Vol. 7 No. 2 Journal of Computational Mathematics April 1989

LARGE MATRIX COMPUTATIONS ON VECTOR COMPUTERS*

YAU SHU WONG
(*University of Alberta, Edmonton, Canada*)

Abstract

Preconditionings have proved to be a powerful technique for accelerating the rate of convergence of an iterative method. This paper, which is concerned with the conjugate gradient algorithm for large matrix computations, investigates an approximate polynomial preconditioning strategy. The method is particularly attractive for implementation on vector computers.

1. Introduction

In recent years, remarkable progress has been reported on large scale numerical simulations for many practical problems in science and engineering. The rapid advances in the field of computer hardware, in particular, the advent of supercomputer technology, have played a significant role in this development. It is important, however, to note that in order to exploit the full potential of the vector or parallel processors available on supercomputers, numerical algorithms must be developed to take advantage of the specific computer architecture. Many standard numerical algorithms, which have been very successful on the conventional scalar computers, could become inefficient when implemented on the supercomputer.

The purpose of this paper is to study efficient numerical algorithms for large matrix computations on vector processors. It is important to develop very efficient numerical algorithms, because such a problem frequently results from a numerical solution to partial differential equations.

---

* Based on the lecture presented at the China-U.S.A. seminar held at the Xián Jiaotong University, December 1987. This work was supported by the Natural Sciences and Engineering Research Council of Canada Grant U0375.

2. Conjugate Gradient Algorithm

Consider the linear system

$$Au = b, \tag{1}$$

where A is a given symmetric and positive definite matrix. Now, introducing a non-singular matrix M, Equation (1) can be rewritten as

$$AM^{-1}Mu = b. \tag{2}$$

Equation (2) is known as the preconditioned system, and M is the preconditioning matrix.

It has been widely accepted that the method of conjugate gradient (CG) [3] is an efficient iterative technique for large matrix calculations. The CG algorithm for solving Equation (2) can be summarized as follows:

Initialization. Start with an approximation to the solution vector u^0, compute the residual $r^0 = b - Au^0$, and set the direction vector $p^0 = M^{-1}r^0$.

Iteration. For $n = 0, 1, 2, \cdots$, do:

Step 1. $\alpha_n = (r^n, M^{-1}r^n)/(p^n, Ap^n)$.

Step 2. $u^{n+1} = u^n + \alpha_n p^n$.

Stpe 3. $r^{n+1} = r^n - \alpha_n Ap^n$.

Stpe 4. $\beta_n = (r^{n+1}, M^{-1}r^{n+1})/(r^n, M^{-1}r^n)$.

Stpe 5. $p^{n+1} = M^{-1}r^{n+1} + \beta_n p^n$.

The process is continued until $\|r^{n+1}\|$ satisfied a convergence criterion. The inner product (x, y) is defined as $x^T y$ for any vectors x and y.

The CG algorithm presented here can be efficiently implemented on a vector computer. For the $CDC\ CYBER$ 205 computer with 2 pipes and 64-bit arithmetic, the maximum computational rate for a linked triad operation (i.e., vector + constant * vector) is 200 million floating-point operations per second (Mflops). It can be easily seen that Step 2, 3 and 5 are the linked triad operations. The two inner products in Steps 1 and 4 can be computed by the Q8SDOT routine available on the CYBER 205 computer, and the maximum rate is 100 Mflops. The major computational work for each CG iteration consists of the matrix by vector multiplication Ap and the preconditioning step $M^{-1}r$. For large and sparse matrices, the non-zero coefficient elements of A can be stored by diagonals. The computation for Ap can then be efficiently implemented almost entirely using the linked triad operations [6].

The computational work for each CG iteration can be expressed as

$$W = W_b + W_p, \tag{3}$$

where W_b is the work for the basic CG algorithm and

$$W_b = 2 * IP + 3 * LT + 1 * MV. \tag{4}$$