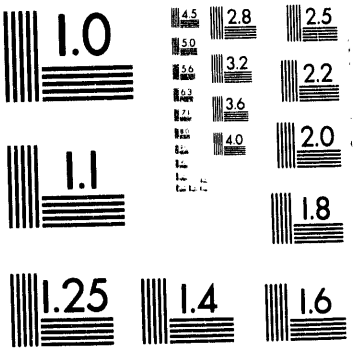
1.0
1.1
1.25
1.4
1.6
1.8
2.0
2.2
2.5
2.8
3.2
3.6
4.0
4.5
5.0
5.6

1 of 1

Conf-9309176--4

Large-Scale 3-D Solutions of the Time-Dependent Dirac Equation

BY

A.S. Umar, J.C. Wells, V.E. Oberacker, and M.R. Strayer

To appear in

Second International Conference On

COMPUTATIONAL PHYSICS

September 13-17, 1993
Beijing, China

Large-Scale 3-D Solutions of the Time-Dependent Dirac Equation

A. S. UMAR[1,3], J. C. WELLS[1,2,3], V. E. OBERACKER[1,3], M. R. STRAYER[1,2]

[1] *Center for Computationally Intensive Physics, Oak Ridge National Laboratory, Oak Ridge, Tennessee 37831, USA;*

[2] *Physics Division, Oak Ridge National Laboratory, Oak Ridge, Tennessee 37831, USA;*

and

[3] *Department of Physics & Astronomy, Vanderbilt University, Nashville, Tennessee 37235, USA*

Abstract

We present the numerical approach used in solving the time-dependent Dirac equation on a three-dimensional Cartesian lattice. Discretization is achieved through the lattice basis-spline collocation method, in which quantum-state vectors and coordinate-space operators are expressed in terms of basis-spline functions on a spatial lattice. All numerical procedures reduce to a series of matrix-vector operations which we perform on the Intel iPSC/860 hypercube, making full use of parallelism. We discuss our solutions to the problems of limited node memory and node-to-node communication overhead inherent in using distributed-memory,

1 Introduction

In this talk, we present the numerical methods developed for solving the time-dependent Dirac equation in three space dimensions on parallel computers. The Dirac equation is one of the fundamental equations of nature, being the relativistic analogue of the Schrödinger equation. We will present computational applications of the Dirac equation to the lepton pair-production problem of quantum electrodynamics. Since this is a computational talk I will talk principally about the numerical methods used in our calculations. The theoretical details of the lepton-pair production and the reduction of the quantum electrodynamics to the well known Dirac equation is extensively discussed in [1]. Here, I will concentrate on the lepton pair production followed by the capture of the negative lepton into the $1s$ atomic bound state of the target nucleus. It is formally shown (using time-reversal invariance) in the above reference that this process can be described by the evolution of the Dirac $1s$ state backwards in time under the influence of the external time-dependent electromagnetic field of the passing projectile. For colliders such as the Relativistic Heavy-Ion Collider (RHIC) this external Coulomb field will be amplified by the relativistic γ factor, which could be as large as 20,000. Thus, we deal with a strong field problem that has no perturbative solution and requires an exact numerical treatment. Figure 1 shows this dynamical process.

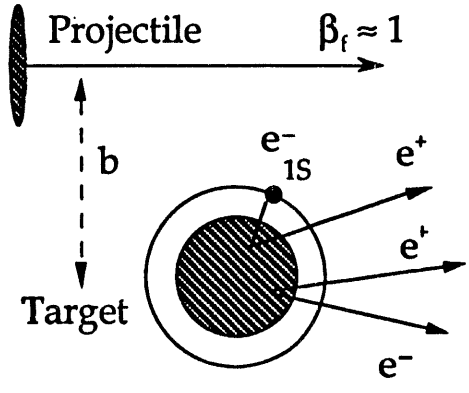

Figure 1: Depicted are two heavy ions colliding at finite b in the target reference frame of a heavy ion, with lepton-pair production with subsequent capture of the negative lepton.

2 Lepton-Pair Production

Our starting point is the set of equations derived in Refs. [1, 2], in which we presented a nonperturbative approach to electromagnetic lepton-pair production in peripheral heavy-ion collisions which is applicable over

a wide range of relativistic energies. Beginning with the general formulation of Quantum Electrodynamics, we made reasonable assumptions about the nature of the lepton and radiation fields in relativistic heavy-ion collisions which reduced the equations of motion to the time-dependent Dirac equation for the lepton field interacting with the classical, electromagnetic fields produced in the heavy-ion collisions. In discussing the solution of the Dirac equation, we use natural units i.e. $\hbar = c = m = 1$.

We study the electromagnetic production of lepton pairs in a reference frame in which one of the nuclei, henceforth referred to as the target, is at rest. The target nucleus and the lepton interact via the static Coulomb field, $A^0_{\rm T}$. The only time-dependent interaction, $A^\mu_{\rm P}(t)$, arises from the classical motion of the projectile. Thus, it is natural to split the Dirac Hamiltonian into static and time-dependent parts. Accordingly, we write the Dirac equation for a lepton described by a spinor $\phi(\vec{r},t)$ coupled to an external, time-dependent electromagnetic field as

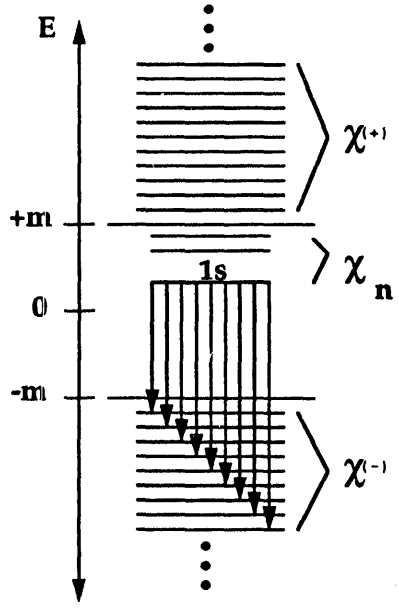

Figure 2: Depicted is the Furry spectrum consisting of bound states (χ_n), positive ($\chi^{(+)}$) and negative energy ($\chi^{(-)}$) free states. Equation (5) is illustrated by showing transitions of the $1s$ state to the negative-energy states.

$$[H_F + H_{\rm P}(t)]\phi(\vec{r},t) = i\frac{\partial}{\partial t}\phi(\vec{r},t), \tag{1}$$

where the static Furry Hamiltonian, $H_{\rm F}$, which describes a lepton in the presence of the strong, external Coulomb field of the target nucleus, is given by

$$H_{\rm F} = -i\vec{\alpha}\cdot\nabla + \beta - eA^0_{\rm T}\,, \tag{2}$$

and the time-dependent interaction of the lepton with the projectile is

$$H_{\rm P}(t) = e\vec{\alpha}\cdot\vec{A}_{\rm P}(t) - eA^0_{\rm P}(t)\,. \tag{3}$$

We define the stationary states of the system, i.e. the eigenstates of the Furry Hamiltonian $H_{\rm F}$ in Eq. (2), as

$$H_{\rm F}\chi_i(\vec{r}) = E_i\chi_i(\vec{r})\,, \tag{4}$$

which are also proper ingoing and outgoing states for asymptotic times $|t| \to \infty$, where the interaction $H_{\rm P}(t)$ is zero. The index i in Eq. (4) represents the complete set of quantum numbers for the single-particle state $\chi_i(\vec{r})$.

The probability P_b for lepton-pair production from the vacuum with capture of the negative lepton into a bound state b may be written

$$P_b(t) = \sum_{r<-m_0c^2} |\langle\chi^{(-)}_r|\phi^{(+)}_b(t)\rangle|^2\,. \tag{5}$$

To compute probabilities for lepton-pair production with capture, we square the projection of single-particle solutions of the time-dependent Dirac equation, initially in the bound state b, onto the static free states with negative energy. Equation (5) is illustrated in Fig. 2. Measurable probabilities are the asymptotic ($t \to \infty$) limit of Eq. (5).

3 Numerical Implementation

We solve the time-dependent Dirac equation using a lattice approach to obtain a discrete representation of all Dirac spinors and coordinate-space operators on a three-dimensional Cartesian mesh. We implement our lattice solution using the basis-spline collocation method, which is briefly summarized in section 3.1. This method uses high-order interpolates to obtain an accurate and stable representation of functions and operators on the lattice. We limit this discussion to the special case of cubic lattices with uniform spacing in all three directions. However, the basis-spline collocation method is equally well suited for nonuniform lattice spacings. In order to provide the initial state for our time-dependent calculations, we must solve for the Furry bound state of interest and then evolve this state in time.

3.1 Lattice Basis-Spline Collocation Method

Splines of order M are functions $S^M(x)$ of a single, real variable belonging to the class $C^{(M-2)}$ with continuous $(M-2)$th derivatives. These functions are piecewise continuous, as they are constructed from continuous

polynomials of $(M-1)$th order joined at points in an ordered set $\{x'_i\}$ called knots. Basis-splines have minimal support in that they are zero outside the range of $M+1$ consecutive knots $x'_i, x'_{i+1}, \ldots, x'_{i+M}$, and non-negative otherwise. We label these functions with the index of their first knot $B_i^M(x)$.

Consider a region of space with boundaries at $x_{\min}$ and $x_{\max}$ containing $n+1$ knots, including the knots on the boundaries. For a set of Mth-order basis-splines to be complete, M of the functions must be nonzero on each knot interval $[x'_i, x'_{i+1}]$ within the physical region. For this to occur, $M-1$ basis-splines must extend outside each boundary. Therefore, we require a knot sequence from x'_1 to x'_{n+2M-1} in order to construct a complete set of functions, where x'_M and x'_{M+n} correspond to the lower and upper physical boundaries, respectively. The basis-splines for the region are naturally numbered as $B_1^M(x), B_2^M(x), \ldots, B_{n+M-1}^M(x)$, as shown in Fig. 3.

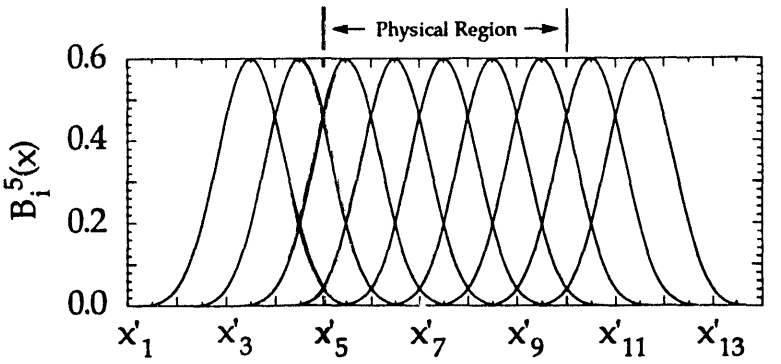

Figure 3: Depicted is a complete set of 5th-order basis-spline functions for the region $x'_5 \le x \le x'_{10}$ for homogeneous lattice spacing. The functions are labeled from the left as $B_1^5(x), B_2^5(x), \ldots, B_9^5(x)$.

We expand each of the four components of the Dirac spinor, distinguished by the index p, in this tensor-product basis as,

$$\chi^{(p)}(x,y,z) = \sum_{i,j,k=1}^{n+M-1} B_i^M(x) B_j^M(y) B_k^M(z) c_{(p)}^{ijk}, \quad p = 1,2,3,4\,, \tag{6}$$

where we suppress quantum-number indices for clarity and $\{c_{(p)}^{ijk}\}$ denotes the complex expansion coefficients for the p^{th} component of the spinor. All state functions in this paper satisfy periodic boundary conditions which are easily imposed by identifying $B_{n+1}^M, \ldots, B_{n+M-1}^M$ with $B_1^M, \ldots, B_{M-1}^M$, respectively. Consequently, the basis-spline index i will run to n instead of $n+M-1$. Procedures for implementing general nonperiodic boundary conditions are discussed in Ref. [3]

We forsake the continuous description of the Dirac spinor for a lattice representation of $\chi(\vec{r})$ through the collocation method, in which the spinor is known only at each of the collocation points $(x_\alpha, y_\beta, z_\gamma)$ which define the lattice; thus $\chi(\vec{r})$ is replaced by $\chi_{\alpha,\beta,\gamma}$. Using boldface print to denote vectors and matrices in collocation space, the Dirac spinor $\boldsymbol{\chi}$ will be a column vector of $4n^3$ complex numbers. To implement the lattice description of the Dirac spinor using the basis-spline expansion, we create a linear system of equations by evaluating Eq. (6) at the collocation points

$$\chi_{\alpha\beta\gamma}^{(p)} = \sum_{i,j,k=1}^{n} B_{\alpha i} B_{\beta j} B_{\gamma k} c_{(p)}^{ijk}\,, \quad p = 1,2,3,4\,, \tag{7}$$

where $B_{\alpha i} \equiv B_i^M(x_\alpha)$ and the order M is omitted for simplicity. Collocation points may be chosen with some freedom. An optimal and simple choice for odd-order basis-splines is to place one collocation point at the center of each equally spaced knot interval within the physical boundaries,

$$x_\alpha = \tfrac{1}{2}(x'_{\alpha+M-1} + x'_{\alpha+M}), \quad \alpha = 1, \ldots, n\,. \tag{8}$$

The collocation points are denoted by Greek indices. The essence of the lattice approach is to eliminate the expansion coefficients $c_{(p)}^{ijk}$ from the set of equations in Eq. (7) using the inverse of the matrices $B_{\alpha i}$, $B_{\beta j}$, and $B_{\gamma k}$,

$$c_{(p)}^{ijk} = \sum_{\alpha,\beta,\gamma}^{n} B^{i\alpha} B^{j\beta} B^{k\gamma} \chi_{\alpha\beta\gamma}^{(p)}\,, \tag{9}$$

where the inverse matrix, denoted $B^{i\alpha} \equiv [B^{-1}]_{\alpha i}$, plays the role of a metric in the discrete collocation space. The choice of the collocation points in Eq. (8) ensures that the matrix $B^{i\alpha}$ is nonsingular.

We now discuss the collocation-lattice representation of a linear, coordinate-space operator $\mathcal{O}$ by considering its action on the basis-spline expansion of $\chi(\vec{r})$ in Eq. (6),

$$[\mathcal{O}\chi^{(p)}]_{\alpha\beta\gamma} = \sum_{i,j,k=1}^{n} [\mathcal{O}B_i^M(x)B_j^M(y)B_k^M(z)]_{x_\alpha,y_\beta,z_\gamma} c_{(p)}^{ijk} \ . \tag{10}$$

We now eliminate the expansion coefficients $c_{(p)}^{ijk}$ from Eq. (10) using Eq. (9) to obtain

$$[\mathcal{O}\chi^{(p)}]_{\alpha\beta\gamma} = \sum_{\mu,\nu,\xi=1}^{n} O_{\alpha\beta\gamma}^{\mu\nu\xi} \chi_{\mu\nu\xi}^{(p)} \ , \tag{11}$$

where we define the lattice representation of the coordinate-space operator as

$$O_{\alpha\beta\gamma}^{\mu\nu\xi} \equiv \sum_{i,j,k=1}^{n} [\mathcal{O}B_i^M(x)B_j^M(y)B_k^M(z)]_{x_\alpha,y_\beta,z_\gamma} B^{i\mu}B^{j\nu}B^{k\xi} \ . \tag{12}$$

In summary, the collocation points define the lattice on which the calculations are performed; neither the splines nor the knots appear explicitly again once the lattice representation of the operators has been obtained at the beginning of the calculation. We have reduced the partial differential equation (Eq. (4)) to a series of linear algebraic equations which may be solved using iterative techniques. As a consequence of eliminating the expansion coefficients from the theory, $\mathbf{H_F}$ has a blocked sparse representation which is self-adjoint for periodic boundary conditions and uniform meshes.

3.2 Computation of the Initial Bound State

The complete eigensolution of $\mathbf{H_F}$, providing its full spectrum of stationary states, currently approaches the state-of-the-art in computational capabilities due to the size of $\mathbf{H_F}$, which is equivalent to a rank $8n^3$ real matrix. We believe convergent calculations will be achieved for $n \approx 100$, based on the length and momentum scales involved, and experience with one-dimensional calculations. For this reason, we compute the lowest energy bound state ($1s$) needed as the initial state for our time-dependent problems by a partial eigensolution of $\mathbf{H_F}$.

Standard methods for partial eigensolution of large matrices which are designed to converge to the lowest energy eigenstate of the spectrum are not directly applicable for computing the $1s$ state of $\mathbf{H_F}$ because its spectrum extends to negative energies. The continuous operator H_F has positive and negative continua, $E > mc^2$ and $E < mc^2$, as well as bound states $|E| < mc^2$; the spectrum of $\mathbf{H_F}$ has the same branches though all the eigenvalues are discrete. In Ref. [2] the $1s$ state was computed using a damped relaxation method discussed in detail in Ref. [4] This algorithm is constructed to remove the high-frequency components from the residual, and does not depend on the spectrum of $\mathbf{H_F}$ being bounded from below.

For larger lattice sizes discussed in this paper, we have developed a more efficient iterative Lanczos algorithm to compute the initial state.[5, 6, 7] The Lanczos algorithm proves attractive for our purposes as the memory requirements are relatively small and the method approximates extremal eigenvalues in the spectrum very well. Since convergence is most rapid for extremal eigenvalues, we solve for the lowest energy eigenstate of $\mathbf{H_F}^2$, which has a positive-definite spectrum. By solving for the ground state of $\mathbf{H_F}^2$, we obtain the lowest-energy bound state of $\mathbf{H_F}$.

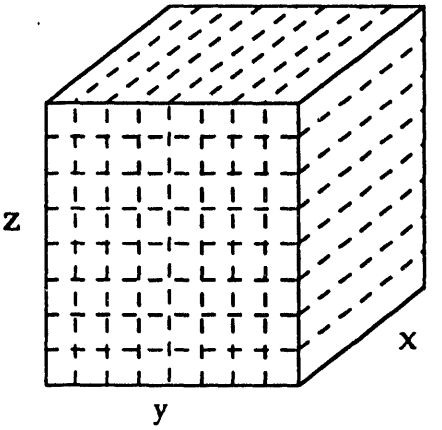

Figure 4: Illustrated is an example of partitioning a 3-D Cartesian lattice with 64^3 points into subblocks. In this example, the number of processors allocated to each of the two dimensions is 8, requiring each of the 64 subblocks to have dimensions $64 \times 8 \times 8$.

3.3 Time Evolution

We discretize time in the sense that the electromagnetic interactions are taken as constant in successive small intervals of possibly varying size Δt_ℓ , and express the evolution operator in successive infinitesimal steps. A

number of different methods have been employed to approximate the infinitesimal time-evolution operator

$$\hat{U}(t_{\ell+1}, t_\ell) = \exp\left(-i\left[H_{\rm F} + H_{\rm P}(t_{\ell+1})\right]\Delta t_{\ell+1}\right) , \tag{13}$$

particularly in studies of the time-dependent Hartree-Fock method applied to atomic and nuclear collisions.[8, 9] The choice of a method usually depends on the dimensionality and structure of the Hamiltonian matrix. Several methods which work well in one- and two-dimensional problems are impractical for unrestricted three-dimensional problems because they require the inversion of part or all of the Hamiltonian matrix.[8] In our three-dimensional solution of the Dirac equation, the exponential operator, Eq. (13), is implemented as a Taylor series expansion

$$\hat{\mathbf{U}}(t_{\ell+1}, t_\ell) \approx \left(1 + \sum_{k=1}^{K} \frac{(-i\,\Delta t_{\ell+1})^k}{k!}\left[\mathbf{H}_{\rm F} + \mathbf{H}_{\rm P}(t_{\ell+1})\right]^k\right) , \tag{14}$$

where K is the maximum number of allowed terms in the Taylor series expansion, chosen at each step according to a convergence criterion on the wavefunction.

4 Details of the Hypercube Implementation

In this section, we discuss the details of implementing the lattice representation of the time-dependent Dirac equation on the Intel iPSC/860/RX hypercube massively parallel computer at the Oak Ridge National Laboratory (ORNL).

As with many parallel implementations, we face the problem of limited memory per node and the optimization of the algorithm to minimize the communication between nodes. We deal with these issues in practice by partitioning the three-dimensional spatial lattice into subblocks and distributing the lattice subblocks to the nodes of the hypercube.

4.1 Spatial Distribution of Dirac Spinors and Local Arrays

As discussed in Sec. 3, Dirac spinors are represented on a three-dimensional Cartesian lattice in our numerical solution of the Dirac equation (Eq. (1))

$$\phi(x,y,z,t) = \begin{pmatrix} \phi^{(1)}(x,y,z,t) \\ \phi^{(2)}(x,y,z,t) \\ \phi^{(3)}(x,y,z,t) \\ \phi^{(4)}(x,y,z,t) \end{pmatrix} \longrightarrow \begin{pmatrix} \phi^{(1)}(x_\alpha,y_\beta,z_\gamma,t) \\ \phi^{(2)}(x_\alpha,y_\beta,z_\gamma,t) \\ \phi^{(3)}(x_\alpha,y_\beta,z_\gamma,t) \\ \phi^{(4)}(x_\alpha,y_\beta,z_\gamma,t) \end{pmatrix} , \tag{15}$$

where x_α, y_β, and z_γ denote the collocation lattice points in the x, y, and z directions, respectively. We suppress quantum number indices for clarity. Indices in parenthesis denote the Dirac spinor component. In the following, we denote the number of lattice points in the three Cartesian directions by n_x, n_y, and n_z. We choose to parallelize the time-dependent Dirac equation by data decomposition. In practice, we choose to partition the y and z dimensions of the lattice into subblocks while maintaining the full x dimension on each node. Since we need to distribute two dimensions of a three-dimensional array across the hypercube, we maintain optimal nearest-neighbor communication between nodes using the two-dimensional Gray lattice identification scheme. We now discuss the details of this partition and distribution of the lattice subblocks onto the processors.

To maximize the occurrence of nearest-neighbor communication, the number of lattice points in the y and z directions are chosen to be powers of two. Given a total number of nodes, the task of partitioning begins with finding the number of nodes needed in the y and z directions to form the Gray lattice. If the number of allocated nodes, p, is an exact square (i.e. 4, 16, 64, etc.) we allocate $p_z = \sqrt{p}$ and $p_y = \sqrt{p}$ nodes in y and z directions, respectively. This results in a square Gray lattice. For intermediate powers of two (i.e. 2, 8, 32, etc.), the partition is performed by $p_z = \sqrt{2p}$, $p_y = \sqrt{p/2}$, thus resulting in a rectangular Gray lattice. After the determination of the node allocations in y and z directions, we determine the number of lattice points kept on each node by

$$m_y = \frac{n_y}{p_y} , \qquad m_z = \frac{n_z}{p_z} . \tag{16}$$

Thus, all local arrays have a spatial dimension of $n_x m_y m_z$ on each node. An example is shown in Fig. 4.

4.2 · The Ring Algorithm for the Dirac Hamiltonian

All of our iterative algorithms for the solution of the Dirac equation make use of the operation of the Dirac Hamiltonian matrix multiplying the lattice representation of a Dirac spinor, $\phi' = \mathbf{H}_D\phi$. In our lattice representation, the action of the Hamiltonian on a spinor is given schematically in Eq. (17). Using Cartesian coordinates, this product naturally decomposes into four parts, one for each coordinate direction (x, y, z), and a diagonal part. This separability makes it easy to define this product implicitly in a storage-efficient way. The explicit Hamiltonian matrix is never created in memory, reducing our memory requirements from order n^6 to order n^3.

The Dirac Hamiltonian matrix $\mathbf{H}_D$ contains local potential terms, which are diagonal matrices, and nonlocal derivative terms, which are dense matrices, as described in Sec. 3.2. Performing matrix-vector multiplications with the nonlocal summations in the y and z dimensions requires node-to-node communication as these dimensions of the lattice are distributed across the processors. These terms which require communication are shown in brackets in Eq. (17).

$$
\begin{aligned}
\left(\phi^{(s)}\right)'(\alpha,\beta,\gamma) = \; & - \; i\sum_{\alpha'=1}^{n_x} D_{\alpha,\alpha'} \sum_{s'=1}^{4} \alpha_x^{(ss')}\phi^{(s')}(\alpha',\beta,\gamma) \\
& - \sum_{i_c=1}^{p_y-1}\left[i \sum_{\beta'=1+m_y(i_c-1)}^{m_y+m_y(i_c-1)} D_{\beta,\beta'} \sum_{s'=1}^{4} \alpha_y^{(ss')}\phi^{(s')}(\alpha,\beta',\gamma)\right] \\
& - \sum_{i_r=1}^{p_z-1}\left[i \sum_{\gamma'=1+m_z(i_r-1)}^{m_z+m_z(i_r-1)} D_{\gamma,\gamma'} \sum_{s'=1}^{4} \alpha_z^{(ss')}\phi^{(s')}(\alpha,\beta,\gamma')\right] \\
& + \; eA_x(i,j,k)\sum_{s'=1}^{4}\alpha_x^{(ss')}\phi^{(s')}(\alpha,\beta,\gamma) \\
& + \; eA_y(i,j,k)\sum_{s'=1}^{4}\alpha_y^{(ss')}\phi^{(s')}(\alpha,\beta,\gamma) \\
& + \; eA_z(i,j,k)\sum_{s'=1}^{4}\alpha_z^{(ss')}\phi^{(s')}(\alpha,\beta,\gamma) \\
& + \; \sum_{s'=1}^{4}\left([\beta]^{(ss')} - eA^0(i,j,k)\delta_{ss'}\right)\phi^{(s')}(\alpha,\beta,\gamma)\,, \qquad (17)
\end{aligned}
$$

where the matrix D is the lattice representation of the first derivative, α and β are the 4×4 Dirac spin matrices, and the diagonal matrices A_1, A_2, A_3, and A^0 are the components of the electromagnetic interaction, all evaluated on the collocation lattice.

In the execution of the y and z nonlocal sums in Eq. (17), we use a ring algorithm to perform these nonlocal matrix-vector operations economically. To accomplish the summation over the entire range of the y and z coordinates, each subblock of the Dirac spinor must visit each node once. This is achieved by having loops over the number of y and z nodes (p_y and p_z) performed on each node as shown in Eq. (17). All the derivative matrices are stored in full on each node.

4.3 Communication Overhead and Speedup

In discussing the performance of our application, we will consider only Eq. (17), the matrix-vector product discussed in Sec. 4.5, as this operation consumes more than 95% of the computational effort needed in solving the time-dependent Dirac equation. Currently, the large memory requirement of our three-dimensional solution limits the size of the lattice we consider to be smaller than the desirable 100^3 points. Therefore, we are particularly interested in the scaling properties of our parallel algorithm to larger parallel machines, such as the Intel Paragon, as well as our codes performance on the Intel iPSC/860.

In our application, the overall run time, as well as the ratio of the communication time T_{comm} to the calculation time T_{calc}, i.e. the fractional communication overhead f_c,[10] is improved by having the communication and calculation occur concurrently whenever possible. However, we neglect this feature in our scaling model, as it complicates the modeling and is not essential in obtaining a reasonable scaling.

To execute Eq. (17) once, the total theoretical time per node needed to perform floating-point operations ($T_{\rm calc}$) is the number of floating-point operations required, multiplied by the time $t_{\rm flop}(n)$ required to perform a single 64-bit floating-point operation within a vector of length n words. Assuming that the lattice has an equal number of points in the three coordinate directions, $n_x = n_y = n_z = n$, the estimated calculation time for Eq. (17) is

$$T_{\rm calc} = (48n + 448)\frac{n^3}{p} t_{\rm flop}(n) \ . \tag{18}$$

The dependence of $t_{\rm flop}(n)$ on n is caused by the pipelined floating-point units of the i860 processor. We determine that $t_{\rm flop}(n)$ varies with n as the inverse of a logarithmic function

$$t_{\rm flop}(n) \approx \frac{1}{(15.1\log(n) - 9.9) \times 10^6} \ \text{seconds} \tag{19}$$

over n ranging from 8 to 128. Substituting Eq. (19) into Eq. (18), we obtain

$$T_{\rm calc} \approx \frac{(48n^4 + 448n^3)}{p(15.1\log(n) - 9.9) \times 10^6} \ \text{seconds} \ . \tag{20}$$

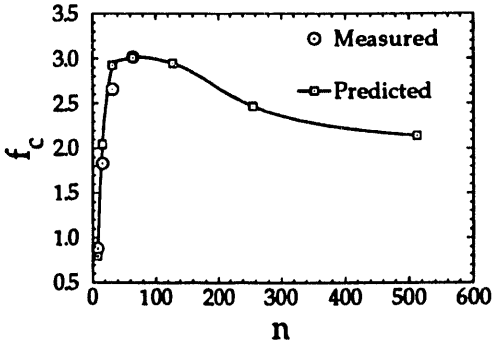

Figure 5: Plotted is the fractional communication overhead f_c as a function of the lattice size n obtained from the predictions in Eqs. (20) and (21).

Empirically, the communication time for a one-hop node-to-node message is a linear function of the size of the message.[11] In performing the nonlocal summations in Eq. (17), we are required to pass $p_y + p_z$ messages of length $8n^3/p$ 64-bit words. Passing these subblocks of the Dirac spinor around the two-dimensional Gray lattice ideally consumes the time

$$T_{\rm comm} = (p_y + p_z)\left(8\frac{n^3}{p} t_{\rm comm} + t_{\rm start}\right) + T_{\rm ohead} \ . \tag{21}$$

where $t_{\rm comm}$ is the typical time needed to actually transmit a single 64-bit word of data between two nodes, and $t_{\rm start}$ is the startup time for a single communication request. Typical times for the iPSC/860 are $t_{\rm comm} = 3.2 \times 10^{-6}$sec, and $t_{\rm start} = 1.36 \times 10^{-4}$sec.[11] Despite the fact that $t_{\rm start}$ is about 40 times larger than $t_{\rm comm}$, the startup contribution to $T_{\rm comm}$ is small for realistic calculations since the size of the messages passed between nodes, i.e $8n^3/p$ words, is large. We adjust $T_{\rm ohead}$ to fit the measured communication time.

Other important measures of the performance of a parallel application are the speedup $S(p)$ and the parallel efficiency $\epsilon(p)$. The speedup is defined as the ratio of time required to complete a given calculation on a single node to the time required to perform the equivalent calculation on p concurrent nodes.[10] We define $T_{\rm hdprd}(p)$ to be the time needed for computing Eq. (17) on a multiprocessor with p nodes in double precision; the subscript is the name of the subroutine which performs this operation. The speedup for Eq. (17) may be expressed in this notation as

$$S(p) \equiv \frac{T_{\rm hdprd}(1)}{T_{\rm hdprd}(p)} \ . \tag{22}$$

The parallel efficiency is defined as the speedup scaled by the number of processors

$$\epsilon(p) \equiv \frac{S(p)}{p} \ . \tag{23}$$

5 Timing Results

The predicted fractional communication overhead obtained using Eqs. (20) and (21) and the measured values of this quantity are compared in Fig. 5. Notice that the predicted and measured values for this quantity agree well throughout the range of problem sizes, $8 \le n \le 64$ and that the communication overhead increases rapidly up to $n = 64$. This initial increase in overhead with problem size at first seems counterintuitive, but is explained by pipelining. Increasing floating-point performance with problem size causes the fractional

communication overhead to increase. Our simple scaling model predicts improved fractional communication overheads for problem sizes greater than $n = 64$.

In Fig. 6, we compare the performance of our solution of the time-dependent Dirac equation on the iPSC/860 with its performance on two other computers to which we have access; a Cray-2 supercomputer and an IBM RS/6000 320H workstation. In computing the floating-point performance of the i860 for the purposes of this comparison, we use the overall time T_{hdprd} for Eq. (17) without factoring the communication. We feel this give a reasonable performance comparison between sequential and parallel machines as in Fig. 6, floating-point performance can be considered proportional to CPU time. We optimize our implementation of Eq. (17) on the Cray-2 machine using the cf77 Fortran compiler with default vectorization, loop unrolling, and no autotasking. For the IBM workstation, we use the IBM AIX XL Fortran Compiler/6000 version 2.2 with full optimization. The performance of the IBM workstation decreases with the problem size because of cache memory effects.

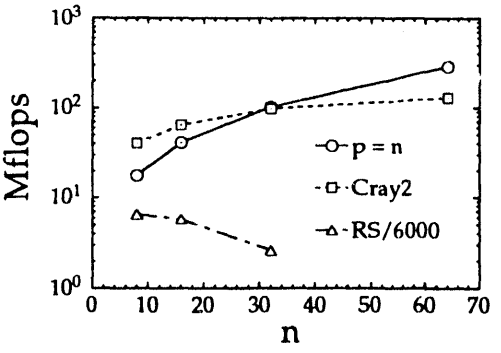

Figure 6: Plotted is a comparison of the performance of implementations of Eq. (17) on the Intel iPSC/860 with $p = n$ processors, on a Cray-2, and on an IBM RS/6000 320H.

Acknowledgements

This research was sponsored in part by the U.S. Department of Energy (DOE) under contract No. DE-AC05-84OR21400 managed by Martin Marietta Energy Systems, Inc., under contract No. DE-FG05-87ER40376 with Vanderbilt University, and by a DOE Graduate Research Participation Fellowship through Oak Ridge Associated Universities. The numerical calculations were carried out on the Intel iPSC/860 hypercube multicomputer at the Oak Ridge National Laboratory, and the CRAY-2 supercomputers at the National Energy Research Supercomputer Center (NERSC) at Lawrence Livermore National Laboratory, and the National Center for Supercomputing Applications (NCSA) in Illinois.

References

[1] J. C. Wells, V. E. Oberacker, S. A. Umar, C. Bottcher, M. R. Strayer, J.-S. Wu, and G. Plunien, *Phys. Rev. A* **45** (1992) 6296.

[2] M. R. Strayer, C. Bottcher, V. E. Oberacker, and A. S. Umar, *Phys. Rev. A* **41** (1990) 1399.

[3] C. Bottcher and M. R. Strayer, *Ann. Phys. (N.Y.)* **175** (1987) 64; A.S. Umar, J. Wu, M. R. Strayer, and C. Bottcher, *J. Comp. Phys.* **93** (1991) 426.

[4] C. Bottcher, M.R. Strayer, A.S. Umar and P.G. Reinhard, *Phys. Rev. A* **40** (1989) 4182.

[5] J. Cullum, R. Willoughby, *Lanczos Algorithms for Large Symmetric Eigenvalue Computations, Vol. I* (Birkhäuser, Boston, 1985).

[6] B. Parlett, *The Symmetric Eigenvalue Problem* (Prentice-Hall, Englewood Cliffs NJ, 1980).

[7] J. C. Wells, V. E. Oberacker, S. A. Umar, C. Bottcher, M. R. Strayer, J.-S. Wu, J. Drake, and R. Flanery *Int. J. Mod. Phys. C* vol.4, no.3 (1993).

[8] A. S. Umar and M. R. Strayer, *Comput. Phys. Commun.* **63** (1991) 179.

[9] C. Bottcher, G. J. Bottrell, and M. R. Strayer, *Comput. Phys. Commun.* **63** (1991) 63.

[10] G. Fox, M. Johnson, G. Lyzenga, S. Otto, J. Salmon, and D. Walker, *Solving Problems on Concurrent Processors, Vol. I* (Prentice-Hall, Englewood Cliffs, 1988), p. 261.

[11] T. H. Dunigan, Oak Ridge National Laboratory Technical Report, Oak Ridge, Tennessee, ORNL Report No. ORNL/TM-11491 1990 (unpublished).

DATE
FILMED
3/9/94
END