ARTICLE IN PRESS

Computer Physics Communications ▮ (▮▮▮▮) ▮▮▮–▮▮▮

Contents lists available at ScienceDirect

Computer Physics Communications

journal homepage: www.elsevier.com/locate/cpc

A finite-temperature Hartree–Fock code for shell-model Hamiltonians☆

G.F. Bertsch *, J.M. Mehlhaff

Institute for Nuclear Theory and Department of Physics, Box 351560, University of Washington, Seattle, WA 98915, USA

ARTICLE INFO

Article history:
Received 8 February 2016
Received in revised form
17 June 2016
Accepted 27 June 2016
Available online xxxx

Keywords:
Hartree–Fock
Shell model
Gradient method
Nuclear levels
Nuclear structure

ABSTRACT

The codes `HFgradZ.py` and `HFgradT.py` find axially symmetric minima of a Hartree–Fock energy functional for a Hamiltonian supplied in a shell model basis. The functional to be minimized is the Hartree–Fock energy for zero-temperature properties or the Hartree–Fock grand potential for finite-temperature properties (thermal energy, entropy). The minimization may be subjected to additional constraints besides axial symmetry and nucleon numbers. A single-particle operator can be used to constrain the minimization by adding it to the single-particle Hamiltonian with a Lagrange multiplier. One can also constrain its expectation value in the zero-temperature code. Also the orbital filling can be constrained in the zero-temperature code, fixing the number of nucleons having given K^π quantum numbers. This is particularly useful to resolve near-degeneracies among distinct minima.

Program summary

Program title: HFgradZ.py, HFgradT.py

Catalogue identifier: AFAX_v1_0

Program summary URL: http://cpc.cs.qub.ac.uk/summaries/AFAX_v1_0.html

Program obtainable from: CPC Program Library, Queen's University, Belfast, N. Ireland

Licensing provisions: GNU General Public License, version 3

No. of lines in distributed program, including test data, etc.: 9547

No. of bytes in distributed program, including test data, etc.: 80195

Distribution format: tar.gz

Programming language: Python (2.7).

Computer: PCs.

Operating system: Unix, Apple OSX.

RAM: 10 MBy

Classification: 4.9, 17.22.

External routines: Numpy (1.6)

Nature of problem:

Find Hartree–Fock minima of shell-model Hamiltonians

☆ This paper and its associated computer program are available via the Computer Physics Communication homepage on ScienceDirect (http://www.sciencedirect.com/science/journal/00104655).

* Corresponding author.
E-mail address: bertsch@uw.edu (G.F. Bertsch).

http://dx.doi.org/10.1016/j.cpc.2016.06.023
0010-4655/Published by Elsevier B.V.

Solution method:
Gradient method with a preconditioner
Running time:
A few minutes

Published by Elsevier B.V.

1. Introduction

The usual methods for finding the Hartree–Fock (HF) minima of nuclear Hamiltonians focus on the equations that must be satisfied at the minimum,

$$\frac{d\langle H\rangle}{d\vec{x}} = 0. \tag{1}$$

Here $\langle H\rangle$ is the Hartree–Fock expression for the energy, Eq. (11), and $\vec{x}$ is the set of variational parameters. Convergence problems can easily arise, as documented in Sect. 5.4 of Ref. [1]. They may be overcome by sophisticated iteration schemes such as the Broyden method [2], but we find that the gradient method advocated in Ref. [1] and adopted Refs. [3,4] is simple and effective. The gradient method is implemented in `HFgrad` by constructing the vector $d\langle H\rangle/d\vec{x}$ and using it to guide the iteration process. This is described in Section 3.

There is one important difference in the two codes presented here. The orbital occupation factors in the ground-state code `HFgradZ.py` are either zero or one, specified in advance. In the finite-temperature code `HFgradT.py`, the occupation factors are the usual Fermi functions and they are computed as part of the gradient iteration.

2. Variables

We consider a basis of shell-model wave functions labeled by ℓ, j, m and τ_z and distinguished by an index $i, j, \ldots$. The orbitals are linear combinations of the shell-model wave functions; they are indexed by Greek letters $\kappa, \lambda, \ldots$. The many-body wave function is characterized by an orthogonal matrix U that transforms from the shell basis to the orbital basis

$$|\kappa\rangle = \sum_i U_{\kappa,i}|i\rangle \tag{2}$$

and a diagonal matrix P specifying the occupation factors in the orbital basis

$$P_{\kappa\lambda} = \delta_{\kappa\lambda} f_\kappa. \tag{3}$$

In the zero-temperature code, $f_\kappa = 1$ or 0 depending on whether the orbital is occupied or not, while it the finite temperature code it can vary between these limits.

The fundamental physical quantity associated with the HF solution is the single-particle density matrix ρ, given by

$$\rho = U^T P U. \tag{4}$$

The nominal dimension of the matrix is $N_d = N_b^2$, where N_b is the number of single-particle states in the shell-model basis. However, the restriction in the code to axially symmetric configurations with good parity considerably reduces the number of nonzero terms; the code takes advantage of the symmetry by separating the matrix into blocks.

3. Basic equations

The code treats Hamiltonians that can be represented as a sum of a diagonal one-body operator in Fock space together with a two-body interaction given by its J-coupled matrix elements. The basis states are the shell-model states $|i\rangle = |\tau_{zi}, \pi_i, j_i, m_i\rangle$ where τ_z is the isospin, j is the angular momentum, π is the parity and m is the z-component of angular momentum. The input Hamiltonian may be written

$$\hat{H} = \hat{K} + \hat{v} \tag{5}$$

where

$$\hat{K} = \sum_i \varepsilon_i \hat{a}_i^\dagger \hat{a}_i \tag{6}$$

$$\hat{v} = \sum_{i<j,k<l} v_{ij,kl}\, \hat{a}_i^\dagger \hat{a}_j^\dagger \hat{a}_l \hat{a}_k \tag{7}$$

and

$$v_{ij,kl} = d_{ij} d_{kl} \sum_{JM} (ij|v|kl)_J\, (j_i j_j\, m_i\, m_j | J\, M)(j_k j_l\, m_k\, m_l | J\, M). \tag{8}$$

Here $(ij|v|kl)_J$ are the antisymmetrized J-coupled interaction matrix elements and $(j_i j_j\, m_i\, m_j | J\, M)$ are Clebsch–Gordon recoupling coefficients. The factors $d_{ij} = (1+\delta_{ij})^{-1/2}$ are needed to produce a correct normalization for the operator $\hat{v}$. Only the elements ρ_{ij} with $(\tau_{zi}, \pi_i, m_i) = (\tau_{zj}, \pi_j, m_j)$ are kept in the array representing the density matrix; the imposed symmetries require that other elements are zero. Similarly, only terms that can give nonzero contributions to the interaction energy are kept in the array representing $\hat{v}$. We emphasize that the shell-model Hamiltonians that are treated in the examples here only apply to a limited valence-shell space. The $\hat{v}$ is intended to be an effective interaction taking into account polarization of the core orbitals. In any case, the calculated energies of the codes represent only the part of the total energy associated with the valence particles.

Both the energy and the gradient are computed using the single-particle potential V as an intermediate array. It is defined

$$V_{ik} = \sum_{ijkl} (v_{ij,kl} - v_{ij,lk})\rho_{jl}. \tag{9}$$

The gradient is derived from the single-particle Hamiltonian

$$H^{sp} = K + V, \tag{10}$$

a matrix with nominal dimension $N_b \times N_b$.

Besides ρ, the matrices U, V and H are block-diagonal with the blocks determined by the triplet of quantum numbers (τ_z, π, m).

For convenience the code is split into two driver modules, `HFgradZ.py` for HF at zero temperature and `HFgradT.py` for finite temperature. The zero-temperature code minimizes the HF energy

$$E = \langle H\rangle = \mathrm{Tr} K\rho + \frac{1}{2}\mathrm{Tr}_2 \rho v \rho. \tag{11}$$

Please cite this article in press as: G.F. Bertsch, J.M. Mehlhaff, A finite-temperature Hartree–Fock code for shell-model Hamiltonians, Computer Physics Communications (2016), http://dx.doi.org/10.1016/j.cpc.2016.06.023

The finite-temperature code minimizes the grand potential. In terms of ρ and f_κ, the grand potential at inverse temperature β is given by

$$\Omega = E - \beta^{-1} S + \left(\sum_{\tau_z} \mu_{\tau_z} N_{\tau_z} \right) \tag{12}$$

with E from Eq. (11) and entropy S given by

$$S = -\sum_\kappa \left(f_\kappa \ln f_\kappa + (1-f_\kappa) \ln(1-f_\kappa) \right), \tag{13}$$

and the expectation values of particle number N_{τ_z} in the last term. The latter are segregated in parentheses because that term has no role in the gradient evaluation; the minimization will be carried out at fixed N_{τ_z}.

4. The hybrid minimization method

The minimization with respect to the elements of U is same in both codes. The constraint that U is orthogonal is satisfied in the iterative process by starting with an orthogonal matrix and updating it by an explicitly orthogonal transformation. The update from U to U' can be expressed as a Thouless transformation of U,

$$U' = e^Z U. \tag{14}$$

Here Z is a skew-symmetric matrix of the independent variables $z_{\kappa\lambda}$ ($\kappa < \lambda$), giving $N_b(N_b-1)/2$ variational parameters in the general case, i.e. without any conserved quantum numbers. The block structure associated with the (τ_{zi}, π_i, m_i) quantum numbers greatly reduces that number.

The gradient of E (Eq. (11)) with respect to the elements of the Z matrix is performed analytically to arrive at the expression

$$\frac{\partial E}{\partial z_{\kappa\lambda}} = H^{orb}_{\kappa\lambda}(f_\kappa - f_\lambda). \tag{15}$$

Here H^{orb} is the single-particle Hamiltonian in the orbital basis,

$$H^{orb} = U H^{sp} U^T. \tag{16}$$

Given the gradient, the simplest algorithm to update U is the steepest descent method. Here one would use Eq. (14) with

$$z_{\kappa\lambda} = \eta_z \frac{\partial E}{\partial z_{\kappa\lambda}} \tag{17}$$

where η_z is some small numerical parameter that controls the stability of the algorithm and its convergence rate. However, convergence of the steepest descent iteration is often poor. A much more efficient algorithm is used by Robledo in his HFB code [3]. It takes into account approximately the curvature of the energy surface by introducing a preconditioner into the right-hand side of Eq. (17).

The present code employs a different method that achieves the same purpose, which we call the hybrid method. At each iteration step, the code diagonalizes a modified orbital Hamiltonian H^{orb}_η with the same diagonal elements as H^{orb} but reduced off-diagonal elements:

$$H^{orb}_\eta|_{ij} = \delta_{ij} H^{orb}_{ii} + \eta_z (1-\delta_{ij}) H^{orb}_{ij}. \tag{18}$$

The transformation matrix U_η that diagonalizes H^{orb}_η is used to update U,

$$U' = U_\eta U. \tag{19}$$

In the limit $\eta_z \ll 1$ the method amounts to a perturbative approximation to the U_η, equivalent to Robledo's preconditioned form

$$z_{\kappa\lambda} = \eta_z \frac{1}{\left| H^{orb}_{\kappa\kappa} - H^{orb}_{\lambda\lambda} \right|} \frac{\partial E}{\partial z_{\kappa\lambda}}. \tag{20}$$

One caveat: the U_η must keep orbitals ordered by the diagonal elements $H^{orb}_{\kappa\kappa}$. The hybrid method also transforms the empty and filled orbitals among themselves, but that does not change ρ or affect any HF observables.

In another limit, namely $\eta_z = 1$, the method amounts to a straightforward diagonalization of the single-particle Hamiltonian. This is often part of the update process in non-gradient methods. Thus, the hybrid method achieves both update techniques under the control of a single parameter.

Part of the update may require forcing a change in the expectation value of a single-particle operator. For that purpose, U is updated by a direct approximation to Eq. (17), as discussed in the next section.

4.1. Operator constraints

Typically, there are many local minima of the Hartree–Fock energy functional. They will also be present in the grand potential, becoming weaker as the temperature of the ensemble increases. It is important to permit additional constraints on the solutions beyond those for the number operators, in order to explore the energy surface and locate the possible minima. This is facilitated in the code by allowing the user to numerically define a single-particle operator Q and constrain its expectation value or just add it as fixed external field. As an external field, the user supplies a Lagrange multiplier λ_q and the gradient is derived from the single-particle Hamiltonian

$$H^{sp}_\lambda = K + V - \lambda_q Q. \tag{21}$$

The other option, constraining $\langle Q \rangle$ to some value q, requires the gradient updating algorithm to carry out two tasks. The first is to correct the wave function to bring $\langle Q \rangle$ closer to its target value. This step is based on a Z matrix with elements given by

$$z_{\kappa\lambda} = \frac{q - \langle Q \rangle}{\mathrm{Tr} Q^{ph} (Q^{ph})^T} Q^{ph} \tag{22}$$

where

$$Q^{ph}_{\kappa\lambda} = Q^{orb}_{\kappa\lambda}(f_\kappa - f_\lambda) \tag{23}$$

and Q^{orb} is the operator in the orbital basis as in Eq. (16). The updating matrix must be orthogonal, but need only approximate the exponential e^Z. The code uses a simple Padé approximant to preserve the orthogonal character [5]

$$e^Z \approx (1 + Z/2)(1 - Z/2)^{-1}. \tag{24}$$

In the presence of the constraint, the U update for minimization must also be modified to project Z to a direction that keeps $\langle Q \rangle$ fixed. This is carried out by replacing H^{orb} by

$$H^{orb'} = H^{orb} - \frac{\mathrm{Tr}(H^{orb} Q^{ph})}{\mathrm{Tr}(Q^{ph}(Q^{ph})^T)} Q^{ph}. \tag{25}$$

4.2. Special at zero temperature

At zero temperature, the occupation numbers f_κ are zero or one for each orbital. For the input data, the set $\{f\}$ is specified by the

Please cite this article in press as: G.F. Bertsch, J.M. Mehlhaff, A finite-temperature Hartree–Fock code for shell-model Hamiltonians, Computer Physics Communications (2016), http://dx.doi.org/10.1016/j.cpc.2016.06.023

particle number in each block rather than orbital-by-orbital. The neutron and proton numbers for the nucleus are determined by the initial $\{f\}$ array, $N_{\tau_z} = \sum_\kappa f_{\tau_z,\kappa}$. Any change in $\{f\}$ is discontinuous so there can be no gradient method to effect a change. The code permits two alternatives to deal with the situation. The $\{f\}$ can be kept fixed throughout the iteration process. As will be shown in the examples, this option gives a very good control to locate nearly degenerate local minima. The code also permits updates of the occupations numbers. In that option, in each iteration cycle the code populates the orbitals with the lowest single-particle energies. Those determined by the diagonalization of H^{orb} or its constrained forms H^{orb}_λ and $H^{orb'}$.

The situation may arise that one wishes to treat nucleus having one or two partially filled shells as spherical. In that case, the zero-temperature code would have be modified to specify fractional occupation numbers for the partially filled shells.

4.3. Finite temperature

The finite-temperature code minimizes the grand potential Ω or equivalently the partition function of the grand canonical ensemble. The occupation factors are now real numbers satisfying $0 \le f_\kappa \le 1$. Rather than using f_κ directly, the code uses variables α_κ related to f by

$$f_\kappa = \frac{1}{1+e^{\alpha_\kappa}}. \tag{26}$$

The gradient of Ω with respect to the α variables can be carried out independently of the gradient with respect to z. The latter has the same form as in the zero-temperature minimization,

$$\frac{\partial \Omega}{\partial z_{\kappa\lambda}} = \frac{\partial E}{\partial z_{\kappa\lambda}} = H^{orb}_{\kappa\lambda}\,(f_\kappa - f_\lambda). \tag{27}$$

The gradient with respect to α is given by

$$\frac{\partial \beta\Omega}{\partial \alpha_\kappa} = \left(\alpha_\kappa - \beta H^{orb}_{\kappa,\kappa}\right) f_\kappa (1-f_\kappa). \tag{28}$$

In the code, the updated set $\{\alpha'_\kappa\}$ is computed as

$$\alpha'_\kappa = (1-\eta_\alpha)\alpha_\kappa + \eta_\alpha\left(\alpha_\kappa - \beta H^{orb}_{\kappa,\kappa}\right) + \alpha_{\tau_z}. \tag{29}$$

Here η_α is the coefficient of the gradient. The second term is proportional to the gradient times the preconditioner $(f_\kappa(1-f_\kappa))^{-1}$. The last term is an τ_z-dependent constant that can be interpreted as β times the chemical potential. It is determined from the equation $N_{\tau_z} = \sum f_\kappa(\alpha'_{\tau_z})$ where N_{τ_z} are the proton and number numbers in the data input. To update $\langle Q\rangle$ at the same time as α would be more complicated (see Eq. (21) of Ref. [4]) and was not implemented in `HFgradT.py`.

In practice, we have not found any convergence difficulty with respect to the α update taking $\eta_\alpha = 1$ as in other iteration schemes. Still, it is reassuring to have a gradient method available for the f variables: it guarantees that every cycle of U and f updates lowers the grand potential for sufficiently small η_z and η_α.

5. Numerical

The most time-consuming operation in the codes is the multiplication carried out in Eq. (9). However, matrix ρ_{ij} is fairly sparse because of the restrictions to axial symmetry and good parity; the nonzero elements are in subblocks of given τ_z and K^π. The code takes advantage of the block structure of ρ_{ij} to store it as a packed array. This requires the array `b2idx0[:]` to store the entry points to the subblocks and the array `b2D[:]` to store the dimensions of the subblocks. Thus the Python code to access the i,j element of the nth subblock is `rho[n,i,j] = rho_packed[b2idx0[n]+b2D[n]*i + j]`. For the examples below, the single-particle spaces have dimensions 40 for protons and 66 for neutrons. Only $K > 0$ orbitals are needed, so the total dimension of the ρ matrix is $53 \times 53 = 2809$. After the packing, the dimension is reduced to 161. The considerations apply to the matrix U and the matrix for the single-particle operator Q. The starting trial U in the examples is the identity matrix. The identity matrix may be specified as a default option or read in explicitly as a packed array. The single-particle operator in the example is the usual quadrupole field with matrix elements

$$Q_{ij} = \int d^3r \phi^*_{j_i,m_i} r^2 Y_{20}(\theta) \phi_{j_i,m_i} \tag{30}$$

where $\phi_{j,m}$ is a shell-model orbital of a Woods–Saxon single-particle potential [6]. It is read in as the packed array `Q.dy162.dat`.

The two-body interaction is also stored with the same packing, allowing Eq. (9) to be evaluated by ordinary matrix–vector multiplication.

6. Running the codes

The user must supply files that specify the shell-model space and the one-body and two-body matrix elements of the Hamiltonian in the space. The files defining the shell-model space and the shell-model Hamiltonian follow the convention defined in Ref. [7]. Note that Hamiltonian interaction matrix elements are input in the neutron–proton formalism rather than the isospin formalism.

The input data also requires a file of the initial occupation numbers $\{f\}$. The initial basis-to-orbital transformation U can be read from a file or optionally taken as the unit matrix. For `HFgradZ`, the occupation numbers refer to blocks and the size of the array is equal to the number of blocks. For the `HFgradT`, the input occupation numbers refer to orbitals and the size of the array is the dimension of the orbital space. Note that only the orbitals with positive m are included in the array; the orbitals with negative m are treated assuming that the wave function is invariant under time reversal.

In practice, the initial transformation matrix can be quite crude, as long as it is an orthogonal matrix. In several of the examples below, the initial U is taken as the unit matrix.

Another array required by the code is the matrix of some one-body field Q such as the quadrupole operator. Both U and Q inputs are in the packed-block array format.

There are three numerical parameters related to the iteration algorithm. They are η_z defined in Eq. (18) and η_α defined in Eq. (29) and a convergence criterion δ. The iteration is terminated in `HFgradZ.py` when the condition $\sum_{ij}(\rho^{new}_{ij} - \rho^{old}_{ij})^2 < \delta$. A similar condition is applied in `HFgradT.py`, but using the change in the H_{sp} matrix to monitor the convergence.

The command line input file contains 6 or more lines as follows:

Line 1. Name of file defining the shell-model space;

Line 2. Name of the file defining the shell-model Hamiltonian;

Line 3. Name of file giving the initial occupation numbers f of the single-particle HF orbitals, followed by a flag: 'F' for fixed occupation numbers, 'U' to update occupation numbers;

Line 4. Name of file defining the initial transformation matrix U; or 'None' for the default option which is U = identity matrix;

Line 5. Ground-state code: η_z, `delta`, `itermax`; or

Line 5'. Finite-temperature code: η_z, η_α, `delta`, `itermax`, Z, N;

Line 6. Name of file defining a single-particle field Q, flag for constraint status (none = 'N',Lagrange = 'L', Constrained = 'C'), λ_q or $\langle Q\rangle$;

Line 7+. Inverse temperature β (MeV^{-1}) (one or more lines in `HFgradT`).

Please cite this article in press as: G.F. Bertsch, J.M. Mehlhaff, A finite-temperature Hartree–Fock code for shell-model Hamiltonians, Computer Physics Communications (2016), http://dx.doi.org/10.1016/j.cpc.2016.06.023

7. Output

The principal outputs of the code, written to the terminal, are the number of iterations `niter`, the final energy E, and the expectation value of the quadrupole operator Q or other single-particle operator provided in the input data. The finite-temperature code also reports the entropy of the ensemble, S in Eq. (13).

The code also writes to terminal a table of orbital properties. The columns are:

(1) index for the orbital;

(2) index of the block containing the orbital;

(3) charge of the nucleon (0 or 1);

(4) K quantum number;

(5) parity π: 0 or 1 for even or odd parity respectively;

(6) occupation number f, integer for zero temperature and floating-point for finite temperature;

(7) single-particle energy.

In addition, the code writes the final U matrix and f array to files `u_new.dat` and `n_new.dat`, respectively. In the zero-temperature code the second file has two lines. The first line gives the number of occupied orbitals in each block and can be used as an input file to `HFgradZ`. The second line gives the occupation number for each orbital in the format needed by `HFgradT`. Apart from that, the two files are in proper format to be used as input to rerun the minimization. If the minimization is converged, the rerun should only require one iteration step.

8. Two examples

The examples use input Hamiltonians for ^{162}Dy and ^{148}Sm, taken from Refs. [6,8]. The shell scripts below illustrate the various options available when running the codes.

8.1. ^{162}Dy

`dy162Z.sh`: This script runs the zero-temperature code allowing occupation number changes during the iteration. The final energy, $E = -371.78$ agrees with Table II of Ref. [9].

`dy162_def-sph.sh`: This script runs the finite temperature code for several β values in the vicinity of the deformed-spherical phase transition. The output quadrupole moments $\langle Q\rangle_\beta$ are shown in Fig. 1. A phase transition at $\beta \approx 0.83$ MeV^{-1} is evident. This is a well-known artifact of mean-field theory and is absent in more refined treatments [6,8],

`dy162ZL.sh,dy162ZC.sh,dy162TZ.sh`:

These scripts exhibit the use of a constraining field. The scripts with an "L" add the field with a Lagrange multiplier. The scripts with a "C" constrain the expectation value of the field. The zero-temperature input parameters have been chosen to show convergence to the same state by both methods. Here the converged solution has $E = -370.23$ and $Q = 587.5$.

8.1.1. ^{148}Sm

`sm148U.sh`: This script shows that the iteration process may fail to converge when the occupation numbers are allowed to change at each iteration step. It turns out that the update cycles between two sets of occupation numbers. The two sets differ by a single pair of neutrons moving between block $K^\pi = 1/2^-$ and block $K^\pi = 3/2^-$.

`sm148F.sh`: This script runs the code for each of the occupation number sets from the previous script. There is no longer an oscillation, and both runs converge. The total energies of the two minima are very close to the entry for that nucleus in Table II of Ref. [9]. The two solutions can be distinguished more clearly by their quadrupole moments, 313 and 342 fm^2 respectively.

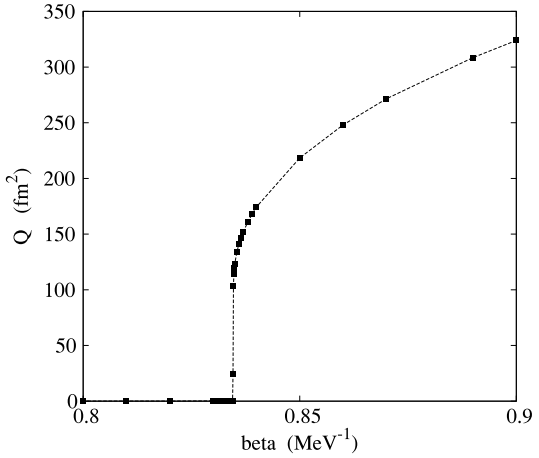

Fig. 1. Quadrupole moment as a function of inverse temperature for ^{162}Dy as computed by the `HFgradT.py` code.

9. Other mean-field approximations

Some insight into the workings of the codes might come from examining how other mean-field approximations could be carried out in the same computational framework. The use of packed arrays simplifies the coding because details of the approximations are buried in the packing. In order of difficulty, we discuss relaxing the constraints on axial quantum numbers, relaxing the constraint on time-reversal invariance, and going to the Hartree–Fock–Bogoliubov generalized mean-field approximation (HFB).

9.1. Axial symmetry and parity invariance

From a coding point of view, the constraints on axial symmetry and parity invariance could be removed with little effort. These constraints are determined in the function `initBlocks` where the block structure is set up, and changing the criteria is trivial. The computation and storage of interaction matrix elements in the function `mk_vv` automatically use the same block structure, so no changes would be required. The matrices of course may become large, but for the single-particle space treated in the examples, the size would still be manageable with the standard linear algebra library package (`numpy.linalg`). Another potential problem is to make sure that unwanted symmetries are already broken in the input wave function file U. This can be easily done by a randomizer as discussed in Ref. [4].

9.2. Breaking time-reversal invariance

To calculate odd-A nuclei and for other purposes one might wish to break the assumed time-reversal symmetry in the present codes. This requires doubling the dimensions of the basis to include both plus and minus m quantum numbers. This leads to a doubling of the number of blocks. The only coding issue is to separately compute the matrix elements of basis orbitals with positive or negative m quantum numbers in `mk_vv`.

Please cite this article in press as: G.F. Bertsch, J.M. Mehlhaff, A finite-temperature Hartree–Fock code for shell-model Hamiltonians, Computer Physics Communications (2016), http://dx.doi.org/10.1016/j.cpc.2016.06.023

9.3. HFB

The authors are not aware of any published codes for computing nuclear properties in the finite-temperature HFB approximation, but there is at least one shell-model code [4] available for calculating HFB ground states. In that code and the ones used in Ref. [3,9] the wave-function matrices are updated in a completely different way from the algorithms implemented in the present codes. However, the generalization from ground state to finite temperature should be straightforward using the method implemented in the present HF codes.

Acknowledgments

We would to thank Y. Alhassid and L. Robledo for discussions leading to this work, and H. Nakada for the use of his Hartree–Fock code to validate the codes presented here. Support for this work was provided by the US Department of Energy under Grant No. DE-FG02-00ER41132.

Appendix. Key functions in the codes

The coded equations from the text above are listed here, together with their location in the code.

Eq. (4): `util.calcRho`

Eq. (8): `hfsetup.mk_vv`

Eq. (9): `util.calcV`

Eqs. (10), (21): `util.calcHsp`

Eq. (11): `util.totalE`

Eq. (12): `HFgradT`

Eq. (13): `util2.entropy`

Eq. (16): `util.calcOrbOp`

Eqs. (18), (19): `util2.updateU`

Eqs. (22)–(24): `util2.resetQ`

Eq. (25): `util2.projectZ4`

Eq. (29): `util2.updatef`

References

[1] P. Ring, P. Schuck, The Nuclear Many-Body Problem, Springer, 1980.
[2] A. Baran, et al., Phys. Rev. C 78 (2008) 014318.
[3] M. Warda, et al., Phys. Rev. C 66 (2002) 014310.
[4] L.M. Robledo, G.F. Bertsch, Phys. Rev. C 84 (2011) 014312. The code may be found in the Supplementary Material.
[5] L. Robledo, private communication.
[6] Y. Alhassid, L. Fang, H. Nakada, Phys. Rev. Lett. 101 (2008) 082501.
[7] B.A. Brown, W.A. Richter, Phys. Rev. C 74 (2006) 034315.
[8] C. Özen, Y. Alhassid, H. Nakada, Phys. Rev. Lett. 110 (2013) 042502.
[9] Y. Alhassid, G.F. Bertsch, C.N. Gilbreth, H. Nakada, Phys. Rev. C 93 (2016) 044320.

Please cite this article in press as: G.F. Bertsch, J.M. Mehlhaff, A finite-temperature Hartree–Fock code for shell-model Hamiltonians, Computer Physics Communications (2016), http://dx.doi.org/10.1016/j.cpc.2016.06.023