## Title and Subtitle
Fast (Parallel) Algorithms for Spherical Transforms and Many-Body Interactions with Applications in Electrostatics, Image Processing and Chemistry

## Authors
S. Lennart Johnsson

## Performing Organization Name(s) and Address(es)
Department of Computer Sciences
University of Houston
4800 Calhoun Blvd.
Houston, TX 77204-2163

## Funding Numbers
F49620-96-1-0289

## Sponsoring/Monitoring Agency Name(s) and Address(es)
AFOSR
801 N. Randolph Street, Room 732
Arlington, VA 22203-1977

## Distribution Statement
Approved for Public Release.

## Abstract
The main accomplishments during this grant period are in the following areas:
1. Fast parallel algorithms for Legendre and spherical transforms. The accomplishments in this area effectively break down into two subarcs, namely:
   (a) Development of fast algorithms for the discrete Legendre transform (FLT).
   (b) Development of factorizations of the Fourier transform (FFT) and the cosine transform (FCT) adapted for the use in spherical transforms.
2. Fast N body algorithms.
3. Efficient parallel codes for electromagnetic field computations.

## Subject Terms

## Number of Pages
10

## Limitation of Abstract


FAST (PARALLEL) ALGORITHMS FOR
SPHERICAL TRANSFORMS AND MANY-BODY
INTERACTIONS WITH APPLICATIONS IN
ELECTROSTATICS, IMAGE PROCESSING AND
CHEMISTRY

GRANT NO F.49620-96-1-0289

Final Report
June 1996 – July 31, 1999

S. Lennart Johnsson and Dragan Mirković
Department of Computer Science
University of Houston
Johnsson@cs.uh.edu, Mirkovic@cs.uh.edu

February 28, 2001
1 Introduction

It has been well known for many years that function representations in spherical harmonics is very effective in several fields of science and engineering, such as quantum chemistry, global weather and climate simulations, image processing, control system design and astrophysics to mention a few. Spherical harmonics are eigenfunctions to the Laplace operator in spherical coordinates.

A few such applications occur in chemistry and molecular dynamics. These applications also require that many-body interactions be modeled and evaluated effectively. Anderson’s method based on Poisson’s formula is a very competitive technique for large particle systems.

The research during this grant period was focused on the development and implementation of fast parallel algorithms for spherical and related Legendre and Fourier transforms, efficient implementations of N-body algorithms and development of efficient parallel codes for electromagnetic field computations.

2 Accomplishments

The main accomplishments during this grant period are in the following areas:

1. Fast parallel algorithms for Legendre and spherical transforms. The accomplishments in this area effectively break down into two subareas, namely:

   (a) Development of fast algorithms for the discrete Legendre transform (FLT).

   (b) Development of factorizations of the Fourier transform (FFT) and the cosine transform (FCT) adapted for the use in spherical transforms.

2. Fast N-body algorithms.

3. Efficient parallel codes for electromagnetic field computations.
2.1 Fast Legendre Transforms

For the fast Legendre transforms of computational complexity $O(N\log_2^2 N)$ we devised a parallel code and investigated its performance on Connection Machine systems. The code is based on an algorithm devised by Driscoll and Healy with subsequent improvements and variations by Healy, Moore and Rockmore [2, 1]. To our knowledge our code is the first and only parallel code of this new algorithm.

The formal algebraic description of algorithms developed as part of this work is suitable for both performance analysis and program transformation techniques. Such manipulation is an integral part of high-level compilation systems. The results of this work has been documented in two technical reports, two conference publications, two journal submissions and one PhD thesis.

2.2 Fast Fourier Transforms

We have developed an efficient FFT library which contains code that adapts itself to the processor architecture by factoring the Fourier transform at runtime depending upon its size. The library contains a number of composable blocks of code, called modules or codelets, each computing a part of the transform, and initialization/execution routines. Given the parameters of the problem the initialization routine selects the optimal execution strategy in terms of the actual execution time on the given architecture. The modules in the library are highly optimized and generated by a special-purpose compiler that we have developed. The execution routine uses a combination of the mixed-radix splitting and the prime factor algorithm (PFA). The approach is similar to what is used in the FFTW project [3], but differs in optimization and execution strategy.

The overall efficiency of the code depends strongly on the efficiency of the modules, but the actual coding and optimization becomes very tedious and difficult for sizes greater than 5. For this reason many authors have found it convenient to build the modules by using different ways of automatic code generation. Our code generator is written in C and it can produce DFT modules of arbitrary size, direction (forward or inverse), and rotation (for PFA.) It first generates an abstraction of the FFT algorithm by using a combination of Rader's algorithm [6], mixed-radix algorithm and PFA. The next step is the optimization and scheduling of operations. This optimization
is also architecture dependent and it is performed only once, during the installation of the library. Finally, the abstract code is unparsed to produce the desired C code. An important feature of the code generator that we have developed is its generality, hence it can be easily extended to problems other than generation of FFT modules, such as generation of modules for FLT's and FCT's.

The initialization of the code consists of the fast computation of twiddle factors and other constants needed for the transform, and of selecting the optimal factorization of the Fourier transform of a given size. This approach makes the code adaptive to the architecture it is running on. First, we use a combination of the mixed-radix [5] and the prime factor algorithm (PFA) [5] splittings to generate a large number of possible factorizations for a given transform size. Next, we select the fastest factorization in terms of the actual execution time on the given architecture.

We have developed recursive execution routines which are based on the combination of the mixed-radix and the prime factor algorithms.

The results of this work have been documented in one conference publication and two MS theses.

2.3 Fast N–body Algorithms

One major accomplishment is the data parallel partitioning technique implemented in High Performance Fortran (HPF). This software is viewed as a major accomplishment in that it is to our knowledge the first implementation of a partitioning routine for irregular data structures in the data parallel language HPF, and the implementation is efficient.

Our implementation of an adaptive N-body code for three-dimensional problems is also viewed as a breakthrough in that it again achieves high efficiency for nonuniform computations despite its implementation in HPF. The breakthrough lies in the techniques used for handling the nonuniformity efficiently. The techniques apply to other nonuniform problems as well.

Another major accomplishment is a technique for calling HPF programs from MPI programs. Such calling sequences are not defined in HPF, while the converse is part of the HPF definition. HPF has global data structures, and the information about these data structures can be inherited by MPI programs through an interface known as intrinsic procedures. However, there is no global data structure information in an MPI program, and thus calling an HPF program from an MPI code represents a serious challenge.
Our techniques, communication with UNIX sockets, communication with MPI and communication using shared memory, have been tested and run on a distributed memory machine (IBM SP2), shared memory machines (HP Exemplar, SGI Origin 2000) and the Globus metacomputing toolkit. We believe our techniques will be useful for integrating existing MPI and HPF codes. In addition, MPI-HPF communication will be useful, where a mixture of data and task parallelism is desired. This is because MPI is popularly used for implementing task parallelism and HPF is an effective language for implementing data parallel applications.

With respect to developing an understanding of the error properties of Anderson's method, and comparing it to other similar methods, such as the Greengard–Rokhlin fast multi-pole method (GRFM), we have empirically established that the errors of the two methods is approximately the same for an equal number of terms in the expansions, i.e., the integration order in Anderson’s method needs to be about twice that of the multipole order of GRFM for approximately the same error. To achieve high accuracy the integration order in Anderson’s method needs to be increased beyond the 14th order for which formulas were provided in the reference Anderson used. We initiated an investigation of higher order integration formulas on the sphere that are computationally efficient.

For the fast N–body codes we made simulations determining the error for Anderson’s method as a function of the integration order (and the number of terms in the expansion) for both some synthetic cases as well as for some molecules used in studies at the materials Laboratory at Wright Patterson Air Force Base. The largest molecule had 80,712 atoms. The error was determined as the difference between the potentials computed by Anderson’s method and the direct N–body algorithm requiring \(O(N^2)\) operations. The same cases were studied using the GRFM code from the material’s Laboratory.

To achieve high accuracy the integration order in Anderson’s method needs to be increased beyond the 14th order for which formulas were provided in the reference Anderson used. We have reconstructed numerical integration formulas for the surface of the sphere of orders 9, 11, 13, 15, 17, 19, and 23, using the approach given in [4].

The adaptive code has been ported and tested on IBM SP2 at University of Houston, HP Convex Exemplar X-Class at California Institute of Technology and on Globus (IBM SP2) testbed with PGHPF compiler(PG-Portland Group).
2.4 Efficient parallel codes for electromagnetic field computations

We have developed a Computational Electromagnetics code based on a high-order algorithm due to Joseph Shang [7] for computing the scattering of an electromagnetic wave from geometrically complex objects. Our parallel code is new (not a port of a sequential code) and it is optimized at the algorithmic and at the (parallel) implementation level. The code is arithmetically more efficient than the original sequential code (it uses about 70% as many arithmetic operations as the original code). It uses an automatic domain partitioning that minimizes the communication needs and loop reordering for minimum looping overhead and maximum efficiency in use of local memory hierarchies.

We have performed a performance analysis which includes estimates of the best possible performance with respect to arithmetic capabilities of the processor architectures and operations in the code, as well as the peak performance with respect to memory bandwidth. On IBM SP platforms, a Fortran 90 with MPI implementation achieves a sequential efficiency of about 70% of estimated peak. Communication accounts for less than 10% of the execution time. The performance is about 40% better than the previous best implementation of the same algorithm on SP systems. The code is highly modular and includes a flexible domain decomposition routine that as a default, distributes the domain evenly across all processors in a way that minimizes the necessary communication. We have also developed a distributed latency-tolerant IBM SP implementation, using the vBNS backbone and the Globus toolkit.
3 Personnel Supported

Principal Investigator: S. Lennart Johnsson
Research Associates: Dragan Mirković and Zdenko Tomašić.

4 Publications

4.1 Ph.D. theses


4.2 MS theses


4.3 Conference and journal publications


References


