, the sequence $\underline{u}_n$, $n = 0, 1, ..$ will represent a sequence of improved approximations to the exact solution $\underline{u}$ of (11).

Since $K^N = R^N$, the exact solution $\underline{u}$ will be obtained in no more than N iterations (in exact arithmetic). In this respect, the conjugate gradient method can also be regarded as a direct method. In fact, this is how the method was first considered. However, what makes the conjugate gradient method attractive in practice is that an acceptable iterate $\underline{u}_n$ can often be achieved for $n \ll N$.

The importance of generating an $\underline{A}$-orthogonal basis is due to the fact that the *global* minimization problem (18) can be made into a series of sequential (or *local*) minimization problems along each of the basis vectors. Specifically, if we know $\underline{u}_{n-1}$, the next approximation $\underline{u}_n$ will automatically be of the form

$$\underline{u}_n = \underline{u}_{n-1} + a_n \underline{p}_n \,. \tag{36}$$

Hence, in (36) we just need to find the value of the new coefficient a_n which minimizes the quadratic functional J^n along $\underline{p}_n$; see Exercise 3. Again, note that this is true *only* if we have access to an $\underline{A}$-orthogonal basis, because the optimal value a_n can then be computed *independently* of a_i, $i = 1, ..., n-1$ via (26).

Note that the error $\| \underline{e}_n \|_A > 0$ as long as K^n does not span the entire space R^N. However, if the solution $\underline{u}$ happens to belong to a subspace of R^N, and K^n spans this subspace for some $n < N$, then $\| \underline{e}_n \|_A = 0$ from the optimality condition (35). That is, we obtain the exact solution in *less* than N iterations (assuming exact arithmetic).

We summarize some of the key points:

$$u = \arg \min_{w \in X} \; J(w) = \frac{1}{2} a(w,w) - l(w) \qquad u \in X = H_0^1(\Omega) \tag{37}$$

$$u_h = \arg \min_{w_h \in X_h} \; J(w) \qquad u_h \in X_h \subset X \tag{38}$$

$$\underline{u} = \arg \min_{\underline{w} \in R^N} \; J^N(\underline{w}) = \frac{1}{2} \underline{w}^T \underline{A}\, \underline{w} - \underline{w}^T \underline{F} \qquad \underline{u} \in R^N \tag{39}$$

$$\underline{u}_n = \arg \min_{\underline{w} \in K^n} \; J^N(\underline{w}) \qquad \underline{u}_n \in K^n \subset R^N \tag{40}$$

The conjugate gradient method is characterized by the minimization statement (40), and where $\underline{u}_n$ is easy to obtain if K^n is spanned by n linearly independent vectors which are *conjugate*. In the context of the finite element method, the minimization statement (39) is equivalent to the minimization statement (38) via the chosen nodal basis for X_h. Furthermore, we note that the discrete solution u_h only represents a minimization of $J(w)$ over X_h and not over the entire space X. Hence, in the finite element context, the conjugate gradient method represents a minimization within a minimization. For many finite element applications, it thus suffices to iterate only until the incomplete iteration error $\| \underline{u} - \underline{u}_n \|_A$ is at the level of the discretization error $| u - u_h |_{H^1}$.

2.2 Bases for K^n

What remains now is to find a computationally efficient way to construct the subspace K^n. One way would, of course, be to find a set of n linearly independent vectors and to $\underline{A}$-orthogonalize these via a Gram-Schmidt procedure. However, this would be a very expensive method, both in terms of computational effort and in terms of memory requirement. As we will discuss shortly, the remarkable fact is that going from K^{n-1} to K^n can be achieved without storing all the previous search directions; simple recurrence relations will suffice.

In practice, the way we span the space K^n is by first starting with the initial residual $\underline{r}_0$. Without any loss of generality, we will assume that the initial guess $\underline{u}_0 = 0$ so that $\underline{r}_0 = \underline{F}$. The alternative is to set $\underline{r}_0 = \underline{F} - \underline{A}\,\underline{u}_0$, however, we can always absorb the term $-\underline{A}\,\underline{u}_0$ into the known right hand side for our system.

Starting with the initial residual $\underline{r}_0$, we generate a sequence of vectors $\underline{r}_0, \underline{A}\,\underline{r}_0, \underline{A}^2\underline{r}_0, \ldots\ldots$, that is, we generate the next member of the sequence by operating upon the previous vector by the matrix $\underline{A}$. The subspace K^n is then

$$K^n = span\{\underline{r}_0, \underline{A}\,\underline{r}_0, \ldots., \underline{A}^{n-1}\underline{r}_0\} \equiv K^n(\underline{A}; \underline{r}_0). \tag{41}$$

A space generated in this fashion is called a *Krylov* space, and is denoted as $K^n(\underline{A}; \underline{r}_0)$ in order to indicate that the "seed" vector is $\underline{r}_0$, and the matrix $\underline{A}$ is used as a "generator" for the basis vectors. It is obviously a convenient way to generate K^n since $\underline{r}_0$ is a known vector, and only matrix-vector products (operator evaluations) with the known (and sparse) matrix $\underline{A}$ are required. We will show that, as long as the residual $\underline{r}_{n-1} = \underline{F} - \underline{A}\,\underline{u}_{n-1} \neq 0$, the vectors $\underline{r}_0, \underline{A}\,\underline{r}_0, \ldots.\underline{A}^{n-1}\underline{r}_0$ are indeed linearly independent, and hence K^n is a larger space than K^{n-1} (K^{n-1} is a subspace of K^n).

The basis for K^n given in (41) is not $\underline{A}$-orthogonal. We thus need to find an efficient way to generate an alternative basis for K^n that is $\underline{A}$-orthogonal. Before we give the details on how to do this, we consider yet another basis for K^n. To this end, consider the residual $\underline{r}^n$ where

$$\underline{r}_n = \underline{F} - \underline{A}\,\underline{u}_n = \underline{r}_0 - \underline{A}\,\underline{u}_n \quad . \tag{42}$$

By construction, since $\underline{u}_n \in K^n$, $\underline{A}\,\underline{u}_n \in K^{n+1}$. Also, by construction, $\underline{r}_0 \in K^{n+1}$. Hence

$$\underline{r}_n \in K^{n+1} \quad . \tag{43}$$

Now, consider the orthogonality condition (34).

$$\begin{aligned} \forall \underline{v} \in K^n, \quad 0 &= \underline{v}^T \underline{A}(\underline{u} - \underline{u}_n) \\ &= \underline{v}^T(\underline{A}\,\underline{u} - \underline{A}\,\underline{u}_n) \\ &= \underline{v}^T(\underline{F} - \underline{A}\,\underline{u}_n) \\ &= \underline{v}^T\underline{r}_n \quad . \end{aligned} \tag{44}$$

Since $\underline{r}_n \in K^{n+1}$, we can choose $\underline{v}$ to be $\underline{r}_0, \underline{r}_1, \ldots, \underline{r}_{n-1}$ in (44). Thus, we have demonstrated that the residuals are mutually orthogonal, i.e.,

$$\begin{aligned} \underline{r}_i^T\underline{r}_j &= 0 \quad , i \neq j \\ &= \rho_i \quad i = j \quad . \end{aligned} \tag{45}$$

As long as $\underline{r}_{n-1} \neq \underline{0}$, $\{\underline{r}_0, \underline{r}_1, ..., \underline{r}_{n-1}\}$ form an *orthogonal* basis for K^n, and $dim(K^n) = n$. This also demonstrates that the vectors $\{\underline{r}_0, \underline{A}\,\underline{r}_0, .., \underline{A}^{n-1}\underline{r}_0\}$ in (41) really form a set of linearly independent basis vectors. Finally, we know that there exists an $\underline{A}$-orthogonal basis for K^n, and we thus see that

$$\begin{aligned} K^n &= span\{\underline{r}_0, \underline{A}\underline{r}_0, ..., \underline{A}^{n-1}\underline{r}_0\} \\ &= span\{\underline{r}_0, \underline{r}_1, ..., \underline{r}_{n-1}\} \\ &= span\{\underline{p}_1, \underline{p}_2, ..., \underline{p}_n\} \quad . \end{aligned} \tag{46}$$

2.3 Derivation of the algorithm

We are now ready to derive all the details of the conjugate gradient algorithm. Let M be the smallest integer such that $\underline{r}_M \neq \underline{0}$ (note that $M < N$). For each $n \leq M$, we express

$$\underline{p}_n = \sum_{j=0}^{n-1} b_j\, \underline{r}_j \in K^n \tag{47}$$

and

$$\underline{u}_n = \sum_{j=1}^{n} a_j\, \underline{p}_j \in K^n \quad . \tag{48}$$

We choose $\{\underline{r}_j\}$, $j = 0, ..., n-1$ as a basis for $\underline{p}_n$ since the residual vectors are mutually orthogonal and $\underline{r}_{n-1} \in K^n$. This ensures that $\underline{p}_n \in K^n$ which is needed for the next iteration. We have earlier found that the coefficients $\underline{a}$ are given by $a_i = \gamma_i/\pi_i$, $i = 1, ..., n$. Since the residual vectors are mutually orthogonal, it also follows that (standard Euclidean projection)

$$b_i = \frac{\underline{r}_i^T \underline{p}_n}{\underline{r}_i^T \underline{r}_i} \quad , i = 0, ..., n-1. \tag{49}$$

But

$$\underline{p}_n^T \underline{r}_i = \underline{p}_n^T\, \underline{A}\, \underline{u} = \gamma_n \quad , i = 0, ..., n-1 \quad , \tag{50}$$

which implies that $b_i = \gamma_n/\rho_i$. Hence,

$$\underline{p}_n = \gamma_n \sum_{j=0}^{n-1} \frac{1}{\rho_j}\, \underline{r}_j \tag{51}$$

$$\underline{u}_n = \sum_{j=1}^{n} \frac{\gamma_j}{\pi_j}\, \underline{p}_j \quad . \tag{52}$$

We can choose the length of the search directions $\underline{p}_n$ arbitrarily; see Exercise 4. A convenient choice is $\gamma_n = \rho_{n-1}$. With this choice we have

$$\underline{p}_n = \underline{r}_{n-1} + \beta_n\, \underline{p}_{n-1} \tag{53}$$

$$\underline{u}_n = \underline{u}_{n-1} + \alpha_n\, \underline{p}_n \quad , \tag{54}$$

where

$$\alpha_n = \rho_{n-1}/\pi_n \tag{55}$$
$$\beta_n = \rho_{n-1}/\rho_{n-2} \tag{56}$$
$$\rho_n = \underline{r}_n^T \underline{r}_n \tag{57}$$
$$\pi_n = \underline{p}_n^T \underline{A}\,\underline{p}_n \tag{58}$$

Since

$$\underline{r}_n = \underline{r}_{n-1} - \alpha_n \underline{A}\,\underline{p}_n \ , \tag{59}$$

we can also compute the residuals recursively. To start the whole procedure, we set

$$\underline{p}_1 = \underline{r}_0 \quad . \tag{60}$$

2.4 The conjugate gradient algorithm

$\underline{u}_0 = \underline{0}$; $\underline{r}_0 = \underline{F}$;

For $k = 1, 2, ...$

$$\beta_k = \underline{r}_{k-1}^T \underline{r}_{k-1} / \underline{r}_{k-2}^T \underline{r}_{k-2} \qquad (\beta_1 \equiv 0) \tag{61}$$
$$\underline{p}_k = \underline{r}_{k-1} + \beta_k \underline{p}_{k-1} \qquad (\underline{p}_1 \equiv \underline{r}_0) \tag{62}$$
$$\alpha_k = \underline{r}_{k-1}^T \underline{r}_{k-1} / \underline{p}_k^T \underline{A}\,\underline{p}_k \tag{63}$$
$$\underline{u}_k = \underline{u}_{k-1} + \alpha_k \underline{p}_k \tag{64}$$
$$\underline{r}_k = \underline{r}_{k-1} - \alpha_k \underline{A}\,\underline{p}_k \ . \tag{65}$$

End for

2.5 Computational complexity

Based on the above algorithm, we can estimate the computational complexity. First, for each iteration, we identify the following basic types of operations:

(a) one matrix-vector product of the type $\underline{y} = \underline{A}\,\underline{x}$;

(b) two innerproducts (which ones?);

(c) multiplication of a vector with a scalar (how many?);

(d) addition of two vectors (how many?)

For a full matrix $\underline{A}$, the computational complexity associated with (a) is $O(N^2)$. However, in the context of finite elements, the matrix $\underline{A}$ is typically sparse. For example, if we use finite elements of order p, the number of non-zero entries per row will (in general) be $O(p^d)$ in R^d (why?). The operation count for (a) is thus $O(p^d\, N)$. For linear finite elements, $p = 1$, and the cost per iteration scales as $O(N)$.

The computational complexity associated with (b), (c) and (d) are all $O(N)$ operations. The total work per iteration is therefore dominated by the $O(p^d\, N)$ term associated with the matrix-vector product evaluation (except when $p = 1$).

Note that, in addition to the listed operations (a)-(d), there is also a cost associated with checking for convergence.

Finally, the total cost of using the conjugate gradient method is equal to the cost per iteration times the number of iterations, m; the latter is determined by the convergence property of the conjugate gradient algorithm which we consider next.

2.6 Convergence rate

Since $K^n = span\{\underline{r}_0, \underline{A}\,\underline{r}_0, ..., \underline{A}^{n-1}\underline{r}_0\}$, we can express any element $\underline{v} \in K^n$ as

$$\underline{v} = \sum_{j=0}^{n-1} c_j\, \underline{A}^j\, \underline{r}_0 \quad . \tag{66}$$

Since we assume that $\underline{u}_0 = \underline{0}$ (with no loss in generality), we can express the initial residual as

$$\underline{r}_0 = \underline{F} = \underline{A}\,\underline{u} \tag{67}$$

and thus

$$\underline{v} = \sum_j c_j\, \underline{A}^{(j+1)}\, \underline{u} \quad . \tag{68}$$

Hence,

$$\begin{aligned} \| \underline{u} - \underline{v} \|_A &= \| \underline{u} - \sum_j c_j\, \underline{A}^{(j+1)}\, \underline{u} \|_A \\ &= \| (\underline{I} - \sum_j c_j\, \underline{A}^{(j+1)})\, \underline{u} \|_A \\ &= \| p_n(\underline{A})\, \underline{u} \|_A \end{aligned} \tag{69}$$

where $p_n(x)$ is a polynomial of degree n such that $p_n(0) = 1$.

Combining this result with the optimality (projection) condition (35) gives

$$\| \underline{u} - \underline{u}_n \|_A = \min_{p_n \in P_n; p_n(0)=1} \| p_n(\underline{A})\, \underline{u} \|_A \tag{70}$$

Since $\underline{A}$ is an SPD matrix, $\underline{A}$ has a complete set of N orthogonal eigenvectors $\underline{q}_j$ with corresponding eigenvalues λ_j. We can expand the exact solution in terms

of the eigenvectors

$$\underline{u} = \sum_j d_j \underline{q}_j \quad , \tag{71}$$

where d_j are the expansion coefficients. It now follows that

$$\begin{aligned} p_n(\underline{A})\,\underline{u} &= \sum_j d_j\, p_n(\underline{A})\,\underline{q}_j \\ &= \sum_j d_j\, p_n(\lambda_j)\,\underline{q}_j \\ &\leq [\max_j p_n(\lambda_j)] \sum_j d_j\, \underline{q}_j \\ &= [\max_j p_n(\lambda_j)]\,\underline{u} \end{aligned} \tag{72}$$

Thus (70) becomes

$$\begin{aligned} \| \underline{u} - \underline{u}_n \|_A &= \min_{p_n \in P_n; p_n(0)=1} \| p_n(\underline{A})\,\underline{u} \|_A \\ &\leq \min_{p_n \in P_n; p_n(0)=1} \max_j |p_n(\lambda_j)| \ \| \underline{u} \|_A \\ &\leq \min_{p_n \in P_n; p_n(0)=1} \max_{\lambda \in [\lambda_{min}, \lambda_{max}]} |p_n(\lambda)| \ \| \underline{u} \|_A \end{aligned} \tag{73}$$

This is a classical min-max problem with the solution

$$\| \underline{u} - \underline{u}_n \|_A \leq 2(\frac{\sqrt{\kappa}-1}{\sqrt{\kappa}+1})^n \ \| \underline{u} \|_A \tag{74}$$

where $\kappa = \kappa(\underline{A}) = \lambda_{max}(\underline{A})/\lambda_{min}(\underline{A})$ is the condition number of $\underline{A}$.

We can also express this result as

$$\| \underline{e}_n \|_A \leq 2(\frac{\sqrt{\kappa}-1}{\sqrt{\kappa}+1})^n \ \| \underline{e}_0 \|_A \tag{75}$$

Hence, the initial error in the A-norm in reduced by a factor of $(\frac{\sqrt{\kappa}-1}{\sqrt{\kappa}+1})^n$. If κ is close to unity, we expect the conjugate gradient method to converge in $O(1)$ iterations. In the general case, we expect the method to converge in $m = O(\sqrt{\kappa})$ iterations, but limited to a maximum of N iterations (in exact arithmetic).

3 One-dimensional linear finite elements

Consider the one-dimensional Poisson equation $-u_{xx} = f$ in $\Omega = (0,1)$ with $u(0) = u(1) = 0$. Let us construct a test case where the exact solution is $u = e^x\, sin(\pi\, x)$. Note that this solution is a non-polynomial solution which can never be exactly represented by piecewise polynomials. The corresponding source term f is given as $f = -e^x\, sin(\pi\, x) - 2\,\pi\, e^x\, cos(\pi\, x) + \pi^2\, e^x\, sin(\pi\, x)$.

We discretize the domain into $K = N + 1$ linear finite elements with mesh size $h = 1/(N + 1)$, and arrive at the linear system of equations $\underline{A}\,\underline{u} = \underline{F}$, to be solved for the N interior nodal values $\underline{u}$. In order to evaluate the right hand side (the linear form $l(v)$ in (7)), we use one-point Gauss Legendre quadrature on each element. We solve the system of algebraic equations by the conjugate gradient method. The iteration is stopped when the L^2-norm of the residual vector, $(\underline{r}^T\underline{r})^{1/2}$ is less than $\varepsilon = 10^{-10}$ (double precision). Table 1 summarizes the results for various values of $K = N + 1$.

Table 1: One-dimensional Poisson problem; numerical results.

K	N	m	$\lambda_{min}(\underline{A})$	$\lambda_{max}(\underline{A})$	$\kappa(\underline{A})$	$\mid u - u_h \mid_{H^1}$	$\parallel \underline{\hat{u}} - \underline{u} \parallel_A$
800	799	799	$1.16\,10^{-2}$	$3.20\,10^{3}$	$2.76\,10^{5}$	$5.50\,10^{-3}$	$1.94\,10^{-6}$
400	399	399	$2.46\,10^{-2}$	$1.60\,10^{3}$	$6.52\,10^{4}$	$1.10\,10^{-2}$	$7.74\,10^{-6}$
200	199	199	$4.93\,10^{-2}$	$8.00\,10^{2}$	$1.62\,10^{4}$	$2.21\,10^{-2}$	$3.10\,10^{-5}$
100	99	99	$9.87\,10^{-2}$	$4.00\,10^{2}$	$4.05\,10^{3}$	$4.42\,10^{-2}$	$1.24\,10^{-4}$

Here, m denotes the number of conjugate gradient iterations it takes in order to reach the specified tolerance, $\lambda_{min}(\underline{A})$ and $\lambda_{max}(\underline{A})$ denote the minimum and maximum eigenvalue of $\underline{A}$, respectively, and

$$\kappa = \kappa(\underline{A}) = \lambda_{max}(\underline{A})/\lambda_{min}(\underline{A}) \tag{76}$$

is the condition number of $\underline{A}$. We also compute the error between the analytical solution and the numerical solution; the semi-norm $\mid u - u_h \mid_{H^1}$ is computed using two-point Gauss Legendre quadrature on each element, while the discrete semi-norm error is $\parallel \underline{\hat{u}} - \underline{u} \parallel_A = ((\underline{\hat{u}} - \underline{u})^T \underline{A}\,(\underline{\hat{u}} - \underline{u}))^{1/2}$; here $\underline{\hat{u}}$ represents the exact solution u at the interior finite element nodes (the interpolant).

We now comment on the numerical results in Table 1. First, we notice that $m = N$, i.e., the number of iterations is equal to the number of unknowns if we want to reduce the residual down to (essentially) machine precision. The good news is that the conjugate gradient iteration lives up to its promise of not requiring more than N iterations (it just made it; in fact, the residual drops several orders of magnitude in the very last iteration). This demonstrates the fact that, in "exact" precision, the conjugate gradient iteration can be regarded as a direct method (it terminates after a predictable, finite number of operations). The bad news is that we were hoping for $m \ll N$ $(\equiv N_1)$.

We also see that the condition number, $\kappa(\underline{A})$, is large, and that it grows like

$$\kappa \sim O(h^{-2})\,. \tag{77}$$

Reducing the node spacing h with a factor of two increases $\kappa(\underline{A})$ with a factor of four.

The discretization error, as measured in the semi-norm, scales as

$$\mid u - u_h \mid_{H^1} \sim O(h)\,, \tag{78}$$

a result which is consistent with the fact that we are using linear finite elements.

4 Multi-dimensional linear finite elements

Fortunately, the situation illustrated in the one-dimensional case is not typical, in particular, for multi-dimensional problems. For example, a tensor-product extension of the one-dimensional problem to d space dimensions would give $N = N_1^d$ while the condition number, $\kappa(\underline{A})$, would only increase with a constant factor $O(d)$. For example, if $d = 3$ and $N_1 = 100$, we have $N = 10^6$, while we expect $\kappa(\underline{A}) \sim 10^4$ (which is only a factor of about 3 larger than for the $N = N_1 = 100$ case). In addition, we generally do not need to decrease the residual to machine precision unless we expect the discretization error to also be very small, or unless there is some other essential reason for solving the system to a very high precision. Last, we do not use any form of preconditioning of the original system $\underline{A}\,\underline{u} = \underline{F}$.

Let us now estimate the *total* cost W_{CG} of using the conjugate gradient algorithm to solve the Poisson problem in R^d, $d = 1, 2, 3$. We assume that we use a linear finite element mesh (i.e., $p = 1$) with $N = N_1^d$ degrees-of-freedom, N_1 in each spatial direction. The number of non-zero entries per row in $\underline{A}$ is $O(1)$. Hence, the cost per iteration scales as $O(N)$. The number of iterations, m, scales as $m \sim \sqrt{\kappa} \sim O(h^{-1}) \sim O(N_1)$. Hence, the total cost scales as $W_{CG} = m \cdot O(N) \sim O(N_1) \cdot O(N_1^d)$, i.e.,

$$W_{CG} = O(N_1^{d+1}) = O(N_1 \cdot N)\,. \tag{79}$$

For example, in R^1, we obtain $W_{CG} = O(N_1^2) = O(N^2)$, which is consistent with the earlier one-dimensional example: each iteration requires $O(N)$ operations, and we needed $O(N)$ iterations.

Let us now compare the estimate (79) with the corresponding cost when using a direct, banded solver. If we assume that we use a natural ordering of the unknowns, the bandwidth b will be $b = N_1^{d-1}$ in R^d. The cost of factoring the banded matrix will dominate the cost of back substitution, and we get the estimate

$$W_{LU} = O(b^2 \cdot N) = O((N_1^{d-1})^2 \cdot N)\,. \tag{80}$$

For example, in R^1, $W_{LU} = O(N)$, which is much better than the $O(N^2)$ result obtained for the conjugate gradient method. However, as we increase the number of space dimensions, d, the conjugate gradient method becomes more and more attractive, something we can clearly see from Table 2 below. In addition to a more favorable cost estimate, the conjugate gradient method also scales very well in terms of memory requirement.

Table 2: Comparison of estimated cost for the conjugate gradient method and banded LU-decomposition in d space dimensions.

d	W_{CG}	W_{LU}
1	$N_1 \cdot N$	N
2	$N_1 \cdot N$	$N_1^2 \cdot N$
3	$N_1 \cdot N$	$N_1^4 \cdot N$

5 Exercises

1. If $\underline{A}$ is SPD, show that $\underline{A}^{-1}$ is also SPD and can be used to define the norm: $\forall \underline{y} \in R^N$, $\| \underline{y} \|_{A^{-1}} = (\underline{y}^T \underline{A}^{-1} \underline{y})^{1/2}$.

2. From the derivation of the conjugate gradient algorithm, we know that, at each iteration, the method minimizes the *error* in the A-norm over all elements in $K^n(\underline{A}; \underline{r}_0)$. Show that, at each iteration, the conjugate gradient method also minimizes the *residual* in the A^{-1} norm. Hint: Express the error (35) in terms of the residual.

3. Each update of the solution in the conjugate gradient algorithm can be expressed as $\underline{u}_n = \underline{u}_{n-1} + \alpha_n \underline{p}_n$ where $\alpha_n = \underline{r}_{n-1}^T \underline{r}_{n-1} / \underline{p}_n^T \underline{A} \underline{p}_n$. Show that α_n is optimal in terms of minimizing the functional $J^N(\underline{v})$ along the search direction $\underline{p}_n$ (line minimization). Hint: Note that (51) implies that $\underline{p}_n^T \underline{r}_{n-1} = \gamma_n = \rho_{n-1} = \underline{r}_{n-1}^T \underline{r}_{n-1}$.

4. Demonstrate that the length of the search directions $\{\underline{p}_i\}$ can be chosen arbitrarily. Hint: How does the coefficients $\{a_i\}$ depend upon the length of the search directions?

5. Assume that the initial residual can be expressed as a sum of m eigenvectors with $m \leq N$, i.e., $\underline{r}_0 = \sum_i c_i \underline{q}_i$ where $\underline{A} \underline{q}_i = \lambda_i \underline{q}_i$. Show that the conjugate gradient algorithm converges in m iterations.

6. Assume that $\underline{A}$ is an $N \times N$ matrix which is SPD and has m distinct eigenvalues ($m < N$). Show that the conjugate gradient algorithm converges in at most m iterations.

7. Prove the optimality (or projection) condition (35).

8. Prove (50).

9. Prove (59).

10. What happens if $\underline{r}_0$ is in the direction of one of the eigenvectors of $\underline{A}$?

11. Answer the questions in (b), (c) and (d) of Section 2.5.

12. We know that, in one space dimension and with exact quadrature of the bilinear and linear forms, the finite element solution u_h is equal to the exact solution u at the finite element nodes (i.e., equal to the interpolant of u). Explain the numerical results in the last column of Table 1.

13. In the one-dimensional finite element example, does it make sense to iterate until the residual $(\underline{r}^T \underline{r})^{1/2} < 10^{-10}$ given the numerical results for $| u - u_h |_{H^1}$ in Table 1?

14. Repeat the comparison in Table 2, but now compare the memory requirement in d space dimensions.