Calculating phase shifts

From Pendry:

The phase shifts can be calculated by integration of the Schrodinger equation starting at the origin, where we have the boundary condition that the wave function must not be singular, and proceeding out to the boundary of the muffin-tin sphere. Once out of range of the potential we know that the wave function has the form:

|  |  |  |
| --- | --- | --- |
|  |  |  |

The r-multiplied wave function has the form:

|  |  |  |
| --- | --- | --- |
|  |  |  |

And has the derivative:

|  |  |  |
| --- | --- | --- |
|  |  |  |

And from details of the integration we calculate the logarithmic derivative at the boundary of the muffin-tin sphere, radius *R*,

|  |  |  |
| --- | --- | --- |
|  |  |  |

The logarithmic derivative for the r-multiplied wave function is:

|  |  |  |
| --- | --- | --- |
|  |  |  |
|  |  |  |
|  |  |  |

Solving eq. 2 for the exponential including the phase shift,

|  |  |  |
| --- | --- | --- |
|  |  |  |

Or

|  |  |  |
| --- | --- | --- |
|  |  |  |

If we know the phase shifts for a real energy E, we can compute the (complex) phase shift for the imaginary energy E (including the optical potential) as follows:

|  |  |  |
| --- | --- | --- |
|  |  |  |

Strategy:

1. Solve the (unbound) Schrodinger equation for real energy and angular momentum l.
2. Compute the logarithmic derivative at the muffin-tin radius
3. Apply equation (4) to compute the phase shift.
4. Use the optical potential to compute complex E .
5. Phase shifts are modified slightly for complex E according to eq. 5.

Computing the optical potential:

|  |  |  |
| --- | --- | --- |
|  |  |  |

Can take V0r from the zero of the muffin-tin potential. More likely it will be overridden as an adjustable parameter. Same can be said for the imaginary part.

Need an implementation of the spherical Hankel functions of the first and second kinds, and their derivatives.

Calculation of the spherical Hankel functions:

|  |  |  |
| --- | --- | --- |
|  |  |  |
|  |  |  |

Where the and are the spherical Bessel functions.

According to Wikipedia,

|  |  |  |
| --- | --- | --- |
|  |  |  |

The major question is – do I solve for the projectile electron wave function self-consistently along with the core electrons, with modifications to push charge outside of Rmt back into the core (by increasing the occupancy of the orbital to give charge neutrality when integrated from 0 to Rmt and setting the core wave functions to 0 outside of Rmt), or do I use Pendry’s method of assuming the core wave functions are unperturbed and work purely with the charge density?

I’m tempted to try it the self-consistent way, since it would make it easier to compute and seems more logical to do that way. The only issue I can see is the occupancy of the projectile wave function is going to scale with the energy and the angle of scattering – each of the angular momentum components are NOT equally occupied, and do not have a full electron in each. So it seems the occupancy is spread out among the L-states depending on the amplitude in that partial wave. So to solve each L with a full charge of -1 would be overestimating the impact of that wave function’s effect on the core electrons, and skewing the resulting phase shift. So for that reason alone it might make more sense to a) neglect the incident electron’s polarization of the core, or b) make it possible to compute the phase shift as a function of L-state occupancy – that would be interesting to look at, in any case.

Other approaches would be to

* Assume each L-state is occupied equally (equipartition), up to Lmax
* Assume that S-state scattering is dominant, and put the full charge into that distribution.\

Following Pendry:

Inside the muffin-tin sphere, we can decompose the incident electron wave function into its partial waves

|  |  |  |
| --- | --- | --- |
|  |  |  |

(Note) – the complexity in the integral involving Clebsch-Gordon coefficients is due to an “exact” treatment of exchange for the incident electron. MUFPOT treats exchange in the Slater approximation.

Also – MUFPOT has an interesting way of computing the spherically averaged potential assuming contributions from nearest neighbors.

Plan – it probably is a good idea to make this polymorphic and allow for different strategies. All of the implemented solutions I’ve seen so far do NOT treat the incident electron+ core electrons self-consistently. Instead, they rely on the charge density of the free atom.

Pendry explicitly says he “pushes” any charge density outside of Rmt back in, by increasing the occupancy of the level and cutting it off at Rmt.

# Constructing the muffin-tin potential:

<http://hermes.phys.uwm.edu/projects/elecstruct/mufpot/MP/MP.Theory1.html>

The strategy with this method is to start with the tabulated wave functions of Herman and Skillman, compute from those the electron density , and then to solve the Poisson equation to determine the potential due to that charge density. The resulting electron potential is added to the nuclear coulomb potential and an exchange term (Slater approximation), and then the Mattheiss prescription is invoked to construct the muffin-tin.

The starting point is the set of free atom radial wave functions tabulated by the Atom class. The (spherically symmetric) electron density can be written:

|  |  |  |
| --- | --- | --- |
|  |  |  |

Pendry does very much the same thing, but with two important differences – instead of using exchange in the Slater approximation he does an integral involving each of the core state wave functions, and instead of the Mattheiss prescription he instead pushes any radial charge density back into the muffin-tin, increasing the amplitude of the outermost orbital to enforce charge neutrality.

In both cases the incident electron is assumed to have no effect on the core electrons.

Van Hove uses a relativistic self-consistent code to solve for the free atom, and then uses an approach similar to the uwm method above to construct the potential.

Changes to existing code:

Factory pattern for Atom – since there are multiple ways to compute the self-consistent potential for a free atom (the starting point for most muffin-tin calculations), I should make the constructor for Atom private and expose a factory method Atom.NewAtom(method) as Atom (or IAtom), and subclass Atom for the different computational methods.

Most general procedure:

Step 1 – define the cluster for the computation,

Atom ID, Element, Configuration (optional), X, Y, Z coordinates

Step 2 – identify unique bonding sites (inequivalent sites) by computing number and (scalar) distance to neighboring atoms, by element.

Step 3 – for each unique site, compute Rmt, defaulting to ionic radius in ambiguous cases.

Step 4 – Solve each unique atom – ideally self-consistently using Rmt, or for free atom is this is complicated. – can make polymorphic for different algorithms, either pushing density back onto the atom self- or non-self consistently, or using the Mattheiss prescription, etc. 🡪 conceptually tie up the self-consistent calculation with the cluster.

Step 5 - Use atomic wave functions to compute charge density and potential for incident electron – this may be able to be done as part of the self-consistent calculation?

Step 6 – Solve for the incident electron and compute phase shift for each angular momentum L, at a range of energies – produce a table of energies. Final phase shifts as a function of E are interpolated on that table.

Step 7 -

I should be able to calculate the phase shifts as follows:

1. Instead of self-consistently solving the free atom, I should specify Rmt (if Rmt = infinity then I should get the free atom result). On each iteration, fold back in any charge density outside of Rmt by increasing the core state wave function amplitude. The integral in HS for the charge density contribution to potential should end at Rmt instead of infinity. Therefore the potential should go to zero at Rmt. At the same time, compute the phase shift for the incident electron of

The core of the issue – I don’t know if the potential computed during the HS self-consistent calculation is the proper one to use for the incident electron or not? It’s hard to imagine why it wouldn’t be… Except for the Latter tail? Need to study up on this.

Technically the atom is a negative ion…

Why doesn’t HS work for negative ions?

Let’s do this:

1. Define the cluster
2. Identify inequivalent sites and Rmt
3. For each site, compute rho(r) from free atom wave functions
4. Solve Poisson’s equation to get the V(r) from electronic wave functions, or compute using essentially the formula of Herman and Skillman.
5. Add nuclear potential and exchange to V(r).
6. Combine with nearest neighbor potentials using Mattheis’ prescription.
7. Solve for incident electron in V(r)

Cluster

Atom ID Element x y z

# Constructing the muffin-tin potential:

We will follow the method outlined for the MUFPOT program, <http://hermes.phys.uwm.edu/projects/elecstruct/mufpot/MP/MP.Theory1.html>.

The strategy with this method is to start with the tabulated wave functions of Herman and Skillman, compute from those the electron density , and then to solve the Poisson equation to determine the potential due to that charge density. The resulting electron potential is added to the nuclear coulomb potential and an exchange term (Slater approximation with statistical factor), and then the Mattheiss prescription is invoked to construct the muffin-tin.

The starting point is the set of free atom radial wave functions tabulated by the Atom class. The (spherically symmetric) electron density can be written:

|  |  |  |
| --- | --- | --- |
|  |  |  |

Note that this expression assumes the occupancy for a closed subshell (2l + 1).

It is relatively straightforward to compute the total radial electronic charge density from the Herman Skillman wavefuntions encapsulated in the Atom class.

* Atom should expose a method to retrieve rho(r).

Once we’ve got rho, the next step is to compute the contribution to the scattering potential due to it:

|  |  |  |
| --- | --- | --- |
|  |  |  |

Where the first term is the contribution from the nucleus and the second term is the electronic contribution defined by:

|  |  |  |
| --- | --- | --- |
|  |  |  |

Since and depend only on the radial coordinate r, we need only the radial part of the Laplacian. This allows us to redefine Eq. 13 to

|  |  |  |
| --- | --- | --- |
|  |  |  |

For a given , this equation can be solved for .

Complication – what about near the origin? Do we work instead with r \* rho and r \* u?

And I don’t think we need to do this anyway – I think the potential as given by Herman and Skillman \*is\* the right one to use.

Following the Pendry way (folding back in charge to ensure neutrality) – here’s one way to do it. When solving self-consistently for the atom with a specified Rmt, bump up the “occupancy” of any orbitals which have (significant) charge density outside of Rmt in such a way that the integrated charge density from 0 to Rmt is the same as the free atom. Inject this into the self-consistent loop and solve, and then simply cut off Rho at Rmt for the potential.

Great care must be taken here to ensure that the latter tail correction works as it should. I tend to think it’ll be “good enough.”

Now I believe that the Herman Skillman potential is the right one to use EXCEPT for the fact that the exchange term in HS compensates for the self-charge of the single electron. The exchange potential in Herman and Skillman is:

|  |  |  |
| --- | --- | --- |
|  |  |  |

In comparison, the exchange term from Pendry is

|  |  |  |
| --- | --- | --- |
|  |  |  |

Where alpha is the statistical exchange parameter and is about 0.7.

Now I wonder if the extra factor of two in Herman-Skillman is due to the units used? Yes, it is – according to the documentation of the HS code,

|  |  |  |
| --- | --- | --- |
|  |  |  |

And in Rydberg units, *e*2 = 2. SO – it appears that the HS potential IS the right one to use, excepting that it does not include the effect of the incident electron on all of the others – which none of the methods does comprehend.

Path Forward:

1. Complete table of results for free atoms
2. Troubleshoot my energy issues for high N
3. Modify routine to accept Rmt as a parameter – OR – adapt code for Atom 🡪 MuffinTin.
   1. Can make MuffinTin accept as input the solution of the free Atom.
   2. Use the orbitals in Atom as starting points for the self-consistent solution.
   3. This is the cleanest separation and preserves abstraction so that other MuffinTin implementations (e.g., using Poisson’s equation) can be done using the same interface(s).

# The effect of the inner potential

One thing I thought about this morning was how to best include the effect of the optical potential (attenuation) in the initial scattering event for bulk atoms far from the surface? In reality, the optical potential takes effect as soon as we traverse the inner potential boundary, at some distance from the surface. How do we properly account for this?

# Cluster

The Cluster class is a collection of MuffinTin objects. Implement using a Flyweight pattern, since most constituent atoms of the cluster are identical muffin-tins.

Cluster is made up of Atoms (not to be confused with the HS Atom class) and bonds. Bonds provide information about how Atoms are connected to each other, and will be useful when computing muffin-tin potentials in the Mattheiss prescription.

On instantiation:

* Solve for muffin tins and compute (real) phase shifts
* Strategy – compute phase shifts up front or on-the-fly with caching? Would be internal to muffin-tin.
* The optical potential is the imaginary part of the energy.

Constructing MuffinTin

* Can be done in several ways
* First way – use Pendry’s method of folding charge from the atomic wave functions back over into the muffin-tin. This requires the Atom() solution (as a starting point) or maybe not – just modify Atom to include the self-consistent part.
* Second way – Mattheiss prescription – requires knowledge of bonds and nearest-neighbor potentials/distances.
* SO – with that second constraint, it makes sense to give the Cluster object the responsibility to construct MuffinTins and feed whatever data is needed into