3D SOLVER README

Tiia Haverinen
David Muir
Gianluca Salvioni

August 3, 2016

Contents

1 The Code **2**
1.1 The Structure of the Code 2
1.2 Compiling and Running the Code 2

2 Modules **3**
2.1 variables.f90 3
2.2 calcwaves.f90 3
2.3 potentials.f90 4

Chapter 1

The Code

This code is a FORTRAN (.f90) code used to numerically solve Schrödinger's equation for a given potential - infinite square well, finite square well, Woods-Saxon potential etc.
This code uses the numerical Numerov method and shooting in order to solve the potentials outlined in this document.
The program is currently being developed to solve a Hartree-Fock problem using a Skyrme interaction (t_0 and t_3) in a 3D spherically symmetric space. Thus, from a coding point of view, it is possible to numerically code a 1D radial solution for the case of a single nucleus located in a given potential and due to symmetry arguments this will have the same solution in 3D i.e. the θ and ϕ coordinates become redundant for this problem.
The test cases for this program are Oxygen-16 and Lead-208.

1.1 The Structure of the Code

The code consists of a main.f90 file and uses three modules - variables.f90, calcwaves.f90 and potentials.f90 - all of which are discussed further in the next chapter.
In the main.f90 program we construct the nucleus of a given atom such as lead-208 or oxygen-16 by defining the number of protons and neutrons the desired nucleus has and the states that should be summed over and populated with these nucleons (assuming that the nucleus is in its ground state).
The main.f90 will call the afore mentioned modules to construct the wavefunction in the given potential and to evaluate this potential, leading to the energy of the states of the nucleons which can in turn be used to calculate the neutron, proton and total matter density of the nuclei.
Clear and detailed comments are included throughout the main program and its associate modules in order to aid users through its use effectively.

1.2 Compiling and Running the Code

The code should be complied and run by entering the following text into the command line,

```
gfortran -fno-whole-file main.f90 variables.f90 potentials.f90 calcwaves.f90 -o main
```

Alternatively, a makefile can be used to complied this program.

Chapter 2

Modules

The modules used in the code are:

1. variables.f90
2. calcwaves.f90
3. potentials.f90

2.1 variables.f90

The module "variables.f90" contains a list of physical constants that are used in the program. We also define the precision of all variables and all following calculations throughout the remaining files here as well. The content of variables.f90 is

```
module variables
implicit none
  INTEGER, PARAMETER :: dp = SELECTED_REAL_KIND(12)
  REAL(KIND=dp), PARAMETER :: pi = 3.14159265358979323846264338327950E0_dp
  REAL(KIND=dp), PARAMETER :: mc2=938.9059000E0_dp
  REAL(KIND=dp), PARAMETER :: hbarc = 197.32891000E0_dp
  REAL(KIND=dp), PARAMETER :: hb2m = 20.73553E0_dp
  REAL(KIND=dp), PARAMETER :: esquare = 1.44E0_dp
end module variables
```

2.2 calcwaves.f90

The module "calcwaves.f90" contains two subroutines - "wavef" (wavefunction) and "parwf" (partial wavefunction). This module requires the use of the variables.f90 module. This module is responsible for calculating a wavefunction given some defined potential. The subroutine "wavef" is responsible for ensuring that the trial wavefunction (composed of an upper and lower bound on the energy Eup and Edown respectively) converges, to within a tolerance of epsil. The wavefunction is constructed by means of a *shooting method* involving propagating from both the right and left sides of some given boundary conditions and matching both of these independent wavefunctions at some arbitrary point. In the case of symmetric potentials the centre of the potential is the best place to match as the solutions must agree at this point. In the case of the nucleus the wavefunction and potential are not symmetric hence, the matching point chosen is the surface of the nucleus. This is due to the fact that the matching procedure may run into difficulties are distances a few times the size of the nucleus due to the rapid drop off of the potential. Additionally, the box

size should not be much larger than the nucleus in question otherwise the results become sensitive to numerical noise.
When propagating a wavefunction from one side of the defined boundary, the numerical solution explodes at the opposite end under certain conditions (such as too large a box size). The shooting method overcomes this key flaw by propagating two wavefunctions, one from each opposite side of the boundary, and matching them at some point. This is the main purpose of the "parwf" subroutine which constructs a partial wavefunction propagating from the left and another propagating from the right and matching them by means of their continuity in their values at the matching point as well as both of them having the same derivative.
A note of caution, due to the central potential calculation we do not start iterating from zero if $l > 5$ but instead start from imin.

2.3 potentials.f90

The module "potentials.f90" is responsible for computing various potentials. This module requires the variables.f90 module. This module contains a function called "potV" used for calculating the desired potential. The predefined potentials that this module can calculate are as follows:

1. kpot = 0 No potential.
2. kpot = 1 Square well potential of size a and depth Vvalue.
3. kpot = 2 Woods-Saxon potential.
4. kpot = 3 Coulomb potential.
5. kpot = 4 Spin-orbit W potential.
6. kpot = 5 Radial potential.

where kpot is a variable in the module which specifies the potential being computed.
The calculation of the Skyrme potential is currently a work in progress.

Appendix A

Variables and Parameters Used in the Code

```
Eup = Upper limit for energy in numerov algorithm
Edown= Lower limit for energy in numerov algorithm
epsil= Demanded precision for convergence
Rmax= Right hand side limit for the box in which we are solving SE
meshsize= Distance between meshpoints
a= diffusivity (0.67 fm)
Vvalue= the depth of the square well
rzero= a constant (1.27 fm)
Nodemax=the maximum number of nodes
N_max = Max number of principal quantum number
N_maxN= Max number of principal q.n. for neutrons
N_maxZ= Max number of principal q.n. for protons
N_act= Active principal quantum number (for loops)
Num_par = number of particles (for loops)
Num_par_max= maximum number of particles
kpot= index for different potentials, see below
Etrial= Energy of the trial wavefunction
Eexp = expected energy eigenvalue (analytical solution, infinite square well)
points= maximum number of points for e.g. wavefunction
ifail= logical argument for checking if something is failed
converg= logical argument for checking convergence (0=not converged,1=converged)
i=index for loops
charge= separator for neutrons and protons (0=neutron, 1=proton)
l=orbital quantum number
jj= "2*j", so two times total angular momenta
iii= index for counting solutions
jmin= lower bound for the angular momenta
jmax= upper bound for the angular momenta
Nodecount = number of nodes
numN= number of neutrons
numZ= number of protons
strval= ??
n_rad = radial quantum number
st_post = ??
minN = index for searching minimum energy of neutron states
minZ= index for searching minimum energy of proton states
nstatN= number of neutron states
nstatZ= number of proton states
num_stat= total number of states
```

```
loop = ??
HF_conv = ??
a1= coefficient in numerov algorithm (manuale p.7)
a2= coefficient in numerov algorithm (manuale p.7)
a3= coefficient in numeror algorithm (manuale p.7)
pot= potential
normal= variable for normalization (integral over wavefunction)
potV=value of potential at some point
rProt=radius (of protons)
trialwf= trial wafefunction
pott= the whole potential (array)
Nodecc= a variable for indexing all different solutions (different N,l..)
```