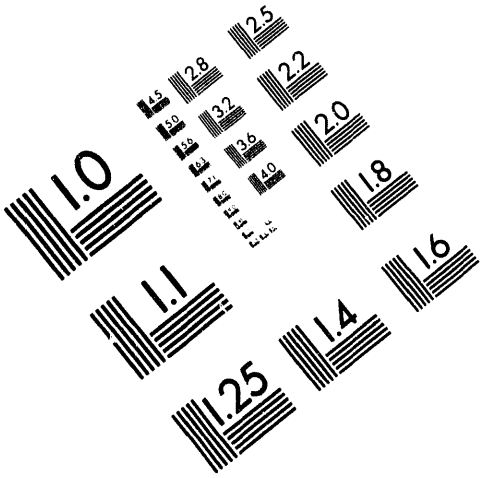
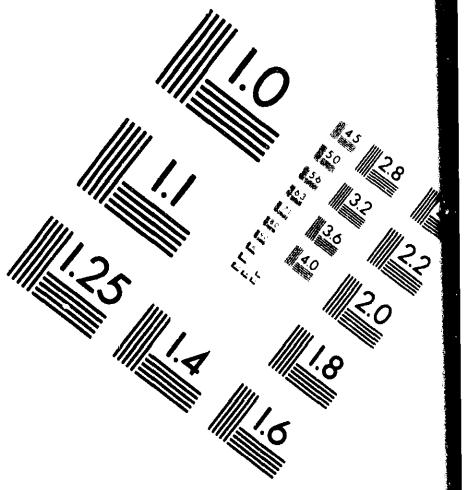
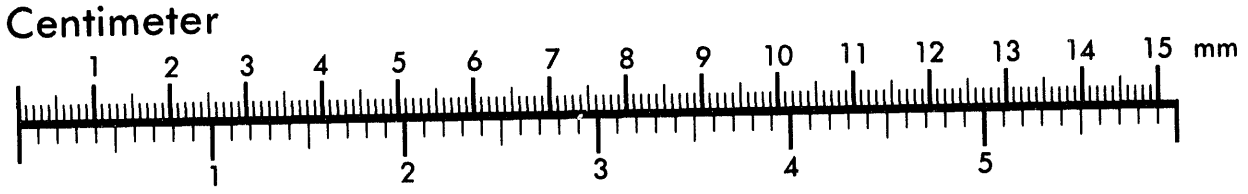
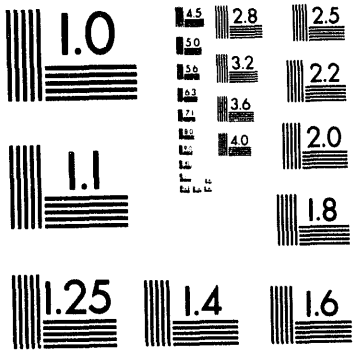
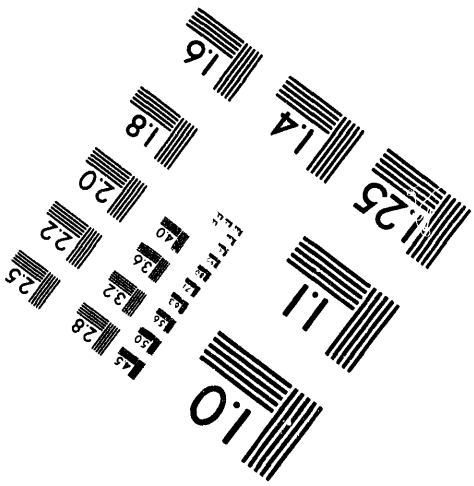


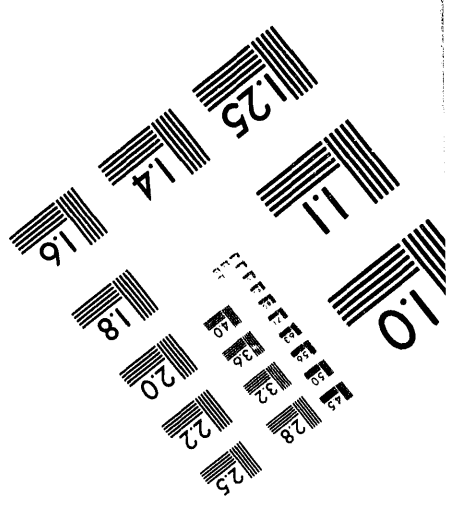
Association for Information and Image Management
1100 Wayne Avenue, Suite 1100
Silver Spring, Maryland 20910
301/587-8202











MANUFACTURED TO AIIM STANDARDS
BY APPLIED IMAGE, INC.

1 of 1

CONF-941118--7

STATIC AND DYNAMIC NUCLEAR MANY-BODY DESCRIPTIONS ON PARALLEL ARCHITECTURES

C. R. Chinn, A. S. Umar
M. R. Strayer, and M. Vallières

to be published in Proceedings of

Supercomputing '94

Washington, D.C.
November 14-18, 1994

Static and Dynamic Nuclear Many-Body Descriptions On Parallel Architectures

C. R. Chinn[†,2], A. S. Umar[1,2], M. R. Strayer[1] and M. Vallières[3]

[1]Center for Computationally Intensive Physics
Physics Division, Oak Ridge National Laboratory, Oak Ridge, TN 37831
[2]Department of Physics & Astronomy, Vanderbilt University
Nashville, TN 37235
[3]Department of Physics & Atmospheric Science, Drexel University,
Philadelphia, PA 19104

[†] Speaker, e-mail address: chinn@compsci.cas.vanderbilt.edu
(615)343-1706, (615)352-9501; FAX: (615)343-7263

April 1, 1994

Abstract

Numerical methods used to solve the system of stiff, nonlinear partial differential equations in static Hartree-Fock descriptions of many-body nuclear systems are presented along with an extension of the time dependent Hartree-Fock solution to linear response theory. A full three dimensional representation is obtained by using a collocation basis spline spatial lattice. Numerical procedures used for parallel applications are discussed. Comparisons are presented for MIMD massively computers, including the Intel Paragon and the iPSC/860 hypercube. Algorithms used to improve communication overhead, especially pipelining the Gramm-Schmidt orthogonalization routine, and difficulties with limited node memory will be discussed.

1 Introduction

Mean field studies of physical systems have provided successful and useful fundamental descriptions of a variety of atomic and nuclear many-body phenomena [1, 2, 3, 4] and promises to provide an important development in future understandings in nuclear astrophysics.

From a numerical point of view, recent new techniques have been developed, which provide a much more sophisticated desciption of Hartree-Fock mean field calculations. In particular, equations of motion were obtained via the variation of the lattice representations of the constants of the motion, such as the total energy [5, 6, 7, 8]. In this variation after discretization approach, resulting equations exactly preserve the constants of the motion. Previous calculations have had to rely on alternate basis expansions, such as harmonic oscillators, which are cumbersome, or on coordinate space lattice representations, which are much easier to use, but typically depend on inaccurate finite difference representations of the derivative operator. Here we use a basis spline collocation lattice, where the derivative operators can be accurately represented in a basis expansion, while also taking advantage of the more tractable lattice grid representation.

Due to limitations in computational power, one tended to assume specific symmetric descriptions, such as spherical or cylindrical symmetry, along with z-parity symmetry and no spin degrees of freedom. Time-reversal symmetry was also used to a great degree. With recent new supercomputer technologies such assumptions are no longer required to perform calculations. In this paper a full three dimensional calculation using basis spline techniques is presented, where symmetries such as time-reversal, need not be imposed. Full spin representations of the wave functions are also included. These calculations are very intensive and require Grand Challenge level computing resources.

The same difficulties that exist in the static mean field case apply to the time-dependent calculation. A new method of calculating linear response theory has been developed which relies on a time-dependent evaluation of the linear response equations [9]. This calculation is constructed on top of the static Hartree-Fock calculation, hence the full sophistication that exists in the static case also exists in the dynamic one. Previous random phase approximation [RPA] calculations have limitations similar to those that have plagued mean field calculations [10, 11, 12, 13]. By not imposing constraints, such as spatial and time-reversal symmetries, one can increase greatly the level of sophistication.

The linear response equations are calculated by applying a specific time-dependent per-

turbation to the static Hartree-Fock hamiltonian. The system is then evolved by solving time-dependent-Hartree-Fock [TDHF] equations. The result is fourier transformed to give the total transition amplitude for a specific collective mode.

Both the static and dynamic calculations are solved on a three dimensional collocation basis spline lattice. The same hamiltonian is used in both the static and dynamic calculations, providing a complete consistency, which historically has not been the usual case. Eventually we would like to pursue linear response studies of $\beta-$decay calculations of exotic nuclei.

In Section 2 the static Hartree-Formalism is presented in brief. A numerical discussion of the numerical discretization of the mean field calculations, specifically the basis spline collocation lattice is given in Section 3. A short discussion of the time-dependent evaluation of linear response theory is in Section 4. A detailed description of the parallel implementation is presented in Section 5, especially the pipelining of the Gramm-Schmidt orthogonalization routine. Time studies are given in Sections 6 and 7 for the static and dynamic calculations, respectively, followed by a Conclusion.

2 Static Hartree-Fock Formalism

2.1 Continuous Equations

The details of the derivation of the Hartree-Fock equations can be found in [5, 6, 7, 8, 9]. The zero-range skyrme force is used to represent the effective two-body interaction in both the static Hartree-Fock and the dynamic linear response calculations. The expectation value of the skyrme hamiltonian is represented in terms of the energy density.

$$E = \langle\phi|\widehat{H}|\phi\rangle = \int d^3r\ \mathcal{H}(r), \tag{1}$$

Using the variational principle we take the functional derivative of the Hartree-Fock energy with respect to the wave function, χ_α:

$$\delta E = \delta \int d^3r \mathcal{H}(r). \tag{2}$$

A coupled set of non-linear partial differential eigenvalue equations is then obtained:

$$\mathrm{h}\chi_\alpha = \epsilon_\alpha \chi_\alpha\ , \tag{3}$$

where χ_α is a two-component vector (spinor)

$$\chi_\alpha = \begin{pmatrix} \chi_\alpha^+ \\ \chi_\alpha^- \end{pmatrix} . \tag{4}$$

The hamiltonian $\mathbf{h}$ has the following form (using natural units $\hbar = 1$, $c = 1$, $m = 1$):

$$\begin{aligned} \mathbf{h} &= -\frac{1}{2}\nabla^2 + W(\rho, \tau, \mathbf{j}, \mathbf{J}) , \\ W &= V_N(\mathbf{r}) + V_C(\mathbf{r}) , \end{aligned} \tag{5}$$

where V_N is the nuclear potential depending on various currents and densities, which in turn depend on the states χ_α. The Coulomb interaction, V_C, requires the solution of the Poisson equation in three-dimensional geometry

$$\nabla^2 V_C(\mathbf{r}) = -4\pi e^2 \rho(\mathbf{r}) . \tag{6}$$

As can be seen from above the solution of the system of equations (3) has to be done self-consistently and an accurate solution requires a good representation of various derivatives of the states χ_α. Currently, most HF and TDHF calculations are performed using finite difference lattice techniques. It is desirable to investigate higher-order interpolation methods which result in the improvement of the overall accuracy and reduction in the total number of lattice points. The lattice solution of differential equations on a discretized mesh of independent variables may be viewed to proceed in two steps: (1) obtain a discrete representation of the functions and operators on the lattice. (2) solve the resulting lattice equations using iterative techniques. Step (1) is an interpolation problem for which we could take advantage of the techniques developed using the spline functions [14, 15, 16]. The use of the spline collocation method leads to a matrix-vector representation on the collocation lattice with a metric describing the transformation properties of the collocation lattice.

The static Hartree-Fock solution is calculated using an iterative scheme as outlined below:

1. Guess a set of orthogonal single-particle states
2. Compute the densities
3. Compute the Hartree-Fock potential
4. Solve the Poisson equation
5. Perform an imaginary time step with damping [17, 4]

6. Do a Gramm-Schmidt orthogonalization of all states

7. Repeat beginning at step 2 until convergence

As a convergence criteria we have required the fluctuations in energy

$$\Delta E^2 \equiv \sqrt{< H^2 > - < H >^2} \tag{7}$$

to be less than 10^{-5}. This is a more stringent condition than the simple energy difference between two iterations, which is about 10^{-10} when the fluctuation accuracy is satisfied. The calculation of the HF hamiltonian also requires the evaluation of the Coulomb contribution given by Eq. (6). Details of solving the Poisson equations using the splines are given in Refs. [15, 16].

3 Nummerical Discretization

3.1 Collocation Basis Splines

An Mth order spline function denoted by B_i^M is constructed from piecewise continuous polynomials up to order $M-1$. The set of points or *knots* $\{x_i\}$ consists of the points where the spline functions are joined continuously up to the $(M-2)$ derivative. The basis spline functions have minimal support in that the ith spline functions is nonzero only in the interval (x_i, x_{i+M}), where the spline function, B_i^M, is labelled by the first knot. For the space containing the $N+1$ knots in one dimension, there must be M nonzero spline functions in each interval, hence $N+2M-1$ total spline functions make up the full basis, where $M-1$ spline functions extend beyond each boundary.

A function, $f(x)$ continuous in the interval (x_{min}, x_{max}) is expanded in terms of the spline basis functions:

$$f(x) = \sum_i^{N+2M-1} B_i^M(x) c^i. \tag{8}$$

The expansion coefficients, c^i are derived from $f(x)$ by evaluating $f(x)$ at a specific set of points called *collocation points*, $\{x'_\alpha\}$. There are various ways of choosing the $\{x'_\alpha\}$. For odd order splines we have chosen the collocation points to lie at the center of each knot interval within the range (x_{min}, x_{max}).

$$\begin{aligned} x_{min} &= x_M, \\ x'_\alpha &= \frac{x_{i+M-1} + x_{i+M}}{2}, \quad i = \alpha, \quad \forall \alpha = 1, ..., N \end{aligned}$$

By evaluating eq. (8) at $\{x'_\alpha\}$, a set of linear equations are constructed which constrain the coefficients, c^i:

$$f(x'_\alpha) = \sum_i^{N+2M-1} B_i^M(x'_\alpha) c^i. \tag{9}$$

Since there are $N + 2M - 1$ unknown coefficients and N points x'_α, it is necessary to introduce $2M - 1$ additional constraining equations as boundary conditions. One can also impose periodic boundary conditions. Combining the functions $B_i^M(x'_\alpha)$ and the boundary conditions into a square invertible matrix, $\mathbf{B}$, the coefficients, c^i can be expressed as:

$$c^i = \sum_\alpha \left[\mathbf{B}^{-1}\right]^{i\alpha} f_\alpha, \tag{10}$$

where $f_\alpha \equiv f(x'_\alpha)$ is the collocation representation of the function, $f(x)$.

Consider the action of an operator upon a function:

$$\hat{O} f(x) = \sum_i^{N+2M-1} \left[\hat{O} B_i^M(x)\right] c^i. \tag{11}$$

If we evaluate the above expression at the collocation points and substitute in eq. (10) for the c^i the following is obtained.

$$\begin{aligned} \hat{O} f(x'_\alpha) &= \sum_i^{N+2M-1} \left[\hat{O} B_i^M(x'_\alpha)\right] \sum_\beta \left[\mathbf{B}^{-1}\right]^{i\beta} f_\beta \\ \hat{O} f_\alpha &= \sum_\beta \hat{O}_\alpha^\beta f_\beta, \end{aligned} \tag{12}$$

where now the quantity, $\hat{O}_\alpha^\beta$ is the collocation representation of the operator, $\hat{O}$ on the lattice.

$$\hat{O}_\alpha^\beta \equiv \sum_i^{N+2M-1} \left[\hat{O} B_i^M(x'_\alpha)\right] \left[\mathbf{B}^{-1}\right]^{i\beta} \tag{13}$$

Note that the representation of $\hat{O}_\alpha^\beta$ is not a sparse matrix.

The function, $f(x)$, and the operator, $\hat{O}$, can both be represented on a lattice, *i.e.* the collocation points, through the use of the collocation basis spline method. This holds true for gradient operators, where the gradient of the basis spline functions is required. The basis spline functions, $B_i^M(x)$ and their derivatives, $\frac{\partial^n B_i^M(x)}{\partial x^n}$ can be evaluated at the collocation points using iterative techniques. Through similar methods one can obtain the appropriate integration weights. For more details on the collocation basis spline method please see Ref. [15].

3.2 HF Equations in Collocation Space

In order to obtain a set of lattice equations which preserve the conservation laws associated with the continuous equations, it is essential to develop a modified variational approach. This goal is achieved by performing a variation to the discretized form of a conserved quantity, i.e. total energy. Consequently, the resulting equations will preserve all of the conserved quantities on the lattice.

$$\sum_{\alpha\beta\gamma} \Delta V_{\alpha\beta\gamma} \left\{ \mathrm{h}(\alpha\beta\gamma) - \epsilon_\lambda \left| \chi_\lambda(\alpha\beta\gamma) \right|^2 \right\} , \tag{14}$$

where indices α, β, and γ denote the lattice points in three-dimensional space, and $\Delta V_{\alpha\beta\gamma}$ is the corresponding infinitesimal volume element. Due to the presence of derivative operators in the hamiltonian, the explicit form of these expressions will depend non-locally on the lattice indices. The general variation, which preserves the properties of the continuous variation, is given by

$$\frac{\delta \boldsymbol{\chi}^*_\mu(\alpha\beta\gamma)}{\delta \boldsymbol{\chi}^*_\lambda(\alpha'\beta'\gamma')} = \frac{1}{\Delta V_{\alpha\beta\gamma}} \delta_{\lambda\mu}\delta_{\alpha'\alpha}\delta_{\beta'\beta}\delta_{\gamma'\gamma} . \tag{15}$$

The details of the discrete variation for the finite-difference case are given in Refs. [5, 6]. The three-dimensional expansion in terms of splines is a simple generalization of Eq. (8)

$$\chi_\alpha(x,y,z) = \sum_{ijk} c_\alpha^{ijk} B_i(x) B_j(y) B_k(z) . \tag{16}$$

The knots and collocation points for each coordinate can be different. With the appropriate definition of boundary conditions, all of the discretization techniques discussed in the previous section can be generalized to the three-dimensional space. The details of this procedure are given in Refs. [15, 16].

A typical nonlocal term is illustrated below:

$$\begin{aligned} (\nabla\chi^\pm_\lambda)_{\alpha\beta\gamma} &= \sum_{\alpha'} D^{\alpha'}_\alpha \chi^\pm_\lambda(\alpha'\beta\gamma)\,\hat{\imath} + \sum_{\beta'} D^{\beta'}_\beta \chi^\pm_\lambda(\alpha\beta'\gamma)\,\hat{\jmath} \\ &+ \sum_{\gamma'} D^{\gamma'}_\gamma \chi^\pm_\lambda(\alpha\beta\gamma')\,\hat{k} \end{aligned}$$

where the matrices $\mathbf{D}$ denote the first derivative matrices in x, y, and z directions (they can be different although the notation does not make this obvious) calculated as described in the previous subsection. Finally, the HF equations can be written as matrix-vector equations on the collocation lattice

$$h\chi^\pm_\alpha \longrightarrow \mathbf{h} \cdot \chi^\pm_\alpha . \tag{17}$$

The essence of this construction is that the terms in the single-particle hamiltonian h are matrices in one coordinate and diagonal in others. Therefore, h need not be stored as a full matrix, which allows the handling of very large systems directly in memory.

4 Dynamical Formalism: Linear Response Theory

The linear response equations can be derived from a specific functional perturbation of the TDHF equations [18]. For a detailed discussion and proof please see Ref. [9].

A specific time-dependent perturbing function is added to the static hamiltonian:

$$\widehat{H}_{tot} = \widehat{H} + \widehat{H}_{ex}(t). \tag{18}$$

This external piece is defined as:

$$\begin{aligned} \widehat{H}_{ex}(t) &= \hat{F} f(t) \\ &= \left[\int d^3x \hat{n}(x,t) F(x) \right] f(t), \end{aligned} \tag{19}$$

where $\hat{n}(x,t)$ is the number density operator and the function $F(x)$ is chosen to represent a particular collective mode. The $f(t)$ will be chosen later.

By solving the time-dependent-Hartree-Fock equations and evolving the system in time, detailed information about the collective dynamic modes of the nuclear system can be extracted. It can be shown that in the linear approximation, the total transition probability can be extracted, corresponding to the collective transition represented by the function, $F(\vec{x})$. By calculating the fluctuations in the nuclear density, $\delta\langle\hat{n}(x,t)\rangle$, as a function of time and then fourier transforming the result into frequency space we obtain the following:

$$\begin{aligned} f(\omega)S(\omega) &= \int d^3x \delta\langle n(x,\omega)\rangle F^\dagger(x) \\ &= \frac{1}{\hbar} \int d^3x \int d^3x' F^\dagger(\vec{x}) D^R(\vec{x},\vec{x}';\omega) F(\vec{x}') f(\omega), \end{aligned} \tag{20}$$

where $S(\omega)$ is the linear response structure function and $D^R(\vec{x},\vec{x}';\omega)$ is the retarded density correlation function. By taking the imaginary part of $S(\omega)$ the total transition probability associated with $F(\vec{x})$ is obtained.

$$IM\left[S(\omega)\right] = -\frac{\pi}{\hbar} \sum_n \left| \int d^3x' \langle \psi_n | \hat{n}(\vec{x}',\omega) | \psi_0 \rangle F(\vec{x}') \right|^2 \delta\left(\omega - \frac{E_n - E_0}{\hbar}\right), \qquad E_n \geq E_0. \tag{21}$$

The structure function is evaluated using a time-dependent perturbative technique. By constructing an explicit form for $f(t)$, one can solve the time evolution of the Hartree-Fock

system using $f(t)$ and $F(\vec{x})$ as perturbing functions. In our case $f(t)$ is chosen to be a Gaussian of the following form:

$$\begin{aligned} f(t) &= \varepsilon e^{-\frac{\alpha}{2}t^2}, \qquad t \geq t_0 \\ f(\omega) &= \varepsilon\sqrt{\frac{2\pi}{\alpha}}e^{-\frac{\omega^2}{2\alpha}}, \end{aligned} \tag{22}$$

where ε is some small number ($\sim 10^{-5}$), chosen such that we are in the linear regime. The parameter, α, is set to be 1.0 c/fm, which allows for a reasonable perturbation of collective energies up to $\approx$ 150 MeV.

In practice, to evaluate the TDHF equations the time-evolution operator is used to evolve the system.

$$U(t, t_0) = T\left[e^{-\frac{i}{\hbar}\int_{t_0}^{t} dt' \widehat{H}_{tot}(t')}\right], \tag{23}$$

where T [...] denotes time-ordering. Using infinitesimal time increments, the time-evolution operator is approximated by

$$\begin{aligned} U(t_{n+1}, t_n) &= e^{-\frac{i}{\hbar}\int_{t_n}^{t_{n+1}} dt' \widehat{H}_{tot}(t')} \\ &\approx e^{-\frac{i}{\hbar}\Delta t \widehat{H}_{tot}(t_n + \frac{\Delta t}{2})} \\ &\approx 1 + \sum_{k=1}^{N}\left[\frac{\left(-\frac{i}{\hbar}\Delta t \widehat{H}_{tot}\right)^k}{k!}\right], \end{aligned} \tag{24}$$

where the quantity $\left(\widehat{H}_{tot}\right)^k$ is evaluated by repeated operations of $\widehat{H}_{tot}$ upon the wave functions. Typically the value of N, the maximum number of applications of $\widehat{H}_{tot}$ for a given iteration and wave function, will be about 4 or 5.

The procedure is to then choose a particular form for $F(\vec{x})$, using eq. (22) for $f(t)$, and time evolve the system using eq. (24). The Fourier transform in time of the result then gives us $f(\omega)S(\omega)$, from which the linear response structure function of the system is extracted.

5 Parallel Implementation

In this section we discuss the details of implementing the lattice representation of the Hartree-Fock equations on the Paragon XPS5 and XPS35, and Intel iPSC/860 hypercube supercomputers at Oak Ridge National Laboratory. These machines are distributed memory, multiple instruction multiple data (MIMD) computers. The Intel iPSC/860 has 128 nodes with 8 MB of memory per node and a peak rating of 60 Mflops per node leading to a 7.6 Gflop

aggregate speed; on the XPS/5 and XPS/35 the peak rating per node is 75 Mflops leading to aggregate speeds of approximately 5 Gflops and 38 Gflops, respectively. Among other differences, iPSC/860 is a hypercube architecture whereas the Paragon is a 2D mesh. The peak internode communication speed of the iPSC/860 is 2.8MB/sec and Paragon is 200 MB/sec. The nodes are connected according to a binary interconnection scheme.

As with most parallel implementations we face the problem of limited memory per node and the optimization of the algorithms to minimize the communication among nodes. It was realized that distribution of the Hilbert space rather than the distribution of the spatial dimensions over the nodes is by far the preferred mode of operation. By placing a subset of the Hilbert space, *i.e.* some of the single particle wave functions, on each node and having all of the spatial operations occur locally on each node, the communication overhead is basically limited to two major operations, which will be discussed below. In general it was found that the best performance is obtained by placing one nucleon wave function of a given isospin on each node.

5.1 Reading and Distribution of Input

The code is set up in a hostless structure, except for i/o to various files. The initial input information is read by one node, designated as node '0', where the information is placed into buffer arrays and broadcasted to the other nodes using the following subroutine:

```
call bcast(iarch,buf,mbytes,0,mtype) .
```

Here *iarch* chooses the appropriate parallel architecture (Paragon or iPSC/860), *buf* contains the real input, *mbytes* is the length of the message in bytes, the "0" corresponds to the originator node of the broadcast, and *mtype* is an integer tag that is incremented every time an i/o operation is performed (tags the many messages being sent between nodes and is used in order of arrival/departure). For output, information is passed to node '0', which then sorts the information and outputs it.

5.2 Gramm-Schmidt Orthonormalization

The Gramm-Schmidt procedure used to orthogonalize the single particle wave functions can be summarized in the following equation:

$$\psi_i'(\vec{x},\tau) \;=\; \psi_i(\vec{x},\tau) - \sum_{j=1}^{i-1} \psi_j(\vec{x},\tau) \frac{\langle \psi_j(\tau)|\psi_i(\tau)\rangle}{\langle \psi_j(\tau)|\psi_j(\tau)\rangle}\,, \qquad i \geq 2 \tag{25}$$

$$\psi_i'(\vec{x},\tau) \;=\; \psi_i'(\vec{x},\tau)/\sqrt{\langle\psi_i'(\tau)|\psi_i'(\tau)\rangle},$$

where τ labels the isospin index, distinguishing between the proton and neutron states, and the vector $\vec{x}$ represents the grid collocation lattice in Cartesian coordinates. The matrix element $\langle\psi_j(\tau)|\psi_i(\tau)\rangle$ is defined as:

$$\langle\psi_j(\tau)|\psi_i(\tau)\rangle \equiv \int d\vec{x}'\psi_j^\dagger(\vec{x}',\tau)\psi_i(\vec{x}',\tau). \tag{26}$$

On a distributed memory parallel machine difficulties arise since the wave functions are spread out over the nodes of the computer, hence the wave function vectors must be passed between the nodes during the orthogonalization process. This creates a large communications overhead since the wave function vectors are dimensioned by the three dimensional collocation lattice and are therefore very large ($\sim 0.25 - 0.5$ Mbytes). The procedure used here is outlined in the flow chart in Fig. 1, where each node, defined locally by the variable, *me*, executes this sequence. Initially on each node, the local wave functions, ψ, are normalized. There is an outer loop, 500, which loops over the node number, where *np* is the total number of nodes. For the one node, *me* = *inode*, the local ψ's are orthogonalized via eq. (25). Here τ is the isospin index, which loops over the proton and neutron states. In many cases time-reversal symmetry is assumed (*itim* = 1), where only half of the total number of states need be explicitly considered. In this case it is still necessary to orthogonalize the wave functions with respect to the time-reversed states:

$$\psi_t(\vec{x},\tau) \equiv \hat{T}\psi(\vec{x},\tau), \tag{27}$$

where the operator $\hat{T}$ is the time-reversal operator. At this time the orthogonalized state is then normalized with a call to *psnorm*. Not only are normalized states desired in the end, but for subsequent orthogonalizations the Gramm-Schmidt procedure in Fig. 1 assumes normalized states. For loop 200, the array $npnu(inode+1,\tau)$ contains the number of wave functions on node *inode* with isospin τ. The state, ψ_{j_1}, on node *inode* is placed into the vector, ψ' and then broadcast to all of the other nodes. For the nodes, *me* > *inode*, the local wave functions, ψ, are orthogonalized with respect to ψ' via eq. (25). Again the time-reversal case can also be considered. For the last node it is not necessary to broadcast the local ψ's. At the end all of the wave functions are already normalized on their respective local nodes.

To optimize the Gramm-Schmidt procedure for use on the Paragon several modifications were made. One change alluded to earlier is to place the neutron and proton states on non-overlapping sets of nodes. The ideal situation is to place either one neutron or one proton

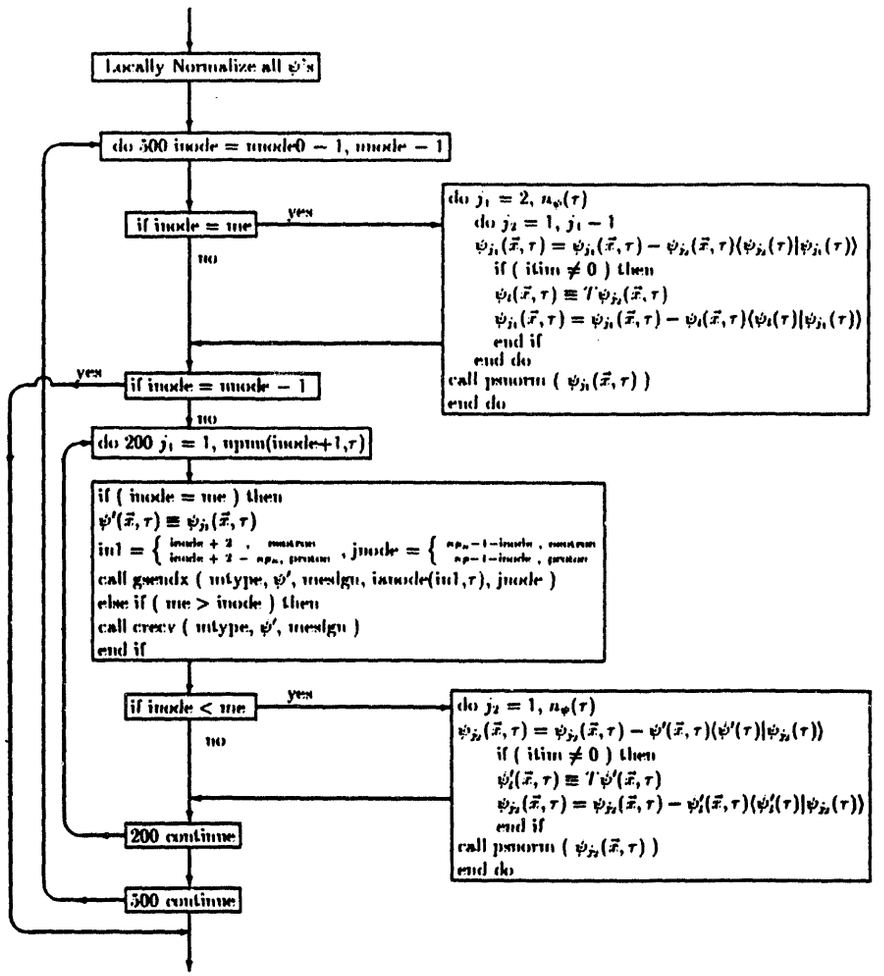


Figure 1: A Schematic diagram outlining the parallel Gramm-Schmidt procedure used in the code

state only, on each node. The Gramm-Schmidt procedure would then proceed among the neutron and proton states separately with no communication between the two sets during the orthogonalization process. In Fig. 1, the index τ is set to be 1 in the neutron sector and 2 in the proton sector. The neutron states are placed in nodes, $0, 1, \ldots, np(1) - 1$, while the proton states are placed into nodes, $np(1), np(1) + 1, \ldots, np(1) + np(2) - 1$.

A second modification involved using the Paragon communication, routine *gsendx*, in place of using *bcast*. It is clear that not all of the nodes need to receive the broadcasted wave function vectors, especially with the separate neutron and proton sectors. The routine, *gsendx* sends a vector to a specific set of destination nodes, defined in an integer array in a semi-global operation. The simplest way to do this is to fill an array with all of the destination nodes in either the proton or neutron sector in decreasing node number, and then to call

gsendx with the appropriate number of destination nodes specified. This algorithm is simple, but inefficient. Because the receiving nodes are listed in decreasing order in the integer array *ianode*, the last node in the Gramm-Schmidt sequence receives the broadcasted vector first. For example, if we list the nodes $np_n \rightarrow 0$ for the neutron sector, in the first pass, node 0 via 'gsendx' sends ψ' to nodes, $np(1)-1,\ldots,2,1$ in this order. The next node to send its local state, ψ', is node 1. Node 1 must wait to receive the vector from node 0 before proceeding to orthogonalize its local ψ' with respect to the wave function vector sent by node 0 and then to broadcast the local ψ'. Therefore, in this scenario, node 1 must wait for all of the destination nodes to receive the vector from node 0, before it can proceed. This represents a bottleneck in the communication sequence. A significant improvement in performance is made when the order of the destination nodes in *ianode* is reversed to increasing order. In this case node 0 broadcasts to nodes $1,2,\ldots,np(1)-1$ in this order. Since node 1 is the first node to receive ψ' from node 0, it can immediately proceed to process and broadcast its own local ψ', even before all of the destination nodes have finished receiving the vector sent by node 0. For cases with a large number of states, several nodes can actually be broadcasting their local vectors, simultaneously. The entire procedure remains naturally sequential and orderly.

This sequence of communications was deciphered and verified with the help of the performace monitoring tool **Paragraph**, which was found to be very useful.

5.3 Broadcasts and Global Summations

Here we discuss some of the algorithms used in performing the communication tasks mentioned above. We have already discussed the communication tasks involved in the Gramm-Schmidt process. In addition to Gramm-Schmidt we have to perform global sums for the densities and currents, which are used to calculate the hamiltonian. In practice, we treat global sums on the iPSC/860 differently from the Paragon's due to their architectural difference (hypercube versus 2D mesh). For the iPSC/860 the basic algorithm is the broadcast algorithm which ensures that messages are transmitted along routes which do not interfere with others and the communication load is distributed in a balanced way. For the hypercube architecture the neighboring nodes are identified by their Gray codes [19, 20]. This is a binary interconnection scheme where the processors are numbered as decimal numbers, beginning with 0, and arranged such that their binary representation only differ by a single

bit location. To perform broadcast and global sums we have used the *subcube broadcast* algorithm [21]. Below is the algorithm we have used on the iPSC/860:

```
k=1
do i=1,n
   if(me.lt.k) then
      call send(...,me+k)
   elseif(me.lt.2*k) then
      call recv(...)
   end if
   k=2*k
end do
```

where n is the dimension of the cube and *me* denotes the node number. The last argument of the *send* routine is the destination node. On the Paragon the broadcast is done by using the synchronous *csend* and *crecv* routines as follows:

```
if(me.eq.0) then
   call csend(msgtyp,buf,mbytes,-1,mptype)
else
   call crecv(msgtyp,buf,mbytes)
end if
```

where most arguments have been described previously and the -1 asks *csend* to send the message to all nodes except itself.

The generalization of the above broadcast algorithm can be used to perform global summations on the iPSC/860. This is done by first performing a reverse broadcast by starting from the bottom of the broadcast tree and accumulating the results at node 0. Subsequently, node 0 performs a forward broadcast to distribute the result to all nodes. The Fortran code used on the iPSC/860 for backward broadcast is given below:

```
k=np
do i=1,n
   k=k/2
   if(me.lt.k) then
```

```
      call recv(...)
   ...add the received quantity to the resident one....
    elseif(me.lt.2*k) then
      call send(...,me-k)
    end if
  end do
```

where most quantities are defined above and np is the total number of nodes. On the Paragon we again use the resident routine *gdsum* for global double precision summation

```
call gdsum(buf,mbytes,dummy).
```

Of course for 2^n nodes, the algorithms used for the iPSC/860 can also be used on the Paragon. It was found that the iPSC/860 algorithms and the Paragon message passing routines discussed here give very similar timings.

6 Timing Studies for Static Calculations

For timing comparisons executions on several platforms were performed. On the parallel machines the maximum number of nodes possible for each case was used, where one nucleon state was placed on each node (A = # of nucleons = # of nodes), unless otherwise stated. Nuclei with equal number of protons, Z, and neutrons, N, were calculated, where in this situation the calculation and communication time for the proton and neutron sectors will be essentially equivalent. For the vast majority of nuclei, N and Z are not equal, and hence the computational burdens of the 2 sectors will be unequal, where basically the time difference will correspond to the different amount of time spent within the Gramm-Schmidt procedure.

Static Hartree-Fock calculations were performed for ^{16}O, ^{32}S, ^{64}Ge and ^{128}Gd nuclei, where N = Z in these cases. The size of the basis spline collocation lattice was also varied, where we studied grids of 16^3, 20^3, 24^3 and 26^3 lattices. For timing comparisons the code was run for 100 iterations, although typical calculations will require about 500 iterations or more.

Due to memory limitations we were unable to perform calculations with grids larger than 22^3 on the iPSC/860. Although the Paragon is a virtual machine, if there is any significant swapping of memory, then the performance on the Paragon deteriorates dramatically. The Paragon models, XPS5 and XPS35 at ORNL presently have 16 Mbytes/node memory, with

about 10 Mbytes available for computational use. The xps35 will be expanded in the near future to 32 Mbytes/node. Eventually ORNL will obtain a machine with 64 Mbytes/node, which will eventually be expanded to 128 Mbytes/node. The increased memory will be very useful for our purposes, because larger lattices will be required for large exotic nuclei. With present memory limitations on the Paragon we are essentially constrained to a maximum of 24^3 lattices sizes.

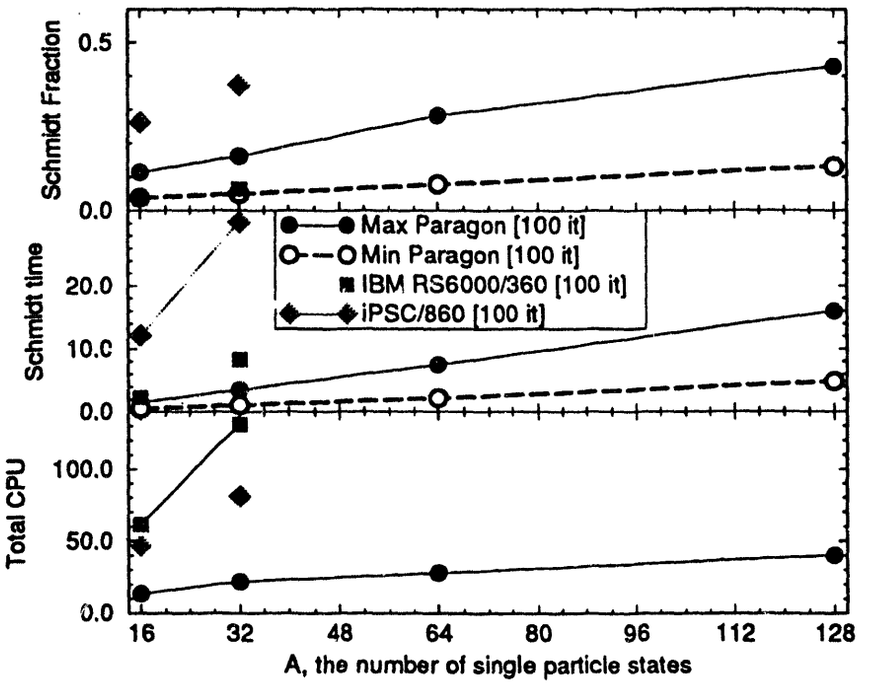


Figure 2: A comparison between the performance of different platforms is shown. The collocation lattice grid is fixed to 20^3 points. The bottom panel shows the total execution time/node in minutes, the middle panel displays the time spent in the Schmidt routine in minutes, while the top panel shows the fraction of time spent in the Schmidt routine.

The total CPU time/node as a function of A is shown in the lower panel of Fig. 2 for a 20^3 collocation lattice. The CPU execution time is retrieved for each node. The 'total CPU time/node', as defined in this paper, is NOT the average execution time per node, but the nodal execution time which is the longest. This would then correspond to the amount of wall clock time required for execution with no time-sharing. The average nodal execution time and the maximum nodal execution time are in general very close due to necessary global synchronizing operations.

In Fig. 2 the timing results are shown for the Paragon, iPSC/860 and for an IBM RS6000/360 workstation, which has been rated at 22.5 Mflops for double precision For-

tran Linnpack. One can see that the Paragon provides a superior platform in comparison to both the IBM workstation and the iPSC/860.

For large A, the execution time/node appears to increase linearly for $A > 32$, although there seems to be a some fluctuations in the calculated CPU time, probably due to traffic and machine fluctuations. For example, for $A = 128$ and a 20^3 lattice we obtained a CPU time/node of 38.48 minutes in one run and 40.23 minutes in another.

The time used for communication resides basically in 2 places. The global double precision sums described in section 3.5 take about 10% of the total nodal CPU execution time. This 10% overhead remains consistent when varying the size of the lattice and the number of nodes or wave functions, and even in comparison between the iPSC/860 and the Paragon's. For cases where the number of nodes $= A/2$, then the global sums take about $6 - 7\%$ of the total nodal execution time.

The largest amount of communication time is used during the Gramm-Schmidt orthogonalization procedure. This procedure cannot be executed in parallel and involves the passing of large messages between the nodes. Since this procedure involves both computation and communication which are in general performed sequentially, the timing of the whole Schmidt procedure will be considered. It is difficult to separate out the communication time, since some nodes only send messages, while other nodes will only receive messages and most nodes will do a combination of both. Because of the nonparallel nature of Gramm-Schmidt, for both the neutron and proton sectors, all of the other nodes in the sector must first finish and pass their local wave vectors before the last node in the procedure can process its local wave vector. Hence the the last node will have some idle time, while the first node will complete its Gramm-Schmidt procedure quickly and take much less time than the last node. For $N \neq Z$ nuclei the sector with more nucleons may take much more time than the other sector.

In the middle panel of Fig. 2 the maximum and minimum nodal time spent within the Schmidt routine is shown to illustrate the nonparallel nature of this procedure. As the number of nodes, A, is increased the maximum and minimum Schmidt execution time increases linearly. This scaling feature will be discussed in more detail later in this section. In the top panel of Fig. 2 the fraction of the total nodal execution time is shown. These fractions are obtained by dividing the times given in the middle panel by the corresponding nodal execution times given in the lower panel. As A increases, it is clear that the Paragon

is much more efficient in communication in comparison with the iPSC/860. On the Paragon for large A the maximum fraction of time spent in the Schmidt is about $40 - 50\%$, thus creating a significant overhead for the program.

In the Fig. 3 a comparison is made between runs using A nodes, where one proton or neutron state resides on each node, and runs with $A/2$ nodes, where either 2 proton or neutron states are on each node. Also shown are runs, where *bcast* is used in an old version of the Schmidt routine, where on each node there is one proton and one neutron state with $A/2$ nodes.

In comparison with the A node calculation represented by the circles, the MINIMUM nodal time spent in the new Schmidt routine is the same as the $A/2$ node calculation using the new Schmidt routine. To make a comparison with the MAXIMUM nodal time in Schmidt we need to consider the amount of communication and computation involved in the time the last node in each sector must spend in the Schmidt routine. First comparing communication time, given N neutrons, for the A node case the last node must perform or wait for $N-1$ sends and receives. For the $A/2$ node case the last node is involved with $2 \times \left(\frac{N}{2} - 1\right) = N - 2$ sends and receives. So even though it would seem that with fewer nodes and a sequential process, there should be less communication, this is not the case. The actual communication time involved with the last node is essentially the same for these two cases. This is reflected in Fig. 3, except for the $A = 128$ point, which is probably high due to fluctuations in traffic on the machine. Since the amount of communication involving the last node is proportional to N, the time spent within the Schmidt routine should scale linearly, which is precisely the pattern observed in Fig. 2.

The Gramm-Schmidt procedure in a strictly sequential sense does not increase linearly, but geometrically. The behavior seen in Fig. 3 can be understood with the following discussion. As described in Section 5.2 the modified new Gramm-Schmidt routine uses *gsendx* in a pipeline fashion. By having a node broadcast the local wave vector in a particular order, i.e. to the next node in the orthogonalizing sequence, one can make the communication more efficient and reduce the idle time. For example the following sequence for the new Schmidt routine can be stated as follows: Node 0 computes, then broadcasts to nodes 1, 2, ..., $N-1$. Since node 1 is the next node in the sequence and is the first to receive the wave vector from node 0, node 1 can immediately receive, compute, and then broadcast its local wave vector to nodes, 2, 3, ..., $N-1$. Node 2 then proceeds as well, et cetera. Hence the

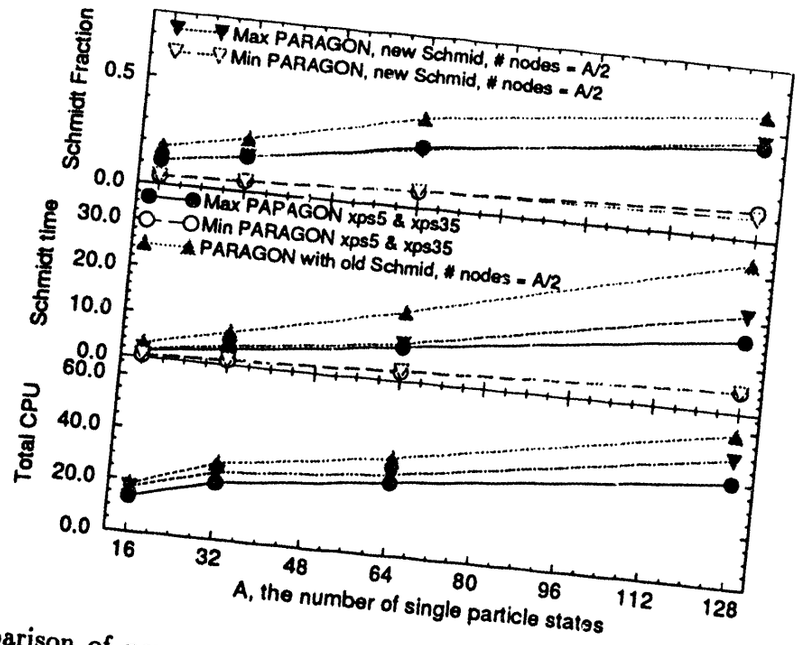


Figure 3: Comparison of new vs. old Gramm-Schmidt routines as a function of A, the number of nucleons. The collocation lattice grid is fixed to 20^3 points. The bottom panel shows the total execution time/node in minutes, the middle panel displays the time spent in the Schmidt routine in minutes, while the top panel shows the fraction of time spent in the Schmidt routine.

broadcasting is performed in a pipeline fashion, since for large N, there can be several wave vectors being broadcasted from several nodes at the same time, all of which will eventually be received by node $N-1$. While for example node 1 is receiving and computing, the other nodes are also receiving and computing at essentially the same time, with some delay. Thus the computations within the Schmidt routine are also performed in a pipeline fashion and hence the computation time should increase linearly with A as well as the communication time.

7 Dynamical Results for Linear Response Theory

Calculations of isovector and isoscalar dipole, octupole and quadrupole collective modes have been calculated for ^{16}O using several parametrizations of the Skyrme interaction. Here results will be shown for axial isoscalar quadrupole collective modes using the skm* Skyrme effective interaction [22]. For skm* the time step used in eq. (24), $\Delta t = 0.4$ fm/c works well and the calculation can be extended to 32768 time steps.

Reasonable results are obtained if the parameter ϵ is chosen to fall in the range $1.0 \times 10^{-5} \leq \epsilon \leq 2 \times 10^{-7}$. By varying the value of ϵ, the amplitude of the time-dependent density fluctuation then scales proportionally to ϵ, thus indicating that we are well within the linear regime of the theory.

The linear response calculations require well converged initial static HF solutions. For ^{16}O it was found that static HF solutions with the energy fluctuation, defined earlier as $\sqrt{\left|\langle \widehat{H}^2 \rangle - \langle \widehat{H} \rangle^2\right|}$, less than about 1.0×10^{-5} provide adequate starting points for the dynamic calculations, although the smaller the energy fluctuation the better.

The dynamic calculations involve using eq. (24) to evolve the system. Since $U(t, t')$ is an unitary operator, the orthonormality of the system is preserved, therefore it is not necessary to re-orthogonalize the solutions after every time-step. This means that the communication intensive Schmidt routine is not needed nor used in the dynamic calculation. The stability of the calculation is checked by testing the preservation of the norm of each wave function. The number of terms in the expansion of the exponent in eq. (24) is determined by requiring the norm to be preserved to a certain accuracy (typically to $\leq 1.0 \times 10^{-8} - 1.0 \times 10^{-10}$).

The time-dependent perturbing part of the hamiltonian is evaluated when the exponent in eq. (22) is greater than some small number, ϵ_{cut}. Since it is not difficult to evaluate the action of the external part of the hamiltonian on the wave function, ϵ_{cut} is chosen to be very small, (1.0×10^{-10}). To allow the fourier transform of $f(t)$ to be evaluated easily, it is necessary to integrate t from $-\infty$ to ∞ and hence we would like the entire Gaussian of the perturbing function, $f(t)$, to be included into the time evolution to the desired accuracy. The parameter t_0 is therefore chosen such that the complete nonzero contribution of the time-dependent perturbation is included. $t_0 = -\Delta t \left(2 + \sqrt{\left|\frac{2 \log \epsilon_{cut}}{\alpha \Delta t}\right|}\right)$.

Pairing can be easily included using the BCS [23] or Lipkin-Nogami [24] prescriptions. These two methods have been included into the static Hartree-Fock calculations and can be easily incorporated into the dynamical calculation. For studies of $\beta-$decay it will be necessary to include pairing, thus producing calculations of responses to quasi-RPA excitation modes.

7.1 Quadrupole Excitation Modes

For the study of the isoscalar quadrupole moment, the perturbing function $F(\vec{x})$, introduced in eq. (19), is chosen to be the mass quadrupole moment, $Q_{20} = 2z^2 - (x^2 + y^2)$. It turns

out that other even multiple modes are also excited at the same time (*i.e.* Q_{40}, Q_{60}, ...). It is therefore possible to study the resonance structure for these other cases, although their transition amplitudes cannot be extracted. The same holds true for the odd multipoles.

The quadrupole collective resonances are calculated for ^{16}O using various Skyrme force parametrizations for comparisons. Although smaller grid sizes are appropriate for static calculations, for the dynamic time-evolution to be able to proceed to large times, it was found that there must be at least a 20^3 lattice. For a 18^3 lattice the time evolution broke down at about 14000 iterations, while for 20^3 lattice points, the calculation was able to proceed to 32768 time steps.

In Fig. 4 the time-dependent evaluation of the multiple moment defined as:

$$\langle \hat{Q}_{20}(t) \rangle = \langle F(x) \rangle = \int d^3x \delta \langle \hat{n}(x,t) \rangle F^{\dagger}(x), \tag{28}$$

is shown for the skm* case. This figure illustrates the periodic character of the calculation, where in this case the smallest oscillation is about 65 fm/c.

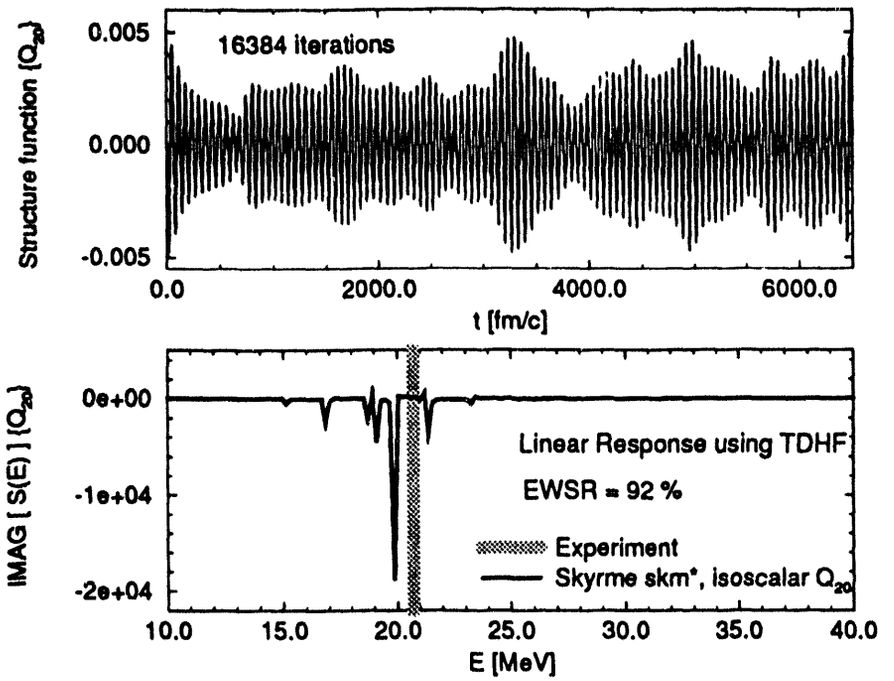


Figure 4: The linear response results using skm*, for the collective axial quadrupole vibrational mode. The upper panel is the time-evolution result, while the lower panel shows the result after fourier transforming into energy space.

A fast fourier transform [FFT] is used to calculate the fourier transform of $\langle \hat{Q}_{20}(t) \rangle$ to give $\langle \hat{Q}_{20}(\omega) \rangle = f(\omega) S_{20}(\omega)$. The time-dependent perturbation function, $f(\omega)$ can be then

easily factored out (20).

The experimental isoscalar quadrupole giant resonance is a broad peak centered about an energy of approximately equal to 20.7 MeV with a width (FWHM) of about 7.5 ± 1 MeV [25]. According to eq. (21) the imaginary part of $S(\omega)$ should be purely negative. As can be seen in the lower panel in Fig. 4, the result here is almost purely negative as theoretically predicted. One constraint is the so-called energy weighted sum rule [EWSR], which provides a confirmation of the accuracy of the response calculation. This sum rule can be shown to depend on the structure of the hamiltonian and can be calculated from the hamiltonian as well as in RPA calculations. It is found that we are able to calculate 92% of the sum rule.

The FFT is designed to give the correct fourier transform when the integral over time encompasses the whole region in which $S(t)$ is nonzero. For other forces the result was not as clean as the result shown in Fig. 4. This is due to the fact that the evolved solution in time does not die off to zero, but maintains a relatively constant strength. This means that the FFT gives only an approximate solution and it may be necessary in some cases to go to larger maximum times.

7.2 Timing for the Dynamic Calculation

Timing comparisons for the dynamic calculation are give in Fig. 5, where the time in minutes is given for executing 100 time steps. The actual runs used many more time steps, so the points in Fig. 5 are normalized to 100 time steps. In the lower panel the total CPU time per node is shown. The two parallel machines are much faster than the sequential workstation, while the Paragon is faster than the iPSC/860 by about a factor of three.

The time-dependent calculation has much less communication overhead than the static calculation, since it is not necessary to perform the Gramm-Schmidt orthogonalization procedure. The only large-scale communication is performed by the global summation routine, *gadd*. In the upper panel the times in minutes for the global sums are given for the parallel machines. For a 20^3 lattice the Paragon is much more efficient, where *gadd* on the iPSC/860 requires about 24% of the total CPU time/node, the corresponding figure for the Paragon is about 7%. For the larger 22^3 lattice grid, the fraction of communication time on the Paragon is about 17%. This surprisingly large increase may be due to some saturation of the communication buffers.

Although the dynamic calculation as a parallel operation is much more efficient than

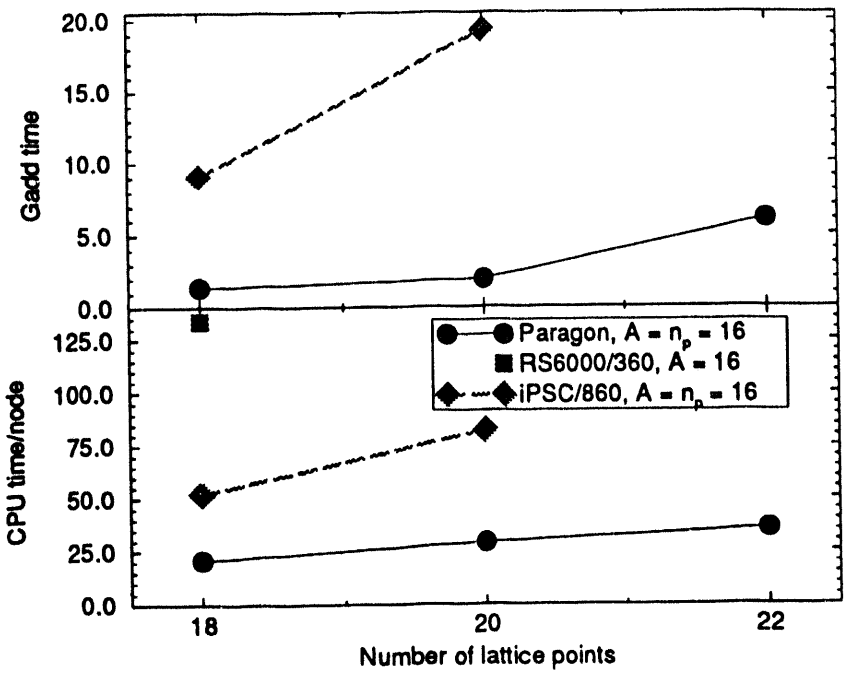


Figure 5: Timing comparisons are given for the dynamic TDHF calculation of linear response theory. The lower panel contains the Total CPU time/node in minutes for 100 time steps of executation. The upper panel contains the time requires for the global summation routine, *gadd.*

the static calculation, the dynamic calculation requires a greater amount of computational resources. Presently, the dynamic calculation has been tested for 16 nodes, and will shortly be expanded to cases involving 32 and 40 nodes. The dynamic calculation requires about 16000 time steps, which corresponds to runs on the order of days. The static calculation requires much fewer iterations. When the Paragon's at Oak Ridge expand to larger memories, we will then be able to proceed to address much large and more exotic nuclei by using much larger collocation lattice spaces. .

8 Conclusions

Massively parallel platforms, such as the iPSC/860 and the Paragon provide a much improved vehicle for performing mean field calculations. Because of the increased computer resources calculations of large complex and exotic nuclear many-body systems can now proceed with a greater sophistication. For the static Hartree-Fock mean field calculation a program, which uses a full three dimensional basis spline collocation lattice, has been developed with no spatial or time-reversal symmetries imposed. This program has been ported to the iPSC/860

and the Paragon. An algorithm was developed, which takes advantage of some of the features of the Paragon to streamline the communication intensive Gramm-Schmidt orthogonalization routine by pipelining the message passing and the computations.

A dynamical extension of the Hartree-Fock mean field theory in the form of time-dependent Hartree-Fock enables us to perform calculations of the linear response of the nucleus. This program is a highly efficient parallel approach, since the time-evolution that is used to perform the calculation involves a unitary operation, which preserves the orthonormality of the many-body system. It is therefore not necessary to perform the Gramm-Schmidt operation and hence there is little necessary communication required in the dynamic calculation. With the future expansion in nodal memory planned for the Paragon computers at Oak Ridge, we should be able to proceed with an active program to investigate further linear response calculations.

Acknowledgements

This research has been supported in part by the U.S. Department of Energy (DOE) Office of Scientific Computing under the High Performance Computing and Communications Program (HPCC), as a Grand Challenge titled the Quantum Structure of Matter, and in part by DOE under contract No. DE-AC05-84OR21400 managed by Martin Marietta Energy Systems, Inc., and under contract No. DE-FG05-87ER40376 with Vanderbilt University. Some of the numerical calculations were carried out on the Intel Paragon and iPSC/860 parallel computers at the Oak Ridge National Laboratory, and Cray computers at the NERSC, Livermore.

References

[1] K. T. R. Davies, K. R. S. Devi, S. E. Koonin, and M. R. Strayer, in *Treatise on Heavy Ion Science*, edited by D. A. Bromley, (Plenum, New York, 1985), Vol.3, page 3.

[2] A. S. Umar and M. R. Strayer, Comp. Phys. Comm. **63**, 179 (1991).

[3] C. Bottcher, G. J. Bottrell, and M. R. Strayer, Comp. Phys. Comm. **63**, 63 (1991).

[4] A. S. Umar, M. R. Strayer, R. Y. Cusson, P.-G. Reinhard, and D. A. Bromley, Phys. Rev. C **32**, 172 (1985).

[5] A. S. Umar, M. R. Strayer, P. -G. Reinhard, K. T. R. Davies, and S. -J. Lee, Phys. Rev. C **40**, 706 (1989).

[6] K. T. R. Davies and S. E. Koonin, Phys. Rev. C **23**, 2042 (1981).

[7] P. Hoodbhoy and J. W. Negele, Nucl. Phys. **A288**, 23 (1977).

[8] S. E. Koonin, K. T. R. Davies, V. Maruhn-Rezwani, H. Feldmeier, S. J. Krieger, and J. W. Negele, Phys. Rev. C **15**, 1359 (1977).

[9] 'Time-Dependent Evaluation of Linear Response Theory', C. R. Chinn, A. S. Umar and M. R. Strayer, To be submitted.

[10] G. F. Bertsch and S. F. Tsai, Phys. Reports **18**, 125 (1975).

[11] S. Krewald, V. Klemt, J. Speth and A. Faessler, Nucl. Phys. **A281**, 166 (1977).

[12] K. F. Liu and G. E. Brown, Nucl. Phys. **A265**, 385 (1976).

[13] N. Van Giai and H. Sagawa, Nucl. Phys. **A371**, 1 (1981).

[14] C. De Boor, *Practical Guide to Splines*, (Springer-Verlag, New York, 1978).

[15] A. S. Umar, J. Wu, M. R. Strayer, and C. Bottcher, J. Comp. Phys. **93**, 426 (1991);

[16] C. Bottcher and M. R. Strayer, Ann. of Phys. **175**, 64 (1987).

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

[18] A. L. Fetter and J. D. Walecka, Quantum Theory of Many-Particle Systems, (St. Louis: McGraw-Hill Book Co., 1971).

[19] 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.

[20] T.F. Chan and Y. Saad, *IEEE Trans. Computers, C-35* (1986) 969.

[21] G. C. Fox, S. W. Otto, and A. J. G. Hey, *Parallel Computing* 4 (1987) 17; and Ref. 27, p. 244.

[22] J. Bartelm, O. Quentin, M. Brack, C. Guet and H. B. Hakansson, Nucl. Phys. **A 386**, 79 (1982).

[23] P. Ring & P. Schuck, The Nuclear Many-Body Problem, (N.Y.: Springer-Verlag 1980).

[24] H. C. Pradhan, Y. Nogami and J. Law, Nucl. Phys. **A201**, 357 (1973); Y. Nogami, Phys. Rev. **134**, B313 (1964).

[25] K. T. Knöpfle, G. J. Wagner, H. Breuer, M. Rogge, C. Mayer-Böricke, Phys. Rev. Lett. **35**, 779 (1975).

DATE
FILMED
10/21/94

END