A Code for Three-Dimensional Static and Time-Dependent Hartree–Fock with a Simplified Effective Force

1 Physical Model

The present code solves the static and time-dependent Hartree–Fock (TDHF) equations in a general three-dimensional geometry with a simple interaction (simplified Skyrme) that simulates nuclei in a crude way. The concept is based on the fundamental paper by Bonche et al. [4]. To get an idea of the nuclear physics applications possible with such a model we refer the reader to Refs. [4, 3, 7, 26, 5, 23, 10, 8, 12, 19, 13, 6, 14, 9, 18], for reviews to Refs. [20, 11]), and for recent developments in the theory to Refs. [11, 19, 22, 15, 16],

Self-consistent mean-field methods for stationary states are explained in the book in Chaps. 5 and 6, while the time-dependent case is described in Sect. 8.2. We take up here the form of Eq. (6.15) in the book and simplify it to

$$E_{\mathrm{Sk}} = \frac{\hbar^2}{2m}\sum_{\alpha=1}^{A}\int \mathrm{d}^3r|\nabla\varphi_\alpha|^2| + \frac{1}{2}\int \mathrm{d}^3r\left[\frac{3t_0}{8}\rho^2 + \frac{3t_3}{32}\rho^3\right], \tag{1}$$

where $\rho = \rho(\mathbf{r})$ denotes the total local density. Again, we consider $A = N+Z$. Furthermore, we neglect the Coulomb energy and thus have $\rho_p = \rho_n = \rho/2$. The force parameters b_0, b_3 in Eq. (6.15) have been replaced by the traditional t_0, t_3 [1]. In fact, the functional (1) is a simplified version of the Bonche–Koonin–Negele interaction (BKN) which was used very often for schematic studies of nuclear TDHF [4]. The physical mechanism described by this force is that the t_0-term provides a density-independent attraction, while the t_3-term simulates a repulsion that increases in strength with density, thus leading to saturation. The mean-field calculated with this force is

$$U(\vec{r}) = \tfrac{3}{4}t_0\rho(\vec{r}) + \tfrac{3}{16}t_3\rho^2(\vec{r}) \quad . \tag{2}$$

The complete single-particle Hamiltonian is thus

$$\hat{h} = -\frac{\hbar^2}{2m}\nabla^2 + U(\vec{r}). \tag{3}$$

2.2 The Static Problem

As pointed out in Chaps. 5 and 6 of the book, the self-consistent mean-field equations pose a nonlinear problem. In most cases, it can be solved only in iterative manner. We use here the damped gradient iteration [21, 2] in the form

$$\phi_\alpha^{(n+1)} = \mathcal{O}\left\{\phi_\alpha^{(n)} - \hat{\mathcal{D}}\left(h^{(n)} - \langle\phi_\alpha^{(n)} \mid h^{(n)} \mid \phi_\alpha^{(n)}\rangle\right)\phi_\alpha^{(n)}\right\} \quad , \tag{4}$$

$$\hat{\mathcal{D}} = \frac{x_0}{E_0}\frac{E_0}{\frac{\hat{p}_x^2}{2m}+E_0}\frac{E_0}{\frac{\hat{p}_y^2}{2m}+E_0}\frac{E_0}{\frac{\hat{p}_z^2}{2m}+E_0} \quad , \tag{5}$$

where the upper label n indicates the number of iteration and x_0, E_0 are numerical parameters to be adjusted for optimal speed and stability. The E_0 should account for the typical depth of the local mean field, i.e., $E_0 \approx U_{\min}$, which amounts to $E_0 \approx 100$ MeV for the case of nuclei. A good overall step size x_0 is typically $x_0 \approx 0.2-0.8$. High values can produce faster convergence, however at the risk of spoiling convergence at all. Low values are safer, but require more iterations. Optimal choices are a matter of experience (and test cases). Note that the step terminates if the mean field equation $h^{(n)}\phi_\alpha^{(n)} = -\langle\phi_\alpha^{(n)}|h^{(n)}|\phi_\alpha^{(n)}\rangle\phi_\alpha^{(n)}$ is fulfilled. The mismatch of the mean-field equations (before convergence) is used to improve the solution. With $\hat{\mathcal{D}} = x_0 =$ constant, one has the simple gradient step which propagates along the downhill gradient in the energy landscape. However, high-energy components in the kinetic energy would require to use extremely small step sizes x_0 and thus very slow convergence. The damping operator $\hat{\mathcal{D}}$ reduces the high-energy components, thus allowing larger steps and with it much faster convergence [21, 2]. We use here a separable form of the damping operator. This is numerically very efficient as it allows to employ successively one-dimensional operations which are rather fast.

One needs observables to check and record proper convergence of the iteration. The most obvious and widely used criterion is the evolution of the total energy $E^{(n)}$ in terms of the energy difference $\delta E^{(n)} = E^{(n)} - E^{(n-1)}$. However, a small value of $\delta E^{(n)}$ does not ensure that a solution has really been found. More critical and reliable is the variance of the mean-field Hamiltonian

$$\Delta h = \sqrt{\frac{\sum_\alpha \Delta h_\alpha^2}{\sum_\alpha}} \quad , \quad \Delta h_\alpha^2 = \langle\phi_\alpha^{(n)}|{h^{(n)}}^2|\phi_\alpha^{(n)}\rangle - \langle\phi_\alpha^{(n)}|h^{(n)}|\phi_\alpha^{(n)}\rangle^2 \quad . \tag{6}$$

The variance Δh_α^2 is a direct measure for the remaining uncertainty on the single-particle energy. Driving this to a sufficiently small value guarantees a known quality of the solution. We will use Δh as criterion for terminating the iteration.

2.3 The Dynamic Problem

Once the static wave functions have been obtained, they are used to initialize the time-dependent problem. The wave functions for the jth nucleus are inserted into

the grid at a given center-of-mass $\vec{R}_j$ and multiplied with a plane wave phase factor $\exp[\mathrm{i}\vec{k}_j \cdot \vec{r}]$, where $\vec{k}_j$ determines the initial direction and kinetic energy of the nucleus.

For the numerical solution of the time-dependent problem the basic idea is to approximate the evolution operator for a short time step Δt. Formally, the solutions to the time-dependent Hartree–Fock equations are given by

$$\phi(\vec{r}, t + \Delta t) = \exp\left(-[\frac{\mathrm{i}}{\hbar}\int_t^{t+\Delta t} \mathrm{d}t'\ H(t')\right)\ \phi(\vec{r}, t) \quad . \tag{7}$$

The time dependence of the single-particle Hamiltonian poses a special problem, since it depends on the unknown wave functions. It is not sufficient to just approximate it by its value at the beginning of the time step; in practice it was found that this procedure violates energy conservation strongly. For example, for the case of a nucleus moving uniformly through space, the mean-field in the Hamiltonian would always correspond to the initial position of the nucleus during the time step and thus lag behind, causing a spurious slowing of the motion. Satisfactory accuracy can be reached if the Hamiltonian is estimated at the center of the time step (the same numerical effect is also illustrated by the midpoint approximation to an integral). The estimation itself can be done with lower accuracy.

The propagation for one time step thus proceeds in two phases:

- Estimation of $H(t + \Delta t/2)$: estimate the wave functions at half time via

$$\phi(\vec{r}, t + \Delta t/2) = \exp\left(-\frac{\mathrm{i}}{\hbar}H(t)\Delta t/2\right)\ \phi(\vec{r}, t) \quad , \tag{8}$$

  and then compute the density and the Hamiltonian according to (3).
- In the second phase use this Hamiltonian for the final propagation

$$\phi(\vec{r}, t + \Delta t) = \exp\left(-\frac{\mathrm{i}}{\hbar}H(t + \Delta t/2)\Delta t\right)\ \phi(\vec{r}, t) \quad . \tag{9}$$

  These wave functions then provide the starting point for the next time step.

The exponential function in these expressions is then approximated by the Taylor expansion, usually to sixth order, although other orders can easily be used as well. Thus

$$e^A\phi \approx \left(1 + A\left(1 + \frac{A}{2}\left(1 + \frac{A}{3}\left(1 + \frac{A}{4}\left(1 + \frac{A}{5}\left(1 + \frac{A}{6}\right)\right)\right)\right)\right)\right)\phi \tag{10}$$

with

$$A = -\frac{\mathrm{i}}{\hbar}H(t + \Delta t/2)\Delta t. \tag{11}$$

This approximation to the time-development operator is neither Hermitian nor energy conserving; it relies purely on high accuracy to realize these properties. In practice a time step between 0.1 and 0.2 fm/c turned out to be adequate for the standard grid spacings. In any case, if the time step is too large, the calculation will become unstable after some time, and the norm of the wave functions and the energy going to highly unphysical values. In this case the time step should be reduced.

Note that when varying the spatial grid spacing, the time step should be changed proportionally.

3 Structure of the Code

In this section we give a brief overview of the source files and their contents. Details should be understandable using the comments in the source itself.

1. `dynamic.f90` This contains the code specific to the dynamic calculation.
   - `dynamcihf`: a wrapper for calling the principal routines, `tsetup` and `tevolv`.
   - `tsetup`: initializes the wave functions by either reading in a restart file (`rdtpsi`) or by setting up from static fragment wave functions (`read_fragments`).
   - `tevolv`: performs the time integration in a loop over time steps.
   - `tstep`: calculates the series expansion of Eq. (10) for either the full or the half step.
   - `tinfo`: produces various kinds of output.
   - `getin_dynamic`: reads the dynamic input data.
2. `energy.f90` contains the subroutine `hfenergy` for calculating the various contributions to the energy.
3. `fields.f90` has basic subroutines for operating on the spatial grid. These are:
   - `init_grid`: sets up the coordinate values and the index vectors for doing differences with periodic boundary conditions. It also calculates the damping matrices via inversion. Uses helper subroutine `getinc` applied to the three coordinate directions.
   - `rpsnorm`: calculates norm of a wave function.
   - `overlap`: calculates overlap of two wave functions.
   - `density`: sums up total density over wave functions.
   - `schmid`: performs Gram–Schmidt orthogonalization for the wave functions.
   - `laplace`: applies the Laplace operator to a wave function using second-order differencing.
   - `cmulx`, `cmuly`, `cmulz`: multiply a wave function with a matrix along one coordinate direction. This is used for applying the damping matrices.

13. `static.f90`: static part of the code.
    - `shf3d`: wrapper for the static calculations containing the initialization and the iteration loop.
    - `grstep`: performs one step of the gradient iteration.
    - `sinfo`: produces informational output.

4 How to Run the Code

To run the code a Fortran-95 compiler is required, such as the free `gfortran`. Since the code uses the module structures of Fortran 95 extensively, the source files have a nontrivial dependency that requires compiling them in a certain order. A standard `Makefile` (adapted to `gfortran`) is provided that allows compiling the code with a simple `make` command issued in the code directory. For a different compiler just replace the name `gfortran` in the `Makefile`. Integrated development systems usually will find out the interdependence of the source files automatically.

In case both techniques are not available, the suggested sequence for translating the source files is:

```
matinv.f90 --- params.f90 --- fields.f90 --- energy.f90 ---
moment.f90 --- fragments.f90 --- mfield.f90 --- static.f90 ---
dynamic.f90 --- info_gnuplot.f90 --- info_util.f90 --- init.f90
--- inout.f90 --- main3d.f90
```

The executable program is called `tdhf.exe`. To run the code it is suggested to put the printer output into a file by using a command such as (for Linux)

```
./tdhf.exe > out.txt &
```

Sample input files `for005` are provided in the subdirectories `Stat` and `Dyn` for a static and a dynamic calculation, respectively. Since a dynamic calculation requires static wave functions, static calculations always have to be run before one or many dynamic calculations using these wave functions for the initial conditions. A detailed description of the input is given in the next section.

The code contains the dimensioning of the wave functions `nx`, `ny`, `nz`, and `npsi` as `PARAMETER` variables in source file `patams.f90`. The correct values have to be inserted here and the code has to be recompiled for each application.

The procedure for a complete calculation therefore looks like this:

1. Run one or several static calculations. For each insert the desired values of `nx`, `ny`, `nz`, and `npsi` into `params.f90`, recompile the code, and run it. The principal result should be a file containing the static wave functions.
2. Once all the static wave functions have been computed, set up `params.f90` for a larger grid and for a total number of wave functions `npsi` equal to the sum of those in the fragments. Recompile and then run it with the dynamic `for005` as input.

It should be mentioned that in the tradition of scientific codes this code does not attempt to catch all input errors except for the most serious ones.

5 Input Data

Note that the code uses units based of fm for distance, fm/c for time, and MeV for energy.

Input for the code is provided in a file `for005`. For ease of use it is in several `NAMELIST`s. A description is given in the following:

1. Namelist `files` allow defining the names of some files written in the course of the calculation. Output can be suppressed by setting these to `'none'` or `'NONE'`.

   - `restartfile`: name of the restart file.
   - `staticfile`: name of the file written at the end of a static calculation to save the converged wave functions.

2. Namelist `main` defines the full set of parameters for the static calculation; some of these are also used in the time-dependent calculation.

   - `imode` determines whether a static `imode=1` or dynamic `imode=2` calculation is desired. In the dynamic case, further input is needed from namelist `dynamic` (see below).
   - `dx` is the spatial grid spacing in fm.
   - `t0` and `t3` are the parameters of the BKN force as defined in 1.
   - `radinx`, `radiny`, `radinz` determine the scaling in the three coordinate directions of the oscillator wave functions used to initialize the static iterations. This allows starting with a deformed shape.
   - `itrbx` determines the maximum allowed number of static iterations. When these are reached, the wave functions are written into the static file and the code stops. If necessary, it can be restarted for more iterations if a restart file was produced.
   - `serr` defines the convergence criterion. If the sum of fluctuations in the single-particle energies 6 becomes less that `serr`, convergence is assumed, the code writes the output wave functions and terminates.
   - `irest` and `mrest` control the restart facility. A restart file is written in every `mrest` iterations or time steps. `irest=0` indicates a new calculation, while a nonzero value corresponds to the iteration or time step number when the restart file was written.
   - `iprint`, `mprint`, and `mplot` control printed output. This can be reduced to a minimum by setting `iprint=0`. `mprint` indicates the frequency of detailed output. `mplot` determines the frequency of three outputs: a simple printer plot of the density in the x–z plane, a file containing `gnuplot` commands for a better quality plot (for details examine subroutine `plot_gnu` in

of the single-particle states are written: the energy fluctuations, norm, kinetic, and total energy.

The files with extension .tdd contain a binary record of the densities at regular intervals in time or static iteration. The detailed contents can be gleaned by looking at subroutine write_densities in source file inout.f90.

6.2 Static Calculation

The file conver.res contains a protocol of convergence. As functions of the iteration number it lists the total energy and the mean square deviations of the single-particle energies as given in Sect. 6. Note that the latter are much more critical than the energy: it takes a lot more iterations to stabilize the single-particle wave functions than the total energy.

The final result of the static calculation is written onto the file whose name is determined by the variable staticfile. Its contents can be seen by examining subroutine putpsi in source file inout.f90.

6.3 Dynamic Calculation

The file dynamic.res contains a listing of the total energy and the kinetic energy as functions of time.

The file plot2D.gnu has the source code for generating a series of two-dimensional density plots in the x–z-plane using gnuplot. The data for the plots are contained in the files "*.rho".

7 Suggestions for Calculations

The sample static calculation calculates a pseudo-^{16}O nucleus. To judge the correct running of the code, the resulting output files are provided in output.zip. Note that the grid size $24 \times 24 \times 24$ and number of nucleons (16, corresponding to four wave functions) are defined in params.f90, not in for005.

The static sample takes two of these nuclei, inserts them into a larger grid of $32 \times 24 \times 32$ points at positions $(-6, 0, 2)$ and $(+6, 0, -2)$ and sets them motion with a total kinetic energy each of 10 MeV, but in opposite direction along the x-axis. This is followed for 6,000 time steps. The resulting output is again provided as output.zip.

Here are a few suggestions for some calculations:

- Test the accuracy of the code by examining the dependence of the results on the size of dx or dt. Another interesting experiment is to shift the center of the static nucleus to a grid point instead of having it in the center of a cell (this can be achieved by changing the computation of the arrays xs, ys, and zs in grids.f90).