Gaussian Elimination on the Cray Y-MP8/64

Cliff Cyphers and Marcin Paprzycki
Department of Mathematics and Computer Science
The University of Texas at the Permian Basin

This note examines the solution of dense systems of linear equations on a Cray Y-MP8/64 computer. Our experiments were performed for the most efficient version of Gaussian elimination with partial pivoting. To decompose the system we employed the JKPVT algorithm [1], which is based on column elimination, column pivoting, and row interchanges. In the most expensive step the j-th column of the matrix is updated using the j-1 previously calculated columns. The back substitution step was performed by updating the right-hand side using columns of the decomposed matrix. Both methods are efficient as far as the number of vector loads and stores is concerned and take full advantage of chaining (see [1] for more details).

We compared three different implementations of Gaussian elimination. The first was coded in Cray Fortran (cif77 compiler with optimization). The second used level 1 BLAS [4] and the third used level 2 BLAS [2]. We replaced all vector-vector operations in the decomposition and back substitution steps by calls to BLAS 1 routines _SCAL, _SWAP, _amax, _axpy and _dot (see [4] for details). We then modified our original code to utilize level 2 BLAS routines _ger (decomposition) and _trsv (back substitution). The results presented here may be related to Sewell [3].

All experiments were performed for N = 100, 200, 400, 600, 800, 1000, and 1200. The coefficient matrix was created by Cray's random number generator. The Euclidean norm of error varied from 1.0E-11 for N = 100 to 1.0E-9 for N = 1200, which is what one would expect for Cray’s single precision. Our experiments were executed on one processor and the program performance was estimated by the perftrace utility.

We compared the performance of three different versions of BLAS:

- the (default) Cray Science Libraries (libsci) BLAS
- Boeing VectorPak (vpak) BLAS (use "-l bcslib" option of segldr)
- NAG BLAS (use "-l nag" option)

[Continued on page 44]
Boeing BLAS consistently displayed the best performance, although the differences (measured in MFLOP rate increase) were small. For level 1 BLAS the vpkak averaged 3.9% better than the Cray BLAS and 4.2% better than the NAG BLAS. The biggest advantage was 7.8% over libsci and 6.7% over NAG was at N = 100. The smallest differences were: 1% over libsci and 1.8% over NAG at N = 1000.

For level 2 BLAS the vpkak averaged 2.1% better than the Cray BLAS and 2.6% better than the NAG BLAS. The biggest advantages were: 5% over libsci at N = 200 and 4.8% over NAG at N = 100. The smallest differences were: 1.61% over libsci at N = 600 and 1% over NAG at N = 1000.

The results are presented for the Boeing BLAS. The comparison of the MFLOP rates for decomposition is illustrated in Figure 1 and for back substitution in Figure 2. Figure 3 presents the percentage of MFLOP rate increases of BLAS 1 and BLAS 2 over the original (no-BLAS) code.

We reached up to 85% performance of one processor when level 2 BLAS was used and N was "large" (N > 1000). This is true for both decomposition and back substitution when treated as units. An observable decrease in performance for N = 800 in the level 2 BLAS version of the LU decomposition was caused by memory bank conflicts (see Sewell [3]). We did not, however, observe a similar effect for back substitution.

There is a lot to be gained from using BLAS. This is especially true for level 2 BLAS (see Figure 3), whether it is used for back substitution or decomposition. Even for relatively small systems (N = 100), level 2 BLAS gives more than twice the MFLOP rate of the no-BLAS code. The difference between the advantages for decomposition and back substitution is substantial for small N; however, it almost disappears when N increases.

Level 1 BLAS, on the other hand, is clearly advantageous only for larger systems. Our experiments show that to avoid penalties caused by function calls, one ought to use Level 1 BLAS for N > 250 in decomposition and for N > 150 in back substitution. There is almost no difference between the advantages for decomposition and back substitution (Figure 3).

In summary, although the optimization/vectorization provided by the Cray cft77 Fortran compiler is useful, the level 2 BLAS gives exceptional advantages in performance.


LU Decomposition

Figure 1.
Back Substitution

No BLAS ★★★
BLAS 1 ★★★★★
BLAS 2 ★★★★★★★

PEAK

Figure 2.
PERCENTAGE INCREASE OF BLAS OVER NO BLAS

Back Substitution
BLAS 1
BLAS 2

LU Decomposition

Figure 3.