Documentation

Navneet Krishnan
Department of Theoretical Physics, Australian National University, Canberra

October 2020

In my work, I wrote a Python script to extract the wavefunctions from the Sky3D output and compute the relevant operator expectation values and response factors. Note that this script was designed for a static calculation from Sky3D, which determines the ground state of a nucleus. It was not designed to be used with a dynamic Sky3D calculation. I provide documentation of the code here, and the script itself can be found at GitHub: `https://github.com/Nav-Krishnan/Observables-and-Response-Factors`. In this documentation, I detail how the code is run and provide a list functions that can be called to compute the relevant expectation values and the nuclear response factors. Directions on potential modifications to the script are commented in the script file itself. Note that the script requires access to the SciPy, SymPy, and NumPy external libraries.

Sky3D generates a binary file (labelled `WFfile` by default) containing the single-particle wavefunctions and other data on the nuclear state. When the script is run, the user will be prompted to enter a file name for the binary file. The script will then extract the relevant information (such as the proton and neutron numbers) from the binary file and define the relevant functions for the user to call. My script can be used if pairing has been included in the Sky3D calculation, as it also extracts and applies the state occupation numbers. In this case, rather than the particle numbers, the number of available states allowed by Sky3D for A_N protons or neutrons is:

$$\mathrm{Nint}(A_N + 1.65 \times A_N^{2/3}),$$

Where Nint denotes the nearest integer of the argument. Regardless of whether pairing is included, the number of neutron and proton states are stored as the variables `nneut` and `nprot` respectively. The total number of states is stored as `nst`.

My script assumes a mesh with 24 points in each direction, with a spacing of 1 fm. The parameters `nx` and `dx` (and likewise for the y and z components) defined in my script can be changed to fit a different choice of grid dimensions.

The code stores the extracted wavefunctions in the dictionary `Rdict` according to the particle number. For example, the wavefunction `Rdict[1]` corresponds to the first particle in the Sky3D output. Neutron wavefunctions are stored first, so the first `nneut` wavefunctions correspond to the neutron states, and the remainder to the proton states. Each wavefunction is stored as a multidimensional array, in order of spin, z, y, and x. $s = 0$ corresponds to the spin-up component, and $s = 1$ to the spin-down component. For example, for a wavefunction ψ denoted `WF` by the code:

$$\psi_+(x_i, y_j, z_k) = \texttt{WF[0][k][j][i]}.$$

The script defines several functions that can be called to determine relevant operator expectation values and response factors. A list of these, how to call them, and their outputs is given below. For one wishing to only get the long-wavelength scalar response factors for a nucleus, the only needed function after running the script is `ResponseFactors`, at the end of this list.

`norm2(WF)`: takes as input a single wavefunction `WF` and returns the norm-squared:

$$\|\psi\|^2 = \sum_{s,i,j,k} |\psi_s(x_i, y_j, z_k)|^2 .$$

The norm should always be 1 for a wavefunction; this function provides a sanity check on the Sky3D output and the script.

`innerproduct(WF1,WF2)`: takes as input two wavefunctions `WF1` and `WF2`, and returns their inner product:

$$\langle\psi|\phi\rangle = \sum_{s,i,j,k} \psi_s^*(x_i, y_j, z_k)\phi_s(x_i, y_j, z_k)\ dv.$$

Here, $dv = dx \cdot dy \cdot dz$ is the volume element for the mesh. This function can be used to calculate the inner product when `WF2` is a single wavefunction, and when it is a vector of three wavefunctions (for example, $\vec{S}\psi$). In the former case, it returns the inner product as a complex number. In the latter case, it returns a vector of three complex numbers.

`partialderx(WF,order)`: takes as input a wavefunction `WF` and an integer `order`, and returns the `order`$^{\text{th}}$ x-derivative of `WF`. As defined, it is only capable of returning the first and second derivatives, though it can be easily modified to add higher order derivatives if the appropriate derivative matrices are defined in the code. The functions `partialdery` and `partialderz` are likewise defined.

`position(WF)`: takes as input a wavefunction `WF` and returns its position expectation value.

`spin(WF)`: takes as input a wavefunction `WF` and outputs the action of the spin operator on it. Output is a vector of three wavefunctions:

$$\vec{S}\psi = (S_x\psi, S_y\psi, S_z\psi) .$$

`spinexp(WF)`: takes as input a wavefunction `WF` and outputs the spin expectation value as a vector:

$$\left\langle \vec{S} \right\rangle = (\langle S_x\rangle , \langle S_y\rangle , \langle S_z\rangle) .$$

`sigma(WF)` and `sigexp(WF)`: these act similarly to `spin` and `spinexp`, but instead calculate the action and expectation values for the pauli matrices σ_i.

`sigma1M(WF)` and `sig1Mexp(WF)`: these act identically to `sigma` and `sigexp`, but calculate the Pauli matrices in a spherical vector basis. I did not use these functions in my work, but the may be useful in calculating the response factors of states with well-defined angular momentum.

`momentum(WF)` and `momexp(WF)`: as with the spin functions, these return the action of the momentum operator on `WF` and the expectation value respectively. Both are output as vectors.

`kinetic(WF)` and `kinexp(WF)`: as with the spin functions, these return the action of the kinetic energy operator on `WF` and its expectation value respectively.

`angmomentum(WF)` and `angmomexp(WF)`: as with the spin functions, these return the action of the orbital angular momentum operator on `WF` and the expectation value respectively. Both are output as vectors.

`sl(WF)`: takes as input a wavefunction `WF` and outputs the expectation value of the spin-orbit operator $\vec{\sigma} \cdot \vec{\ell}$.

`totangexp(WF)`: takes as input a wavefunction `WF` and returns the expectation value for the total angular momentum $\vec{J}$. Output is a vector. I did not use this function in my work, but it could be useful to confirm the success of an angular momentum projection on a nuclear state.

`expwriter(name,l)`: takes as input a filename `name` (which must be given as a string) and an integer `l`, and saves a text file `<name>.txt`. The file contains, for each wavefunction up to the `l`$^{\text{th}}$:

- The norm-squared $\langle\psi|\psi\rangle$,
- The position expectation value $\langle \psi \left| \vec{x} \right| \psi \rangle$,
- The Pauli matrix expectation values $\langle \psi \left| \vec{\sigma} \right| \psi \rangle$,
- The momentum expectation values $\langle \psi \left| \vec{p} \right| \psi \rangle$,
- The kinetic energy expectation value $\left\langle \psi \left| \frac{p^2}{2m} \right| \psi \right\rangle$,
- The angular momentum expectation values $\left\langle \psi \left| \vec{\ell} \right| \psi \right\rangle$.

This function is useful in checking the script operates properly: the data in the text file can be compared with the Sky3D standard output file (called `for006.static` by default) to confirm the two agree. Note that there may be minor differences in the two results due to rounding. Additionally, the Sky3D output for spin uses the operator $\vec{S} = \frac{1}{2}\vec{\sigma}$, rather than the Pauli matrices, so a factor of two difference should be expected in the spin values between this code and Sky3D.

`ResponseFactors(name)`: Saves a text file `<name>.txt`. The file contains the response factors $F_r^{(N,N')}$ and $F_{r,s}^{(N,N')}$, as defined in Chapter 3 of the thesis, along with the integrated response factors:

$$\int_0^{100\text{ MeV}} \frac{q}{2} F_{r,s}^{(N,N')} dq.$$

Acknowledgments

In my script, I used packages from the SciPy family of libraries, which can be found at `https://www.scipy.org/`.