Polynomial preconditions for the CG method on the CM2

Pascal Joly, Olivier Perlot

Computing methods in Applied Sciences and Engineering (Proceedings of the 9th International Conference on Numerical Methods in Applied Sciences and Engineering, Paris 1990) (1990), 55–120, SIAM, Philadelphia.

[7] P. Joly, *Résolution de l'équation de Burgers par paquets d'ondelettes*, Publications du Laboratoire d'Analyse Numérique (1995) Université Pierre et Marie Curie, (to appear).

[8] P. Joly, Y. Maday and V. Perrier, *Towards a method for solving partial differential equations by using wavelet packet bases*, Comput. Methods Appl. Mech. Engrg. **116** no. 1–4 (1994), 301–307.

[9] P. G. Lemarié, *Ondelettes à localisation exponentielle*, J. Math. Pures Appl. (9) **67** no. 3 (1988), 227–236.

[10] J. Liandrat, V. Perrier and P. Tchamitchian, *Numerical resolution of Nonlinear Partial Differential Equations using the Wavelet approach.*, Wavelets and their Applications (Ruskai et al., eds.), Jones and Barlett, 1992, pp. 227–238.

[11] S. G. Mallat, *Multiresolution approximation and wavelet orthonormal bases of* $L^2(\mathbb{R})$, Trans. Amer. Math. Soc. **315** no. 1 (1989), 69–87.

[12] Y. Meyer, *Ondelettes et opérateurs*, vol. tomes I à III Hermann, Paris, 1990.

[13] V. Perrier and C. Basdevant, *La décomposition en ondelettes périodiques, un outil pour l'analyse de champs inhomogènes. Théorie et algorithmes*, La Recherche Aérospatiale **3** (1989), 53–67.

[14] J. R. Williams, K. Amaratunga, *Introduction to wavelets in engineering*, Internat. J. Numer. Methods Engrg. **37** no. 14 (1994), 2365–2388.

PASCAL JOLY
UNIVERSITÉ PARIS IV
LABORATOIRE D'ANALYSE NUMÉRIQUE
4 PLACE JUSSIEU
(75252) PARIS (05)
FRANCE
e-mail: joly@ann.jussieu.fr

Memorias III Escuela de Verano
Geometría Diferencial
Ecuaciones Diferenciales Parciales
y Análisis Numérico
Universidad de los Andes, 20 a 30 de junio de 1995
páginas 83–90

Polynomial preconditions for the CG method on the CM2

PASCAL JOLY & OLIVIER PERLOT

Université Paris 6, France

§1 Introduction

This paper is concerned with the implementation of parallel iterative methods for solving large sparse symmetric positive definite linear systems arising from finite difference methods, $Ax = b$, on an SIMD computer, the CM-2. Specifically, we are interested in the Conjugate Gradient (CG) method introduced by Hestenes and Stiefel [8]. This method is a popular and effective linear systems solver, notably when combined with a preconditioner; one attractive possibility considered by many authors ([1], [6], [9] and [13]), is polynomial preconditioning. Its main advantage is its suitability for vector and/or parallel computers, when the matrix by vector product is parallelizable. Whenever A has a regular sparsity structure (multidiagonal), polynomial preconditioning is effective on parallel machines.

The outline of this paper is as follows: Section 2 describes the linear systems we are solving, Section 3 presents the main characteristics of the CM-2 computer, Section 4 is devoted to polynomial preconditioners and in Section 5, we present some numerical results.

§2 Description of model problems

We consider linear systems of equations that arise from the approximation of the solution of the problem:

$$\begin{aligned} -\mathrm{div}(\lambda \nabla u) &= f, \qquad \text{in } \Omega =]0,1[\times]0,1[, \\ u|_{\partial\Omega} &= 0. \end{aligned}$$

The first problem is the Poisson problem, $\lambda \equiv 1$.

The second problem is

$$\begin{cases} \lambda = 1000 & \text{in } \Omega =]\frac{1}{4},\frac{3}{4}[\times]0,1[, \\ \lambda = 1 & \text{elsewhere.} \end{cases}$$

For both problems we use a standard five-point scheme on an uniform mesh ($h = 1/(n+1)$), and thus we obtain a linear system

$$Ax = b \tag{1}$$

The matrix A is a symmetric positive definite matrix of order n^2, and is a pentadiagonal matrix with a natural ordering of the unknowns. If the matrix is stored by diagonals, then the matrix by vector product can be performed in parallel. To solve (1) we use the Preconditioned Conjugate Gradient (PCG) method of Concus, Golub and O'Leary [4].

§3 The Connection-Machine

The Connection Machine is a massively parallel local memory SIMD (Single Instruction stream Multiple Data stream) supercomputer. In fact, all processing elements (PEs, single bit processor) can execute immediately the same instructions on local different data. If they need other data from other PEs, they use a communication network (a hypercube network links sets of processors). But these communications are not very fast. Moreover the CM-2 being a synchronous supercomputer, the PEs must execute the same instruction (on different data) or nothing at the same time. Then the programming is sequential with vectors and synchronous data transfers between processors. Thus some instructions are well suited to the CM-2; saxpy, multidiagonal matrix vector product, etc. .... Therefore we are trying to use these kernels as frequently as possible to optimize the computing time. Then the CG with Diagonal preconditioner seems well-suited to the CM-2's architecture *a priori*, but all the same we will avoid point recursions like ICCG methods. The CM-2 consists at most of 65536 PEs (2048 Weitek chips), and it operates under the control of a host computer (a SUN workstation in general), and the peak Mflops rate with 65536 processors is about 30 Gflops.

Actually all tests have been performed in single precision on 8192 PEs of the CM-2 with the CM-FORTRAN language, and without any CM Scientific Software Libraries.

§4 Polynomial preconditioners

The principle of polynomial preconditioning consists in finding a preconditioner M so that:

$$M^{-1}A = P_k(A)A$$

where P_k is a polynomial, of degree less or equal to k (we denote the polynomials with some capital letter when they are applied to matrices, and with the corresponding lower case letter when they are applied to scalars). This idea may seem, at the same time, natural and strange. Natural, because from the Cayley-Hamilton theorem, the inverse A^{-1} can be expressed as a polynomial in A, but strange because CG generates an optimal polynomial (see [8]) and, in that respect, the CG polynomial after $m(k+1)$ iterations with no preconditioner is better than the polynomial generated by m iterations with a polynomial preconditioner of degree k.

In this section we review several choices for p_k. $P_k(A)$ should be in some sense an approximation of A^{-1}. Of course there are several ways of doing this. As proved by Ashby et al. in [2], there is no single best polynomial, the choice depends on the eigenvalue distribution of A, which is seldom known *a priori*.

One of the most important practical issues for using polynomial preconditioners is the estimation of smallest and largest eigenvalues of A (denoted a and b). For the model Poisson problem, we take its well-known eigenvalues $a = 8\sin^2(\pi/2(n+1))$ and $b = 8\sin^2(\pi n/2(n+1))$, which are not in fact the optimum values (see [2] and [13]). However, these eigenvalues are unknown in realistic applications. Then for solving the second problem with the more elaborate polynomial preconditioners, we propose to scale symmetrically A to have a unit diagonal. Instead of solving (1) we solve

$$D^{-1/2}AD^{-1/2}y = \widetilde{A}y = D^{-1}b, \qquad \text{and} \qquad x = D^{-1/2}y. \tag{2}$$

We know that the eigenvalues of $\widetilde{A}$ are in the set]0,2[, thus to solve (2) we choose $b = 2$. After an important numerical study on the behaviour of polynomial preconditioners in relation to the estimation of a, we choose $a = 0.25\ 10^{-3}$ for this problem. This value is not for this problem the optimum smallest eigenvalue, but it is sufficiently representative of the polynomial preconditioners' numerical effectiveness.

4.1 Diagonal preconditioner.

The simplest approximation M of A we consider is the diagonal matrix

$$M = \mathrm{diag}(A).$$

4.2 Truncated Neumann's series.

The second approximation, proposed by Dubois, Greenbaum and Rodrigue in [5], is based on Neumann's series. We denote $A = D - L - L^T$ by $D = \mathrm{diag}(A)$, with L strictly lower triangular. If A is a positive definite matrix, then

$$A^{-1} = D^{-1/2}\left(I - D^{-1/2}(L+L^T)D^{-1/2}\right)^{-1}D^{-1/2}.$$

Moreover, if A is diagonally dominant then the spectral radius

$$\rho\left(D^{-1/2}(L+L^T)D^{-1/2}\right) < 1$$

and the Neumann's series for the inverse of $I - D^{-1/2}(L+L^T)D^{-1/2}$ converges. We denote by NEUM1 the following preconditioner:

$$M^{-1} = D^{-1} + D^{-1}(L+L^T)D^{-1},$$

and by NEUM3:

$$\begin{aligned} M^{-1} = {} & D^{-1} + D^{-1}(L+L^T)D^{-1} + D^{-1}(L+L^T)D^{-1}(L+L^T)D^{-1} \\ & + D^{-1}(L+L^T)D^{-1}(L+L^T)D^{-1}(L+L^T)D^{-1}. \end{aligned}$$

According to Ashby [1], NEUM2 is worse than NEUM1.

4.3 Polynomials minimizing a generalization of the condition number.

The previous preconditioner is very easy to use, as there are no parameters to estimate. Unfortunately it may yield a poor preconditioner in relation to the convergence rate (see numerical results). Therefore we have to consider more elaborate polynomial preconditioners. In this section, we are interested in polynomials minimizing an upper bound of the condition number; the minmax polynomial.

Let $[a,b]$ be a set containing the eigenvalues of A, $0 \notin [a,b]$, and let

$$\mathcal{Q}_k = \left\{\text{polynomials } q_k \text{ of degree } \leq k \mid q_k(\lambda) > 0 \ \forall \lambda \in [a,b], \text{ and } q_k(0) = 0\right\}$$

We are looking for $q \in \mathcal{Q}_{k+1}$ minimizing:

$$\frac{\max_{\lambda\in[a,b]} q(\lambda)}{\min_{\lambda\in[a,b]} q(\lambda)}.$$

Let $\mu(\lambda) = 2\dfrac{\lambda - b - a}{b-a}$ be the linear mapping taking $[a,b]$ into $[-1,+1]$. Then the solution is given, for example by Johnson, Michelli and Paul in [9], as:

$$q_{k+1}(\lambda) = 1 - \frac{T_{k+1}(\mu(\lambda))}{T_{k+1}(\mu(0))},$$

T_k being the Chebyshev polynomial of the first kind of order k. Then we obtain p_k by:

$$p_k(\lambda) = \frac{1}{\lambda}\left(1 - \frac{T_{k+1}(\mu(\lambda))}{T_{k+1}(\mu(0))}\right).$$

Now, to evaluate $z = P_k(A)r = \sum_{i=0}^{k} \alpha_i A^i r$, we want to conserve both the sparsity of A and also the effective matrix by vector product. Moreover, the computation of α_i should stay simple and fast. Thus we use first a simple Hörner's scheme, which is given as follows:

$$\begin{cases} z_k = \alpha_k r, \\ z_i = \alpha_i r + A z_{i+1}, & i = k-1, \ldots, 0 \end{cases}$$

Then $z = z_0 = P_k(A)r$.

4.4 Least squares polynomials.

As an alternative choice, we consider polynomials minimizing some quadratic norm of the residual polynomial [9]

$$\int_a^b (1 - \lambda p(\lambda))^2 \omega(\lambda)\, d\lambda,$$

where $\omega(\lambda)$ is a weight function able to emphasize the portion of the spectrum which is the most important. In practice, we take a Jacobi weight function:

$$\omega(\alpha,\beta,\lambda) = (b-\lambda)^\alpha(\lambda - a)^\beta.$$

We denote by $s_i(\lambda)$ the associated orthonormal polynomials. The solution of this problem is:

$$p_k(\lambda) = \sum_{j=0}^{k+1} b_j t_j(\lambda),$$

with

$$b_j = \frac{s_j(0)}{\sum_{i=0}^{k+1} s_i^2(0)} \quad \text{and} \quad t_j(\lambda) = \frac{s_j(0) - s_j(\lambda)}{\lambda}.$$

Now we would like to compute $p_k(\lambda) = \sum_{j=0}^{k+1} b_j t_j(\lambda)$. The s_i are orthonormal with respect to ω. Then they satisfy a three-term recursion:

$$s_{j+1}(\lambda) = \alpha_j \nu(\lambda) s_j(\lambda) - \gamma_{j-1} s_{j-1}(\lambda), \qquad \text{for } j \geq 1,$$

with

$$\frac{\nu(0) - \nu(\lambda)}{\lambda} = C,$$

C being a constant.

Specifically, we have studied two kinds of Jacobi polynomials; the Chebyshev polynomials ($\alpha = \beta = -\frac{1}{2}$), and the Legendre polynomials ($\alpha = \beta = 0$).

§5 Numerical results on CM-2

In this section, we give results of numerical experiments in order to point out some additional facts about polynomial preconditioning on massively parallel computers. The PCG algorithms stopped as soon as the approximate residual $r_j = b - Ax_j = r_{j-1} - \alpha_{i-1}Ap_{i-1}$, satisfied $\|r_j\| \leq \epsilon\|r_0\|$, where $\epsilon = 10^{-6}$. We study the number of iterations, the Mflops rate and the computing time as a function of the polynomial degree. At first we test the model Poisson problem with a 512×512 mesh.

The first results concern the number of iterations of the PCG with these preconditioners (in fact, table 1 shows the degree which minimizes the computing time for all preconditioners).

Preconditioner	Optimal degree	Number of iterations	Total time (s)	Mflops
DIAG	0	1521	70.5	124.5
NEUM1	1	761	54.1	114.4
NEUM3	3	537	58.1	119.0
MINMAX	12	109	36.8	164.4
MCARRE	16	100	45.4	171.6
LEG	10	163	49.6	174.4

TABLE 1:
CM-2'S OPTIMAL RESULTS ON PROBLEM 1 WITH A 512×512 MESH.

The convergence rate of the polynomial preconditioners is always better than those of the diagonal preconditioner, but does depend on the choice of the polynomial. For example, the decrease in the number of iterations using truncated Neumann's series is not sufficient to allow degrees greater than 3, because of the cost of the polynomial computation. The three-term recursion evaluations are very stable and actually competitive. In fact, we can even use much higher degrees (500) with the algorithms MINMAX, MCARRE and LEG.

It has sometimes been claimed that the diagonal preconditioner can deliver the smallest computing times on massively parallel supercomputers. Table 1 proves that the polynomial preconditioners are better massively parallel preconditioners for this problem; with the optimum degree of MINMAX (12), the time required to solve the problem with the diagonal preconditioner has been nearly cut in half.

Some algorithms have a better MFlops rate than others, because at each iteration they compute more operations without communication between processors. The polynomial preconditioners are very massively parallel, because they have a very high parallelism degree, 89.5% with MINMAX and its optimum degree (12), or 98.5% with a higher degree (100).

Now, we are interested in solving an ill-conditioned linear system; the discrete version of the second problem with a 256×256 mesh. The eigenvalues of the matrix A are not explicitly known for this problem, so we have symmetrically scaled the matrix A in order to have a unit diagonal, and then we solve (2) instead of (1).

Preconditioner	Optimal degree	Number of iterations	Total time (s)	Mflops
DIAG	0	7368	129.0	82.3
NEUM1	1	3792	108.5	71.0
NEUM3	3	3238	126.3	82.4
MINMAX	31	223	52.9	142.7
MCARRE	70	108	56.4	153.4
NORM	90	87	56.6	147.4

TABLE 2:
CM-2'S OPTIMAL RESULTS ON PROBLEM 2 WITH A 256×256 MESH.

Table 2 confirms that the diagonal preconditioner is no longer the best massively parallel preconditioner for these kinds of problems. The CM-2 achieves fewer Mflops than for problem 1, only because the mesh is smaller. Nevertheless, we keep a good degree of parallelism, 93% with MCARRE and polynomial degree 100.

Finally, MINMAX, MCARRE and NORM are some of the very efficient preconditioners on massively parallel supercomputers like the CM-2.

§6 Conclusions

We have reviewed some techniques for an efficient use of PCG on massively parallel supercomputers. The main conclusion of this paper is that the polynomial preconditioners associated with a fast matrix by vector product can run much faster than the simpler but more easily parallelizable diagonal preconditioner.

Thus, for an efficient use of PCG on massively parallel supercomputers, we propose to use the MINMAX or NORM preconditioners. MINMAX gives the better results of these preconditioners, but it needs an estimation of the minimum eigenvalue $a \neq 0$, whereas NORM is independent of the eigenvalues.

The problems we propose may seem academic, but the method can handle more realistic ones, on grids topologically equivalent to 2D-rectangular grids; and it can also be used as a linear system solver in a fictitious domain approach for more general meshes.

References

[1] S. F. Ashby, *Minimax polynomial preconditioning for hermitian linear systems*, SIAM J. Matrix Anal. Appl. **12** no. 4 (1991), 766–789.

[2] S. F. Ashby, T. A. Manteufell and J. S. Otto, *A comparison of adaptive Chebyshev and least squares polynomial preconditioning for Hermitian positive definite linear systems*, SIAM J. Sci. Statist. Comput. **13** no. 1 (1992), 1–29.

[3] P. Ciarlet Jr., G. Meurant and O. J. Perlot, *Préconditionnements massivement parallèles*, Journées du SEH (1992).

[4] P. Concus, G. H. Golub and D. P. O'Leary, *A generalized conjugate gradient method for the numerical solution of elliptic partial differential equations*, in "Sparse Matrix Computations" (Proc. Sympos., Argonne Nat. Lib., Lemont, Ill., 1975) (J. R. Bunch and D. J. Rose, eds.); Academic Press, New York, 1976, pp. 309–332.

[5] P. F. Dubois, A. Greenbaum and G. H. Rodrigue, *Approximating the inverse of a matrix for use in iterative algorithms on vector processors*, Computing **22** no. 3 (1979), 257–268.

[6] R. W. Freund, *Polynomial preconditioners for Hermitian and certain Nonhermitian Matrices*, presented at SIAM Annual Meeting, San Diego, CA, July 1989.

[7] R. W. Freund and B. Fischer, *On adaptive weighted polynomial preconditioning for Hermitian positive definite matrices*, in "Proceedings of the Copper Mountain Conference on Iterative Methods in Numerical Linear Algebra, Copper Mountain, April 1992" SIAM J. Sci. Comput., vol. 15, pp. 408–426.

[8] M. R. Hestenes and E. Stiefel, *Methods of conjugate gradients for solving linear systems*, J. Research Nat. Bur. Standards **49** (1952), 409–435.

[9] O. G. Johnson, C. A. Michelli and G. Paul, *Polynomial preconditioners for conjugate gradient calculations*, SIAM J. Numer. Anal. **20** no. 2 (1983), 362–376.

[10] J. A. Meijerink and H. van der Vorst, *An iterative solution method for linear systems of which the coefficient matrix is a symmetric M-matrix*, Math. Comp. **31** no. 137 (1977), 148–162.

[11] G. Meurant, *Practical use of the C.G.M on parallel supercomputers*, Comput. Phys. Comm. **53** (1989), 467–477.

[12] S. G. Petiton and C. J. Weill-Duflos, *Very sparse preconditioned conjugate gradient on massively parallel architectures*, SEH publications, 1992.

[13] Y. Saad, *Practical use of polynomial preconditionings for the conjugate gradient method*, SIAM J. Sci. Statist. Comput. **6** no. 4 (1985), 865–881.

[14] C. Tong, *The preconditioned conjugate gradient method on the Connection Machine*, Proceedings of the Conference on Scientific Applications of the Connection Machine (Simon (Horst D), ed.), World Scientific.

PASCAL JOLY
UNIVERSITÉ PARIS IV
LABORATOIRE D'ANALYSE NUMÉRIQUE
4 PLACE JUSSIEU
(75252) PARIS (05)
FRANCE
e-mail: joly@ann.jussieu.fr

OLIVIER PERLOT
UNIVERSITÉ PARIS IV
LABORATOIRE D'ANALYSE NUMÉRIQUE
4 PLACE JUSSIEU
(75252) PARIS (05)
FRANCE

Memorias III Escuela de Verano
Geometría Diferencial
Ecuaciones Diferenciales Parciales
y Análisis Numérico
Universidad de los Andes, 20 a 30 de junio de 1995
páginas 91–96

Pointwise error estimates and asymptotic error expansion inequalities for the finite element method on irregular grids

ALFRED H. SCHATZ[1]

Cornell University, USA

The aim of this lecture is to derive new pointwise error estimates for the finite element method on general quasi–uniform meshes for second order elliptic boundary value problems in $\mathbb{R}^N$, $N \geq 2$. In a sense to be discussed below, these estimates represent an improvement on the now standard quasi–optimal L_∞ estimates. In order to fix the ideas, here we will deal with global estimates for a model Neumann problem with smooth solutions. Local estimates, both interior and up to the boundary, which are applicable to a variety of problems with both smooth and nonsmooth solutions can also be derived. As a consequence of these estimates, some new and useful inequalities will be given which are in the form of error expansions. They are valid for large classes of finite elements on general quasi–uniform meshes in $\mathbb{R}^N$ and have application to both superconvergence and extrapolation. Let us begin by giving a brief description of some of the main results.

Let Ω be a bounded domain in $\mathbb{R}^N$, $N \geq 2$ with smooth boundary $\partial\Omega$. Let

$$A(u,v) = \int_\Omega \Big(\sum_{i,j=1}^N a_{ij}(x) \frac{\partial u}{\partial x_i} \frac{\partial v}{\partial x_j} + \sum_{i=1}^N b_i(x) \frac{\partial u}{\partial x_i} v + c(x) uv \Big) dx \quad (1)$$

[1] Supported in part by the National Science Foundation Grant DMS 9403512