Theor Chim Acta (1991) 79: 297–311

Theoretica Chimica Acta

© Springer-Verlag 1991

On the role of parallel architecture supercomputers in time-dependent approaches to quantum scattering

David K. Hoffman[1], Omar A. Sharafeddin[2], Donald J. Kouri[2]***, Michael Carter[3], Naresh Nayar[3], and John Gustafson[3]**

[1] Department of Chemistry and Ames Laboratory*, Iowa State University, Ames, Iowa 50011, USA

[2] Department of Chemistry and Department of Physics, University of Houston, Houston, Texas 77204-5641, USA

[3] Applied Mathematical Sciences, Ames Laboratory* and Scalable Computing Facility, Iowa State University, Ames, Iowa 50011, USA

Received October 7, 1990/Accepted November 13, 1990

Summary. Results of our initial study of the use of parallel architecture supercomputers in solving time-dependent quantum scattering equations are reported. The specific equations solved are obtained from the time-dependent Lippmann–Schwinger integral equation by means of a quadrature approximation to the time integral. This leads to a modified Cayley transform algorithm in which the primary computational step is a matrix-vector multiplication. Implementation has been carried out both for the MasPar MP-1 and the NCUBE 6400 parallel machines. The codes are written in a modular form that greatly facilitates porting from one machine architecture to another. Both parallel machines prove to be more powerful for this application than the serial architecture VAX 8650. Specific analysis of machine performance is given.

Key words: Wave packets — Integral equations — Parallel computing

1. Introduction

The emerging impact of vector processing supercomputers on the problem of solving the multidimensional Schrödinger equation describing molecular collisions can hardly be overestimated [1]. Until now such studies have been carried out for the most part with either serial, or minimally parallel, architecture supercomputers. Recent advances in computer development are, however, leading rapidly to the availability of massively parallel computers of several types. Because these new computers will ultimately incorporate many thousands of processing elements (PEs), it is of great interest to study their suitability for carrying out converged quantum scattering calculations. An extremely important aspect of such studies is the exploration of methods which may not necessarily

* Ames Laboratory is operated for the U.S. Department of Energy by Iowa State University under Contract No. 2-7405-ENG-82. This research was supported by the Division of Chemical Sciences and Applied Mathematical Sciences, Office of Basic Energy Sciences

** R.A. Welch Predoctoral Fellow under R.A. Welch Foundation Grant E-608

*** Supported in part under National Science Foundation Grant CHE89-07429 be efficient for serially designed computers, but which may be ideally suited to massively parallel processing.

Of special interest to us are the time-dependent wave packet methods [2–58] and, in particular, a variety of integral equation approaches to solving the time-dependent Schrödinger equation (TDSE) for scattering [48, 50, 51, 58], ultimately including inelastic and rearrangement processes. In particular, we have as a goal the study of systems containing three bodies. Such systems lead to partial differential equations in six spatial variables, plus the time (three additional variables describe the center of mass motion of the atoms and are, hence, exactly separable in the absence of external forces). By considering the TDSE, we take advantage of its *initial value character*, as opposed to the boundary value nature of the time-independent Schrödinger equation. For the latter, simultaneous imposition of all appropriate boundary conditions creates substantial (though not insurmountable) complications [1]. The integral equation form of the TDSE provides the unique solution that describes the collision process in a manner most directly corresponding to experiment. The asymptotic behavior of the wave function is automatically ensured by the time evolution of the packet. Further, scattering information is obtained over the range of energies included in the original initial wave packet; one deals with rigorously quadratically integrable functions, and the integral equation approach automatically and rigorously separates the dynamics into a part governed by a reference Hamiltonian, H_r, and a disturbance Hamiltonian, H_d, (which do not commute with one another). This leads to a great deal of flexibility in the general time-integral equation approach we are exploring [58]. The separation of H_r and H_d implies that one can evaluate the action of each in its diagonal representation, and switch between representations as appropriate [58]. In the case of three body systems, one may further reduce the number of degrees of freedom from seven to four by taking advantage of the isotropy of space. This implies conservation of total angular momentum, and one may exactly separate three angular degrees of freedom (the three Eulerian angles which specify the orientation of the three particle triangle in space). This leads to coupled time-dependent equations containing the three remaining variables which specify the shape and size of the triangle formed by the three bodies [1, 59].

The basic procedure employed to solve the TDSE is independent of whether one deals with simple two body scattering (which is equivalent to scattering of an single "effective" particle of reduced mass μ by an infinitely massive center of force) or a three body system. The essential features of the approach are (a) the calculation of what amounts to a discretized time translation operator or matrix (which evolves the wave packet forward in time), (b) the calculation of the initial vector (the initial wave packet evaluated on a grid of points at time $t = 0$), (c) the multiplication of the state vector by the time translational matrix, (d) the calculation of the scattering amplitudes at the desired energies. The last item can be done in several ways, including projection of the wave packet onto the final states at a time when the packet has left the interaction region [25, 31], or the calculation of the time-to-energy Fourier transformation of the time-dependent amplitude density and projection onto the appropriate final state [50, 55, 56]. The former yields the S-matrix while the latter yields the T-matrix. As we shall see, the dominant computational step is the calculation of the product of the time translation matrix with the solution vector. This is ideally suited to parallel architecture supercomputers.

The time-integral equation approach has a number of attractive features: (a) there is great flexibility in the choice of the reference and disturbance dynamics [58] and in the choice of computational algorithm to be used [48, 50, 51, 58], (b) the approach can readily treat time-dependent interactions, (c) results can be obtained for the scattering at any energy having significant amplitude in the original wave packet [48, 50, 58], (d) the main computational step in all of the algorithms we are considering is forming the product of a matrix and a vector, (e) boundary conditions are automatically satisfied due to the initial value nature of the TDSE [2–58], (f) the approach is naturally suited to treatment using massively parallel computers, (g) the presence of energetically nonaccessible (closed) channels creates no problems with stability.

In this paper, we report the results of our first implementation of the time-dependent integral equation approach to scattering on both moderately and massively parallel computers. The problem chosen for this first study is a standard potential scattering of a particle by a spherically symmetric potential. In this case, the angular degrees of freedom are exactly separable, leaving one with a two-dimensional (time and radial distance) partial differential equation. Even for this simple case, however, one may test the relative efficiencies of parallel architecture computers since the time and distance are discretized, leading to an expression for the wave packet at the new time in terms of the product of an $N \times N$ matrix ($N =$ the number of radial grid points) times an $N \times 1$ vector (the values of the wave packet on the radial grid at the preceding time). Thus, though the problem is a simple one, it will nonetheless enable us to illustrate the potential power of parallel processing computers for scattering.

We note that the present study takes advantage of the Iowa State University/Ames Laboratory Scalable Computation Facility (SCF). The SCF is a β-test site for parallel architecture computers, and, as such, provides an ideal environment for exploring the relative merits of such machines. At present, the SCF has available a MasPar MP-1 and an NCUBE 6400. The MP-1 is a massively parallel machine while the NCUBE is a moderately parallel machine. Other parallel architecture computers will be added when available.

The general strategy was to develop separate "modules" appropriate to the machine architectures for the matrix-vector operations. The relevant modules for the particular parallel computer to be used were inserted into an applications code, which is essentially the same for both computers. This *greatly* facilitates portability of the codes between different machines.

The plan of this paper is as follows. In Sect. 2, we briefly summarize the equations describing a simple scattering system. In Sect. 3, we discuss some properties of and the implementation of the problem on the MP-1. Section 4 contains a similar discussion for the NCUBE 6400. The results are presented and discussed in Sect. 5. Finally, an indication of future work is given in Sect. 6.

2. Time-dependent integral equations for scattering

The time-dependent Schrödinger equation is given by:

$$i\hbar \frac{\partial}{\partial t} \Psi(t) = H\Psi(t) \tag{1}$$

It is easy to verify that if the full Hamiltonian, H, is written as the sum of a "reference dynamics" Hamiltonian, H_r, and a "disturbance" Hamiltonian, H_d, then the *exact* solution of Eq. (1) may be written as [51, 58]:

$$\Psi(t) = -\frac{i}{\hbar}\int_{t-\tau}^{t} dt' \exp[-iH_r(t-t')/\hbar]H_d\Psi(t') + \exp(-H_r\tau/\hbar)\Psi(t-\tau) \quad (2)$$

Here, $\Psi(t)$ is the wave function at the current time t, and $\Psi(t-\tau)$ is the wave function at the previous time $t-\tau$. We note that $\Psi(t)$ depends also on all of the spatial variables needed to specify the positions of the particles in the system. We are exploring a variety of methods for developing algorithms for calculating $\Psi(t)$. For the present study, we simply approximate the integral over t' by the trapezoidal rule, and then solve formally for $\Psi(t)$. The result is [51, 58]:

$$\Psi(t) = \left[1 + \frac{i\tau}{2\hbar}H_d\right]^{-1} \exp(-iH_r\tau/\hbar)\left[1 - \frac{i\tau}{2\hbar}H_d\right]\Psi(t-\tau) \quad (3)$$

In this paper, we shall take as H_r the potential:

$$H_r = V \quad (4)$$

and the kinetic energy, K, as H_d; that is:

$$H_d = K \quad (5)$$

We consider a structureless particle with spherical polar coordinates (R, θ, ϕ), and assume spherical symmetry, so that:

$$V = V(R) \quad (6)$$

where $V(R)$ is taken explicitly to be an attractive exponential interaction of the form, $-\exp(-R)$. The initial wave packet $\Psi(t=0)$ is assumed to be:

$$\Psi(t=0) = Y_{LM}(\theta, \phi)\chi_L(R\,|\,0) \quad (7)$$

corresponding to an initial state with well defined orbital angular momentum:

$$L^2 Y_{LM}(\theta, \phi) = L(L+1)\hbar^2 Y_{LM}(\theta, \phi) \quad (8)$$

with z-component of angular momentum:

$$L_z Y_{LM}(\theta, \phi) = M\hbar Y_{LM}(\theta, \phi) \quad (9)$$

and with an initial radial wave packet, $\chi_L(R\,|\,0)$. Then it is easy to express Eq. (3) as [58]:

$$\chi_L(K\,|\,t) = \left[1 + \frac{i\tau}{4\hbar\mu}K^2\right]^{-1} \frac{2}{\pi}\int_0^\infty dRR^2 j_L(KR)\exp[-iV(R)\tau/\hbar] \times \int_0^\infty dK'K'^2 j_L(K'R)\left[1 - \frac{i\tau}{4\hbar\mu}K'^2\right]\chi_L(K'\,|\,t-\tau) \quad (10)$$

where μ is the particle's mass, $\chi_L(K\,|\,t)$ is the Bessel transform:

$$\chi_L(K\,|\,t) = \frac{2}{\pi}\int_0^\infty dRR^2 j_L(KR)\chi_L(R\,|\,t) \quad (11)$$

and $j_L(KR)$ is a spherical Bessel function. Finally, one discretizes K and R so that Eq. (10) becomes:

$$\chi_L(K_j\,|\,t) = \frac{2}{\pi}\left[1 + \frac{i\tau}{4\hbar\mu}K_j^2\right]^{-1} \sum_{m=1}^{N} \Delta RR_m^2 j_L(K_jR_m)\exp[-iV(R_m)\tau/\hbar] \times \sum_{n=1}^{N} \Delta KK_n^2 j_L(K_nR_m)\left[1 - \frac{i\tau}{4\hbar\mu}K_n^2\right]\chi_L(K_n\,|\,t-\tau) \quad (12)$$

Note that:

$$R_m = m\Delta R \tag{13}$$

$$K_n = n\Delta K \tag{14}$$

and:

$$\chi_L(R\,|\,t) = \int_0^\infty dKK^2 j_L(KR)\chi_L(K\,|\,t) \tag{15}$$

Clearly, these equations involve the product of an $N \times N$ matrix and a vector of dimension $N \times 1$. Note, one also may define the matrix:

$$M(j, n) = \left[1 - \frac{i\tau}{4\hbar\mu}K_n^2\right]\left[1 + \frac{i\tau}{4\hbar\mu}K_j^2\right]^{-1} K_n^2\,\Delta R\,\Delta K\frac{2}{\pi}$$
$$\times \sum_{m=1}^{N} R_m^2 j_L(K_j R_m)\exp[-iV(R_m)\tau/\hbar]\,j_L(K_n R_m) \tag{16}$$

which must be computed *once* at the beginning of the calculation. Then the time evolution simply involves forming the product of the matrix $M(j\,|\,n)$ with the vector $\chi_L(K_n\,|\,t-\tau)$. In the present study we have used the matrix $j_L(K_m R_n)$; however, if the number of time steps is much larger than N, the $M(j, n)$ approach will be more efficient.

3. MasPar implementation

The strategy for the computation (for both parallel architectures) involves (1) setting up the problem (specification of various computational parameters such as the time step τ, step sizes ΔR and ΔK, number of R and K grid points, etc.), (2) implementing the time step loop in which the current wave function and the current contribution to the tangent of the phase shift are computed, (3) carrying out the final analysis. In the time step loop, the various spherical Bessel transforms are computed; each of these involves taking the product of an $N \times N$ matrix with an $N \times 1$ vector. In both calculations (i.e., for the MasPar and the NCUBE machines) this is the time consuming step and therefore the goal is to have as efficient an algorithm for this step as is possible. We now discuss some details of the MasPar implementation, following the discussion given in Ref. [60].

A strictly top-down approach (i.e., one where the principal objectives are identified and the code structure is dictated by expanding the level of detail) was used in the software development resulting in an extremely clean and well structured code. As with more conventional supercomputers, specific vector operations were implemented as stand-alone calls and sequestered into a machine-specific vector library separate from the application code. These vector library routines will be discussed in detail later. Separation of machine-independent routines helped enormously in porting to the NCUBE.

The MasPar MP-1 is a single-instruction stream multiple-data stream (SIMD), or "data parallel" computer. A single instruction, issued by a central control unit and executed on each of the processing elements (PEs), operates on a different datum on each PE. There exist control structures to turn off selected PEs giving more control over which data the PEs operate on. All PEs are connected in a very high-speed toroidal mesh. The MP-1 configuration currently available at the SCF is an 8192 PE machine with 16 Kbytes of memory per PE. Peak speed is rated at 556 single-precision MFLOPS, and 10.2 BIPS. The 8K PEs are arranged in a 128 by 64 grid.

On all parallel computers it is important to involve as many processors as possible in the computation; for the MP-1 this is especially true. Since all PEs execute the same instruction stream, load balancing becomes a critical issue.

In this vein, we have chosen to distribute matrix elements to the PEs in a scattered decomposition as shown in Fig. 1. The PEs differ at most by one matrix element in the computation load they bear. Formally, matrix element a_{ij} is mapped to PE (x, y) by the following relations: $x = j \mathit{mod}\, \mathit{nxproc}$; $y = i \mathit{mod}\, \mathit{nyproc}$, where *nxproc* is the number of PEs in a row (128), and *nyproc* is the number of PEs in a column (64).

This scattered decomposition has the desirable feature that a given row of the matrix is stored entirely on an easily computed row of PEs. The same holds true for columns. This property contributes greatly to the efficiency and ease of implementation of the matrix-vector multiply routine.

An important fact must be noted at this point about the maximum possible size of a vector. All vectors in this application are of the same length, equal to the dimension of the Spherical Bessel transformation matrix. The amount of memory, M, consumed by this matrix per PE is:

$$M = [(N/\mathit{nxproc})][(N/\mathit{nyproc})](\text{storage needed for datum}) \qquad (17)$$

where the bracket [ ] indicates that the enclosed number is rounded to the next highest integer. The storage needed for a double precision word is 8 bytes, and for a single precision word it is 4 bytes.

As stated earlier the bulk of the computational effort is expended in the matrix-vector multiply operation. Since the Spherical Bessel matrix is symmetric, and is the only matrix in this application, vectors may be thought of as either columns or rows. We choose the row interpretation here.

In order to (post)multiply an $N \times N$ matrix by an $N \times 1$ column vector, we must first transpose the vector into a row vector, then compute dot products with each of the N rows of the matrix, and finally transpose the resulting column vector into a row vector. To perform these operations on the (scattered) matrix and (linearly distributed) vector stored on the MP-1 PE array, we use the following algorithm (see Fig. 2):

(1) Consolidate vector from linearly distributed decomposition to a scattered-row format on a single PE row,

(2) Broadcast consolidated vector to all PE rows,

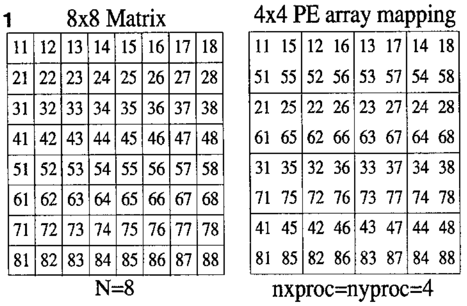
(3) Perform partial dot products with their matrix and vector elements for all PEs,

(4) Add up partial sums on all rows to give a resultant column vector,

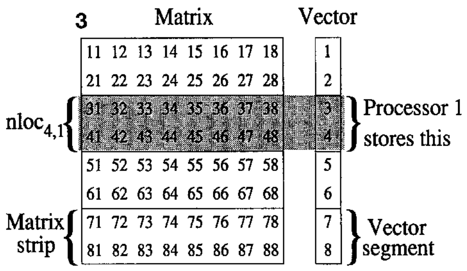
(5) Transpose and move resultant column vector into the standard linearly distributed format.

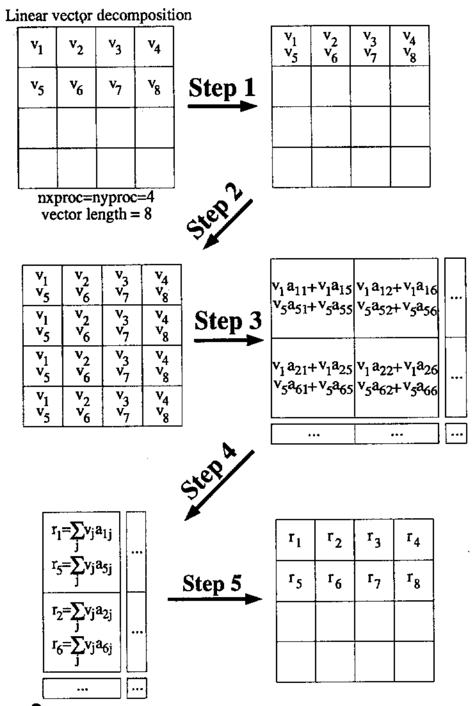
Note that all but the third step in the above algorithm imply a considerable amount of communication between PEs. This counter-intuitive method actually improves the efficiency of the matrix-multiply over methods which use less inter-PE communication.

1

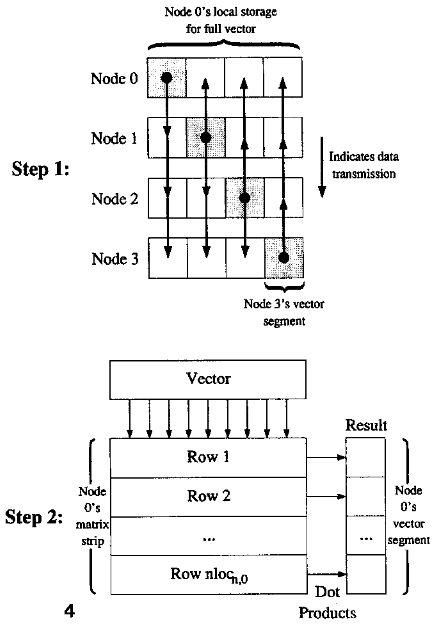


3





2



4

Fig. 1. MasPar matrix decomposition. An 8×8 matrix is shown for illustration purposes only. In general the matrix is $N \times N$. Also, a 4×4 PE array is shown for illustration. Actually *nxproc* = 128 and *nyproc* = 64 on the system used. In the illustrations, digit pairs represent (row, column) matrix element indices

Fig. 2. MasPar matrix-vector multiplication. Here v is the initial vector and r is the final vector. See text for a complete description of the multiply steps

Fig. 3. NCUBE matrix and vector decomposition

Fig. 4. NCUBE matrix-vector multiplication. See text for a complete description of the process

4. NCUBE implementation

Porting the scattering code to the NCUBE was largely a matter of providing the appropriate vector and matrix libraries. In fact, the wave-propagation loop transferred and ran *unaltered* to the NCUBE. Only the machine-dependent code had to be implemented anew. This observation is perhaps the first step toward debunking the myth that it is categorically difficult to port an application from one parallel architecture to another, especially between different classes of parallel architectures (i.e., SIMD to MIMD).

The NCUBE 6400 is a multiple-instruction stream, multiple-data stream (MIMD) parallel computer. Each processing element, or "node", executes its own program asynchronously from the other nodes. Nodes coordinate their activities by sending messages to one another; they are connected by a 2.2 MByte/s bidirectional links in a hypercube configuration. There is a 60 μs delay for each message before data are actually transmitted, due to software overhead.

The NCUBE 6400 system configuration at the Ames Laboratory is a 64 node system. Node 0 has 4 MBytes of memory; the others have 1 MByte each. Peak speed is rated at 171 single-precision MFLOPS, 130 double precision MFLOPS, and 640 MIPS.

Here we develop a mathematical construct which is of considerable value in describing the decomposition of both matrices and vectors in the NCUBE. While a scattered decomposition works well on the MP-1, it is not applicable to the MIMD NCUBE where communication is not nearly as fast. A detailed discussion of why a scattered decomposition is not used for the NCUBE implementation is given below.

Vectors are more easily managed if they are partitioned into nearly equal length segments. In cases where the length of a vector is a multiple of the number of processors, the length of a segment is simply the vector length divided by the number of processors. For nonintegral multiple sizes, a slightly modified formula must be used. First, we define a function which gives the total number of elements belonging to all processors with processor number less than that of the processor in question. That is:

$$cume_{N,i} = \{N(i)/(nproc)\} \tag{18}$$

where i is the processor number in question, and *nproc* is the total number of processors. Here the bracket $\{\ \}$ indicates the integral part of the number it contains. We next define the function $nloc_{N,i}$ by:

$$nloc_{N,i} = cume_{N,i+1} - cume_{N,i} \tag{19}$$

It tells how many rows of a matrix are stored by processor i; this is also the number of vector elements stored by i. We now have an easily evaluated function which yields the number of elements a particular processor possesses, even for vector sizes which are nonintegral multiples of the number of processors.

Matrices are decomposed in horizontal strips, or groups of rows. Thus for an $N \times N$ matrix, processor i will store rows $cume_{N,i} \rightarrow cume_{N,i+1} - 1$ inclusive. A given processor stores the same rows of a matrix as it does elements of a vector, as shown in Fig. 3.

In the vector consolidation step for the MP-1 matrix-vector multiply algorithm, the vector is quickly and easily consolidated onto a single row of PEs because of the very fast communication links. This operation takes place with the number of communication steps proportional to the vector length. On the MP-1, both the time to initiate a message and the time to send a datum are much faster than the time required to perform a floating point operation.

Such a communication performance is not typical of MIMD computers, and the NCUBE is no exception. Message start-up time alone is over 100 times more expensive than a single floating point operation. Once data actually start moving, the relative cost drops to about 4. Clearly, a few long messages are much preferred over many short ones for the NCUBE. This fact must be used as a guide in designing the task of vector consolidation on the NCUBE. Let us accept for the moment that the entire vector must be assembled on every node in the hypercube. We must devise a decomposition which allows us to put the vector together with the minimum *number* of messages between nodes, and with the least amount of unnecessary data movement. If the vector were distributed in the scattered decomposition, much rearranging would be necessary to put the elements in proper order once they were consolidated onto a single node. If each node stores contiguous elements of the vector, then no rearranging will be necessary.

The above discussion solves the problem of unnecessary data movement, but we must also consider the number of messages. Many efficient algorithms have been developed that utilize the hypercube interconnect to its full potential. Of interest to us here is the "hypercube collapse" method. Pairs of nodes exchange data across each hypercube dimension in turn. This scheme consolidates data in a number of communication steps equal to the log of the number of processors, which is absolute minimum number of messages possible.

Matrix-vector multiplication is less complicated with a strip-wise matrix decomposition since a single node contains whole rows of the matrix. Below is the algorithm we use to perform the multiplication, which is illustrated in Fig. 4:

(1) Consolidate vector from segment-wise decomposition onto every node in the hypercube. (Note that the hypercube broadcast is used here, which is not the inference that might be drawn from the 2-D depiction of arrows in Fig. 4.)

(2) Perform complete dot products between consolidated vector and each matrix row that a node owns and store dot products into result vector. Note that the result is already decomposed in segments.

5. Results

There are a variety of comparisons that can be made in order to assess the performance and efficiency of parallel supercomputers. By "performance" we mean the rate at which operations are carried out in aggregate for all processors of a machine. "Efficiency" is the ratio of observed performance to maximum theoretical performance. First, it was found that both the MP-1 and the NCUBE gave the correct phase shifts to within rounding accuracy and both machines significantly surpassed the serial architecture VAX 8650 in performance. Calculations were done using both single and double precision, with good accuracy for both and higher computation performance for the former. (However, it is not clear whether single precision will be adequate for more challenging three body systems.) Second, it was found, as expected, that the matrix vector multiplication was the major computational effort for both parallel machines. It should be noted also that both C-language and Fortran codes were tested on the MasPar machine. However, the Fortran compiler is far from optimal and produces machine code that ran no faster than the VAX 8650.

Third, it is of interest to compare the MP-1 and NCUBE 6400 as the dimensions of the matrices and vectors are increased. In Fig. 5 we see that the machines have nearly the same performance for $N \leqslant 512$. (The performance of

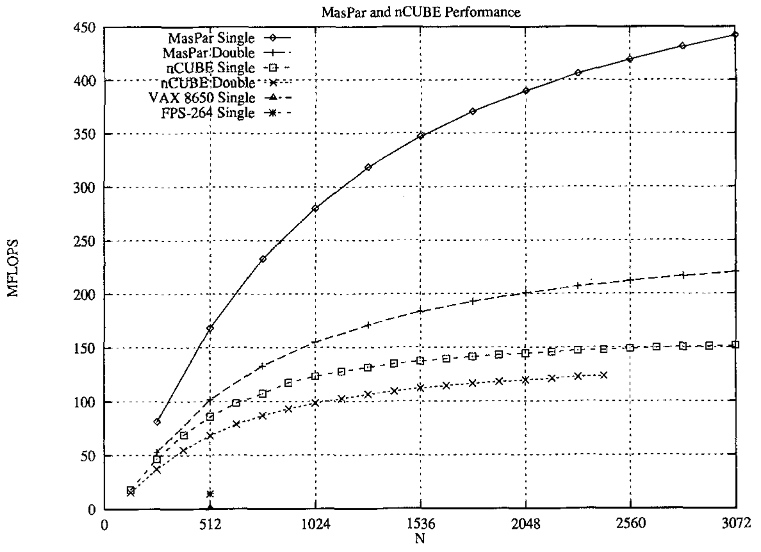


Fig. 5. MasPar and NCUBE performance

the VAX 8650 and the FPS-264 are also included in the figure for comparison.) In fact, for the scattering system considered, results are completely converged with 512 radial grid points. However, by considering larger N, we can gain insight into how the two machines will perform when more challenging systems are studied. It is clear from Fig. 5 that, for the present application the NCUBE 6400 reaches its peak computation speed for floating point operations at a lower value of N than the MP-1. Of course, it is also clear that the MP-1 is inherently the faster computer for floating point operations, and this will become manifest when treating larger problems.

However, in Fig. 6 we compare the efficiency of the two machines. It is seen that the NCUBE attains a higher efficiency, and does so at a lower N than the

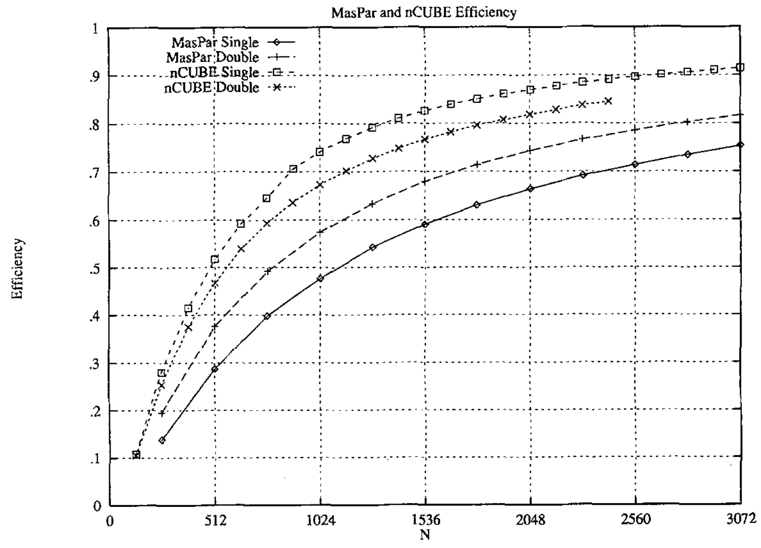


Fig. 6. MasPar and NCUBE efficiency

MP-1. Detailed analyses indicate that the principal factors responsible for this behavior are (a) the high initial cost of starting a message on the NCUBE decreases in relative importance as N increases, and (b) the high loop-start-up time of the MP-1 [60].

A fourth aspect of great interest is the manner in which the calculation scales with problem size, N, on the two computers. Since the dominant computational effort is the matrix-vector multiply, we know that the present integral equation method should scale as N^2 on a serial machine. On a parallel computer one expects, theoretically, a scaling reduction that is determined by the number of PEs that can be employed. If this number is P, then the theoretical limit is a scale factor of N^2/P. For the simple problem at hand the total number of processors for the MP-1 far exceeds the number of grid points required for a converged calculation, and thus if one processor could be assigned to each grid point, the calculation would scale as N. However, we actually do somewhat better than this because we are able to use all of the processors in the calculation. Thus, even though the calculation still scales as N^2 (like for the serial machine) the performance is better than the linear dependence which would be obtained if there were one processor per grid point. The situation is graphically displayed in Fig. 7, and illustrates the obvious but important fact that for any finite calculation both the scaling *and* the size of the coefficients in the scaling relation are important in determining the dependence of performance on problem size. The quadratic dependence of the NCUBE calculation on problem size is shown in Fig. 8.

On the basis of this initial study we have concluded the following. First, although the two machines we have used incorporate different architectures, the performance of each, measured in MFLOPS, is essentially the same (at least for $N \leqslant 512$). For $N > 512$, the greater power inherent in the MP-1 comes into play, and this machine is able to achieve a higher MFLOPS performance. However, the NCUBE remains more efficient, attaining close to its nominal MFLOPS performance at a smaller N value.

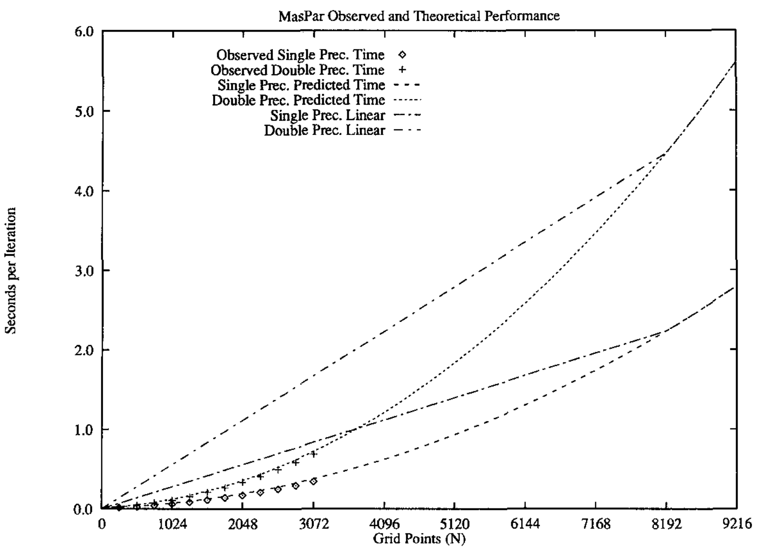


Fig. 7. MasPar observed and theoretical performance

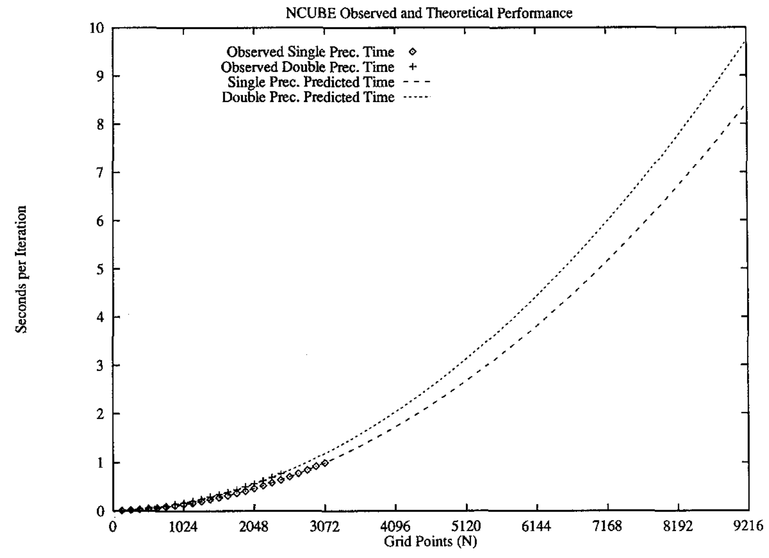


Fig. 8. NCUBE observed and theoretical performance

Second, both parallel machines outperformed the serial architecture VAX 8650 used for comparison. Further, the power of the parallel machines becomes increasingly manifest as the problem size is increased, due to the low scaling rate with N on the parallel machines compared to the serial VAX machine. This bodes very well for the potential usefulness of parallel machines for treating really large scattering problems, which are impractical for serial machines. In absolute terms, the NCUBE 6400 achieved 150 MFLOPS (single precision) and 120 MFLOPS (double precision) when $N = 2048$, while the MP-1 attained about 350 MFLOPS (single precision) and 190 MFLOPS (double precision) for the same size problem. These speeds are quite respectable compared to what can be achieved with a CRAY-type supercomputer, and attest to the suitability of parallel architecture computers for quantum scattering treated via the time-dependent integral equation formalism.

Third, we find that portability of codes between the two machines of quite different parallel architectures posed little problem due to the modular structure of the code. It was possible to concentrate the machine dependence of the codes into the modules. Thus, the various matrix-vector operations were compartmentalized, and written in the appropriate manner for the specific machine on which the code was to be run. These were then inserted as appropriate into a machine independent "driver" code. We anticipate that this is a general strategy that should be followed in order to minimize the effort required to port the code between various parallel computers.

Fourth, we have found that, at least for the existing compilers, codes written in C-language are much more efficient than those written in Fortran and parallelized by the compiler. Of course, as more effort is expended in optimizing the Fortran compiler, this situation may change. However, at present, only codes written in C will be able to take full advantage of the parallelism.

Finally, the present study indicates that parallel architecture will make possible the efficient implementation of new strategies for carrying out quantum scattering calculations.

6. Future studies

The next step in exploring the use of parallel processing supercomputers will be to carry out calculations for more challenging atom-diatom collision systems. The necessary generalization of the modified Cayley equations to describe such systems has already been carried out [58]. The major complication is that the single packet $\Psi(t)$ is replaced by a finite set of $\psi_n(t)$, where n is a quantum index that refers to the additional internal degrees of freedom in the system. The quantum representation can be chosen in a variety of ways, depending on which operators one desires to make diagonal. We expect to carry out such calculations in the near future.

In addition to the modified Cayley method, we are developing a number of other time-dependent integral equation approaches which should also be highly suited for use on parallel processing supercomputers. The most promising of these, at the present, is a three-term recursion formula for computing the wave packet at the current time from its value at two previous times [61]. Again, the implementation for atom-molecule collisions can be readily carried out. We will be adapting these recursion methods to parallel architecture computers, and applying them to a variety of scattering problems.

Finally, we will be testing these time-dependent methods on other architecture parallel processing supercomputers.

References

1. See the recent work on gas phase reactive scattering: (a) Time-independent, non-variational methods include Kuppermann A, Hipes PG (1986) J Chem Phys 84:5962 and (1987) Chem Phys Lett 133:1; Webster F, Light JC (1986) J Chem Phys 85:4744; (1989) *ibid.* 90:265 and 300; Haug K, Schwenke DW, Shima Y, Truhlar DG, Zhang JZH, Kouri DJ (1986) J Phys Chem 90:6757 and Zhang JZH, Kouri DJ, Haug K, Schwenke DW, Shima Y, Truhlar DG (1988) J Chem Phys 88:2492; Parker GA, Pack RT, Archer BJ, Walker RB (1987) Chem Phys Lett 137:564 and Pack RT, Parker GA (1987) J Chem Phys 87:3888; Baer M, Shima Y (1987) Phys Rev A35: 5252 and Baer M (1987) J Phys Chem 91:5846 and (1989) J Chem Phys 90:3043; Linderberg J, Vessel B (1987) Int J Quant Chem 31:65 and Linderberg J, Padkjaer SB, Ohrn Y, Vessel B (1989) J Chem Phys 90:6254; Schatz GC (1988) Chem Phys Lett 150:92 and (1988) 151:409; Launay JM, Lepetit D (1988) Chem Phys Lett 151:287 and (1988) 152:23 and Launay JM, Le Dourneff M (1989) *ibid.* 163:178. (b) Time-independent variational methods include Schwenke DW, Haug K, Truhlar DG, Sun Y, Zhang JZH, Kouri DJ (1987) J Phys Chem 91:6080 and Schwenke DW, Haug K, Zhao M, Truhlar DG, Sun Y, Zhang JZH, Kouri DJ (1988) *ibid.* 92:3202; Zhang JZH, Miller WH (1987) Chem Phys Lett 140:329 and (1988) J Chem Phys 88:449; Manolopoulos D, Wyatt RE (1988) Chem Phys Lett 152:23 and (1990) J Chem Phys 92:810. (c) Time-dependent studies have been reported by Neuhauser D, Baer M, Judson RS, Kouri DJ (1990) J Chem Phys 93:312 and Judson RS, Kouri DJ, Neuhauser D, Baer M (1990) Phys Rev A42:351
2. Some recent reviews include: (a) for molecule-surface scattering, Gerber RB, Kosloff R, Berman M (1986) Computer Phys Repts 5:59; (b) for colinear gas phase reactive scattering, Mohan V, Sathymurthy N (1988) Comp Phys Repts 7:213; and (c) a general review by Kosloff R (1988) J Phys Chem 92:2087
3. Mazur J, Rubin RJ (1959) J Chem Phys 31:1395
4. McCullough EA, Wyatt RE (1971) J Chem Phys 54:3578 and 3592
5. Zubert Ch, Kamal T, Zulike L (1975) Chem Phys Lett 36:396
6. Heller EJ (1975) J Chem Phys 62:1544; (1976) 65:4979; Kulander KC, Heller EJ (1978) *ibid.* 69:2439; Drolshagen G, Heller EJ (1983) *ibid.* 79:2072

7. Askar A, Cakmak AS (1978) J Chem Phys 68:2794
8. Kulander KC (1978) J Chem Phys 69:5064; Gray JC, Fraser GA, Truhlar DG, Kulander KC (1980) *ibid.* 73:5726; Orel AE, Kulander KC (1988) Chem Phys Lett 146:428
9. LeForestier C, Bergerson G, Hiberty PC (1981) Chem Phys Lett 84:385 (see also LeForestier C (1984) Chem Phys 87:241)
10. Agrawal PM, Raff LM (1982) J Chem Phys 77:3946
11. Heller EJ, Sunberg RL, Tannor DJ (1982) J Phys Chem 86:1822
12. Feit MD, Fleck JA (1983) J Chem Phys 79:301; (1984) 80:2578
13. Kosloff D, Kosloff R (1983) J Comput Phys 52:35; (1983) J Chem Phys 79:1823
14. LeForestier C (1984) Chem Phys 87:241; and in Clary DC (ed) Theory of chemical reaction dynamics (Reidel, Dordrecht, 1986) p 235; LeQuere F, LeForstier C (1990) J Chem Phys 92:247
15. Drolshagen G, Heller EJ (1983) J Chem Phys 79:2072
16. Tal-Ezer H, Kosloff R (1984) J Chem Phys 81:3967
17. Kosloff R, Cerjan C (1984) J Chem Phys 81:3722
18. Yinnon AT, Kosloff R, Gerber RB (1984) Surf Sci 148:148; Gerber RB, Yinnon AT, Kosloff R (1984) Chem Phys Lett 105:523
19. Imre D, Kinsey J, Sinha A, Krenos J (1984) J Phys Chem 88:395
20. Skodje RT (1984) Chem Phys Lett 109:227
21. Mowrey RC, Kouri DJ (1985) Chem Phys Lett 119:285
22. Jackson B, Metiu H (1985) J Chem Phys 83:1952; Sawada S, Heather R, Jackson B, Metiu H (1985) J Chem Phys 83:3009
23. Zhang ZH, Kouri DJ (1986) Phys Rev A34:2678
24. Jackson B, Metiu H (1986) J Chem Phys 84:3535; (1986) J Chem Phys 85:4192; (1987) J Chem Phys 86:1026
25. Mowrey RC, Kouri DJ (1986) J Chem Phys 84:6466
26. Park TJ, Light JC (1986) J Chem Phys 85:5870
27. Mohan V, Sathymurthy N (1986) Curr Sci 55:115; Thareja S, Sathymurthy N (1987) J Phys Chem 91:1970
28. Mowrey RC, Kouri DJ (1987) J Chem Phys 86:6140
29. Heather R, Metiu H (1987) J Chem Phys 86:5009; (1989) 90:6116; (1989) 91:1596
30. Kouri DJ, Mowrey RC (1987) J Chem Phys 87:2087
31. Sun Y, Mowrey RC, Kouri DJ (1987) J Chem Phys 87:339
32. Mowrey RC, Bowen HF, Kouri DJ (1987) J Chem Phys 86:2441; Mowrey RC, Bowen HF, Yinnon T, Gerber RB (1988) J Chem Phys 89:3925 and in preparation
33. Huber D, Heller EJ (1987) J Chem Phys 87:5302; (1988) 89:4752; Huber D, Heller EJ, Littejohn RG (1987) J Chem Phys 87:5302; Huber D, Ling S, Imre DG, Heller EJ (1989) J Chem Phys 90:7317
34. Sun Y, Kouri DJ (1988) J Chem Phys 89:2958
35. Jackson B (1988) J Chem Phys 88:1383; (1988) 89:2473; (1991) Computer Phys Commun 63:154
36. Chaseman D, Tannor DJ, Imre DG (1988) J Chem Phys 89:6667
37. Williams SO, Imre DJ (1988) J Phys Chem 92:6648
38. Tannor DJ, Rice SA (1988) Adv Chem Phys 70:441
39. Sun Y, Judson RS, Kouri DJ (1989) J Chem Phys 90:241; Sun Y, Kouri DJ, Schwenke DW, Truhlar DG (1991) Computer Phys Commun 63:51
40. Neuhauser D, Baer M (1989) J Chem Phys 90:4351 and (1989) J Phys Chem 93:2872; (1989) J Chem Phys 91:4651
41. Neuhauser D, Baer M, Judson RS, Kouri DJ (1989) J Chem Phys 90:5882 and (1990) 93:312; Comput Phys Comm 63:460
42. Jiang X-P, Heather R, Metiu H (1989) J Chem Phys 90:2555
43. Jacon M, Atabek O, LeForestier C (1989) J Chem Phys 91:1585
44. Zhang J, Imre DG (1989) J Chem Phys 90:1666
45. Dixon RN (1989) Mol Phys 68:263
46. Judson RS, Kouri DJ, Neuhauser D, Baer M (1990) Phys Rev A42:351
47. Zhang JZH (1989) Chem Phys Lett 160:417 and (1990) J Chem Phys 92:324
48. Hoffman DK, Sharafeddin O, Judson RS, Kouri DJ (1990) J Chem Phys 92:4167

49. Das S, Tannor DJ (1990) J Chem Phys 92:3403
50. Sharafeddin OA, Kouri DJ, Judson RS, Hoffman DK (1990) J Chem Phys 93:5580
51. Judson RS, McGarrah DB, Sharafeddin OA, Kouri DJ, Hoffman DK (1991) J Chem Phys 94:3577
52. Neuhauser D, Judson RS J Chem Phys (submitted)
53. Neuhauser D, Judson RS, Baer M, Kouri DJ (1991) Chem Phys Lett 176:546
54. Sharafeddin OA, Bowen HF, Kouri DJ, Hoffman DK J Comput Phys (in press)
55. Viswanathan R, Shi S, Villalonga E, Rabitz H (1989) J Chem Phys 91:2333
56. Neuhauser D J Chem Phys (in press)
57. Founargiotakis M, Light JC (1990) J Chem Phys 93:633
58. Sharafeddin OA, Kouri DJ, Judson RS, Hoffman DK (unpublished)
59. Hirschfelder JO, Wigner EP (1935) Proc Natl Acad Sci (USA) 21:113; Curtiss CF, Hirschfelder JO, Adler FT (1950) J Chem Phys 18:1638; Curtiss CF (1953) *ibid.* 21:1199; Kouri DJ, Curtiss CF (1966) *ibid.* 44:2120; Curtiss CF (1968) *ibid.* 48:1725; Thorson WR (1965) *ibid.* 42:3878; Pack RT, Hirschfelder JO (1968) *ibid.* 49:4009; Lawley KP, Ross J (1965) *ibid.* 43:2930, 2943; Jacob M, Wick GC (1959) Ann Phys (NY) 7:404; Pack RT (1974) J Chem Phys 60:633; McGuire P, Kouri DJ (1974) 60:2499; Klar H (1973) J Phys B6:2139; Tamir M, Shapiro M (1975) Chem Phys Lett 31:166; Kouri DJ, Heil TG, Shimoni Y (1976) J Chem Phys 65:226
60. Carter M, Nayer N, Gustafson J, Hoffman DK, Sharafeddin OA, Kouri DJ (to be published)
61. Hoffman DK, Ma X, Kouri DJ (in preparation) J Chem Phys