DATA STRUCTURES TO VECTORIZE CG ALGORITHMS FOR GENERAL SPARSITY PATTERNS

GAIA VALERIA PAOLINI and GIUSEPPE RADICATI DI BROZOLO
IBM European Center for Scientific and Engineering Computing, via Giorgione 159, 00147 Roma, Italy

Abstract.
We describe an implementation of Conjugate Gradient-type iterative algorithms for problems with general sparsity patterns on a vector processor with a hierarchy of memories, such as the IBM 3090/VF. The implementation relies on the wavefront approach to vectorize the solution of the two sparse triangular systems that arise when using ILU type preconditioners. The data structure is the key to an effective implementation of sparse computational kernels on a vector processor. A data structure is a combination of a layout of the matrix coefficients and ordering schemes for the vectors to increase data locality. With the data structure we describe, we achieve comparable performance on both the matrix-vector product and the solution of the sparse triangular systems on a variety of real problems, such as those arising from large scale reservoir simulation or structural analysis.

AMS Classification: 65F10.

1. Introduction.

Consider the linear system of equations $Au = b$, where $A$ is a large $m \times m$, sparse, non-singular matrix, and $b$ is a vector with $m$ components. Conjugate Gradient is an efficient and robust iterative algorithm for solving sparse symmetric linear systems of equations ([9]; [31]). Many generalizations of Conjugate Gradient have been proposed for non-symmetric problems ([30]; [33]; [24]; [23]; [28]). A suitable preconditioner is crucial in obtaining rapid convergence of CG-type methods. Preconditioners based on pointwise Incomplete LU factorizations are often used as they both accelerate convergence rate efficiently and are easy to generate and use.

From an implementation point of view, all iterative algorithms based on generalizations of Conjugate Gradient with ILU-type preconditioners share three basic computational kernels:

- Dense algebra: inner products and vector updates.
- Sparse matrix-vector product.
- Solution of sparse triangular systems.

The implementation of the first kernel is straightforward. The sparse matrix-vector product can be efficiently implemented on vector and parallel proces-
sors if the sparsity patterns have some regularities. We propose a data structure that allows us to vectorize this kernel for very general sparsity patterns.

The vectorization of ILU preconditioners is more challenging, and many alternatives to these preconditioners have been proposed. The unknowns of sparse triangular systems, such as those that arise when using pointwise ILU preconditioners, can be ordered in level sets, or wavefronts, of non-interconnected variables which can be solved concurrently. This implementation is algebraically equivalent, up to round-off, to the standard recursive algorithm. This technique is applicable to any problem that involves the solution of a sparse triangular system within an iteration loop, such as SOR and Gauss-Seidel, or CG-methods with SSOR preconditioning.

Many authors have discussed implementations of CG-type codes on vector processors which take advantage of the concurrency in the solve step of the ILU preconditioners for problems with regularities in the sparsity pattern, such as those arising from the finite difference discretization of an elliptic PDE on a cube ([27]; [2]). Others discuss implementations on fine grain parallel machines, in which independent unknowns are dispatched across the processors, for problems with irregular sparsity patterns, such as those that arise in many large scale engineering problems when using finite element-type methods on irregular grids ([1]; [4]). In this work we show how these results can be extended to vectorize the solution of the sparse triangular systems for problems with irregular sparsity patterns on vector processors with a hierarchy of memories. Our experiments were performed on an IBM 3090E/VF, which exemplifies this class of architectures.

The effectiveness of the vectorized implementation of the two sparse kernels depends on the choice of data structures. The issues are to keep the length of the vector operations as large as possible, and to optimize the memory traffic by keeping the vector elements contiguous, or – if they are involved in 
\textit{gather} operations – as close together in the main memory as possible. We propose a data structure suitable for both kernels and that allows us to deal efficiently with general sparsity patterns. It is based on a layout of the non-zero coefficients, index vectors, and ordering schemes for the vectors.

The data structure we have used to vectorize CG-type codes is likely to be effective also on processors other than the IBM 3090/VF. We expect these techniques to be of interest in iterative algorithms, where the cost of the preprocessing phase – namely rearranging the sparse matrix coefficients in an optimized data structure and identifying and sorting the wavefronts – may be spread across many iterations. If implementations similar to the one which we describe prove effective on other machines, it is hoped that standards for data structures for sparse matrices and for the basic computational kernels will emerge.

In section 2 we analyze the data structure. In section 3 we summarize the main techniques used to vectorize preconditioners, and we describe our vector implementation of the solution of sparse triangular systems for problems with general sparsity patterns. We present performance data on the IBM 3090/VF in section 4, and our conclusions in section 5.