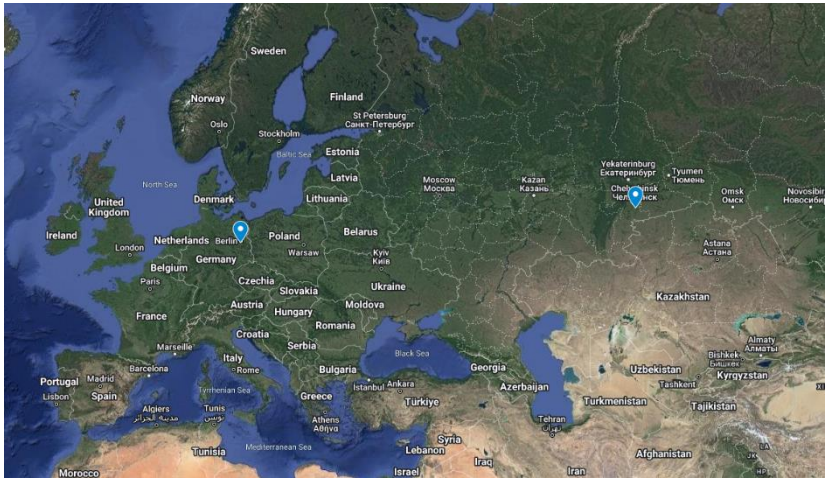
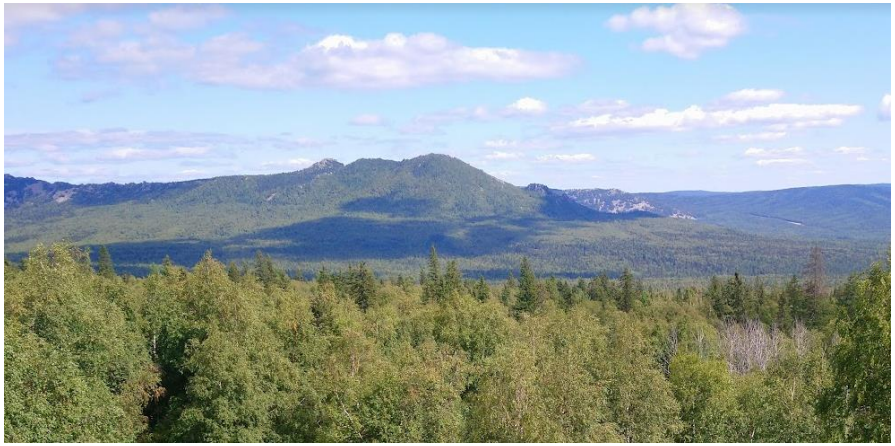
Minimal basis solver for FHI-aims

or the rant about documentation

Andrei Sobolev, MS1P e.V.

Let me introduce myself…

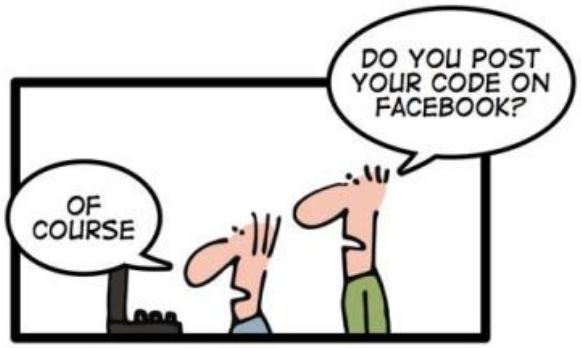


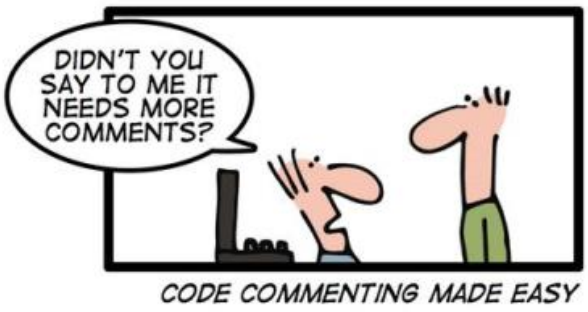


Let me introduce myself…

The outline

- Quantum Mechanics 101
  - The concepts and how they are used in AIMS
- Minimal basis solvers in AIMS
  - Yes, there are several
- The rant about documentation
  - We need more comments!





QM 101

- **Schrödinger equation (in B-O approximation):**

$$\hat{H}\Psi = E\Psi$$

$$\hat{H} = \sum_k \frac{p_k^2}{2m_e} + \sum_{I,k} \frac{Z_I}{2|\vec{R}_I - \vec{r}_k|} - \sum_{k \neq k'} \frac{1}{2|\vec{r}_k - \vec{r}_{k'}|}$$

- **Density functional theory** (rewrite everything in terms of electron density):

$$E_{tot} \leq \langle \Psi | \hat{H} | \Psi \rangle; \Psi = \Psi[n]; \exists n_0 : E_{total} \rightarrow min$$

$$E_{tot} = E[n] = T[n] + V[n] + V_{es}[n] + E_{xc}[n]$$

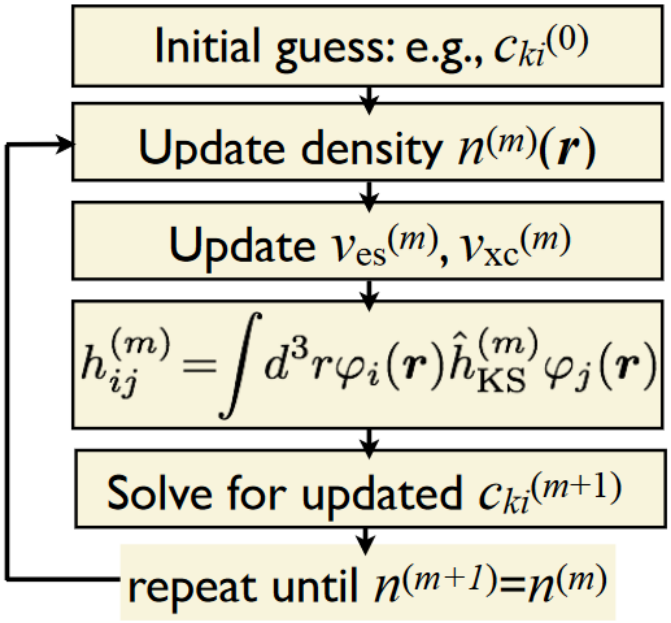
- **Kohn-Sham equations:**

$$\left[-\frac{\nabla^2}{2} + v(\vec{r}) \right] \psi_k(\vec{r}) = \epsilon_k \psi_k(\vec{r})$$

$$\sum_k |\psi_k(\vec{r})|^2 = n(\vec{r}); \;\; v(\vec{r}) = v_{ei}(\vec{r}) + v_{es}(\vec{r}) + v_{xc}(\vec{r})$$

More QM 101

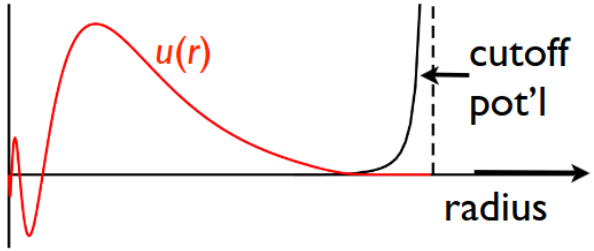
- **Solving Kohn-Sham equations:**
  - Rewrite them in terms of basis decomposition coefficients in matrix form; solve selfconsistently:



$$\psi_k(r) = \sum_i c_{ki}\phi_i(r)\,; \qquad \sum_i c_{ki}h_{ji} = \sum_i \epsilon_k c_{ki}s_{ji}$$

- **An all-electron electronic structure code based on numeric atom-centered orbitals (NAOs)**
- **NAOs:**

$$\phi_{i[lm]}(\vec{r}) = \frac{u_i(r)}{r} \cdot Y_{lm}(\Omega)$$



- $u_i(\vec{r})$ **– tabulated**
  - The choice allows to subdivide KS equations into radial and angular parts and simplify the radial equation:

$$\left[-\frac{1}{2}\nabla^2 + \frac{1}{2}\frac{l(l+1)}{r^2} + v(r) + v_{cut}(r)\right] u_i(r) = \epsilon_i u_i(r)$$

Basis in FHI-aims

- **Minimal + "Tiers"**
  - Minimal basis = occupied free atom orbitals $\{u\}^{(0)}$
  - Tiers composed of ionic (2+) and hydrogen-like functions
    - Ionic 2+ wavefunctions needed!

```
#######################################################################################
#
#  Definition of "minimal" basis
#
#######################################################################################
#     valence basis states
    valence      2  s   2.
    valence      2  p   2.
#     ion occupancy
    ion_occ      2  s   1.
    ion_occ      2  p   1.
#######################################################################################
#
#  Suggested additional basis functions. For production calculations,
#  uncomment them one after another (the most important basis functions are
#  listed first).
#
#  Constructed for dimers: 1.0 A, 1.25 A, 1.5 A, 2.0 A, 3.0 A
#
#######################################################################################
#  "First tier" - improvements: -1214.57 meV to -155.61 meV
     hydro 2 p 1.7
     hydro 3 d 6
     hydro 2 s 4.9
#  "Second tier" - improvements: -67.75 meV to -5.23 meV
#     hydro 4 f 9.8
#     hydro 3 p 5.2
#     hydro 3 s 4.3
#     hydro 5 g 14.4
#     hydro 3 d 6.2
#  "Third tier" - improvements: -2.43 meV to -0.60 meV
```

More QM 101: Zeroth order regular approximation

- Relativistic Dirac equation:

$$\begin{pmatrix} V & c\boldsymbol{\sigma}\cdot\boldsymbol{p} \\ c\boldsymbol{\sigma}\cdot\boldsymbol{p} & -2c^2+V \end{pmatrix}\begin{pmatrix}\phi\\ \chi\end{pmatrix} = \epsilon\begin{pmatrix}\phi\\ \chi\end{pmatrix}$$

- Eliminate small component:

$$\hat{\mathrm{H}}^{\mathrm{esc}}\phi = \left(V + \boldsymbol{\sigma}\cdot\boldsymbol{p}\frac{c^2}{2c^2+\epsilon-V}\boldsymbol{\sigma}\cdot\boldsymbol{p}\right)\phi = \epsilon\phi$$

- Put Pauli matrices in, rewrite:

$$\left(V + \boldsymbol{p}\frac{c^2}{2c^2+\epsilon-V}\boldsymbol{p} + i\boldsymbol{p}\frac{c^2}{2c^2+\epsilon-V}\times\boldsymbol{p}\cdot\boldsymbol{\sigma}\right)\phi = \epsilon\phi$$

Scalar relativity term
(for elements with Z > 20)

Spin-orbit coupling
(for heavy elements)

ZORA

- Expand $\hat{H}^{esc}$ in $\frac{1}{2c^2-V}$, take zeroth expansion order

$$\hat{H}^{zora} = V + \boldsymbol{p}\frac{c^2}{2c^2 - V}\boldsymbol{p}$$

- Substitute $\hat{H}^{zora}$ for Hamiltonian in Schrödinger equation, do the math once again, arrive to the following radial KS equation:

$$\left[-\frac{c^2}{2c^2 - V}\nabla^2 + \frac{c^2}{2c^2 - V}\frac{l(l+1)}{r^2} + v(r) + v_{cut}(r)\right]u_i(r) = \epsilon_i u_i(r)$$

- Van Wüllen, 1999: Substitute V for $V_{free\ atom}$ ("atomic ZORA")
  - Regains gauge invariancy
  - Removes unphysical forces in ZORA calculations

Minimum basis solvers in aims

- **SRATOM**
  - A self-consistent solver of radial Schrödinger equation with scalar relativity on a logarithmic grid
- **AtomSphere**
  - SCF free atom solver based on the (unpublished) work done by Stefan Goedecker (seems that it was initially the part of ABINIT?)

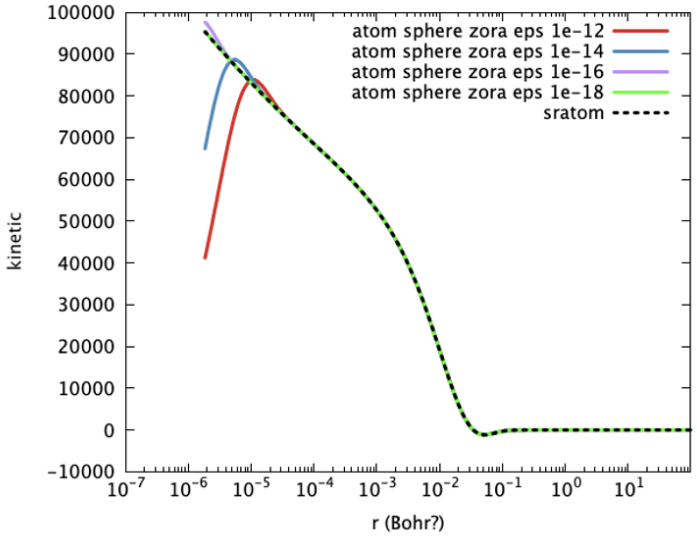
AtomSphere

```
- I don't understand how this subroutine works, I don't have the time to understand
  it, and they're not paying me to understand it.  You're on your own, kid!
```

- **Uses xclib interface**
- **Scalar relativity was initially not implemented**
  - Work done by Rungdong Zhao and Yi Yao
- **Also uses logarithmic grid**
  - But different that that in aims: $r_i = r_0(e^{\alpha i} - 1)$

Work done by Yi Yao

	minimal	Time(s)	
atom_sphere_zora_eps 1e-12	-205851.160476344	6.3s	1s kinetic
atom_sphere_zora_eps 1e-14	-205851.166311959	9.2s	
atom_sphere_zora_eps 1e-16	-205851.168340841	22.5s	Xe atom
atom_sphere_zora_eps 1e-18	-205851.168544115	69.0s	
sratom	-205851.169346916		



To summarize my modification:

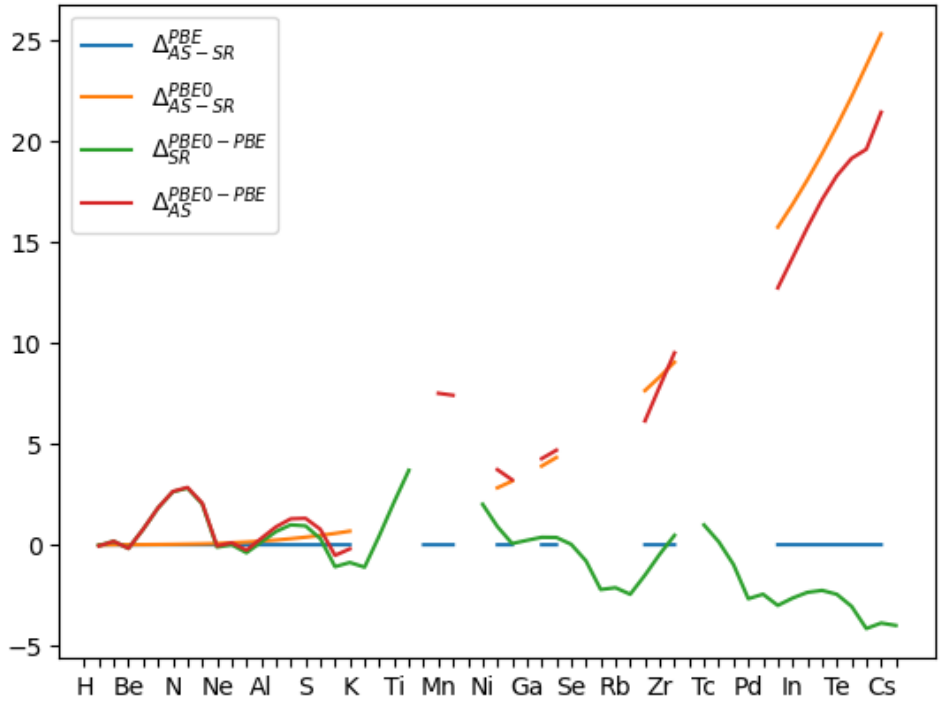
1. the atom_sphere+atomic_zora works.
2. I optimized the atom_sphere zora solver with better initial guess for relativistic potential. A single convergence parameter atom_sphere_zora_eps is introduced in the control.in file. I set the default to be 1e-14 for now.

Wish list:

1. initial guess (both wavefunctions and relativistic potential from sratom PBE).
2. extensive tests for different atoms/xc…

Checking if AtomSphere really works…

The difference between free atom energies got from different minimum basis solvers

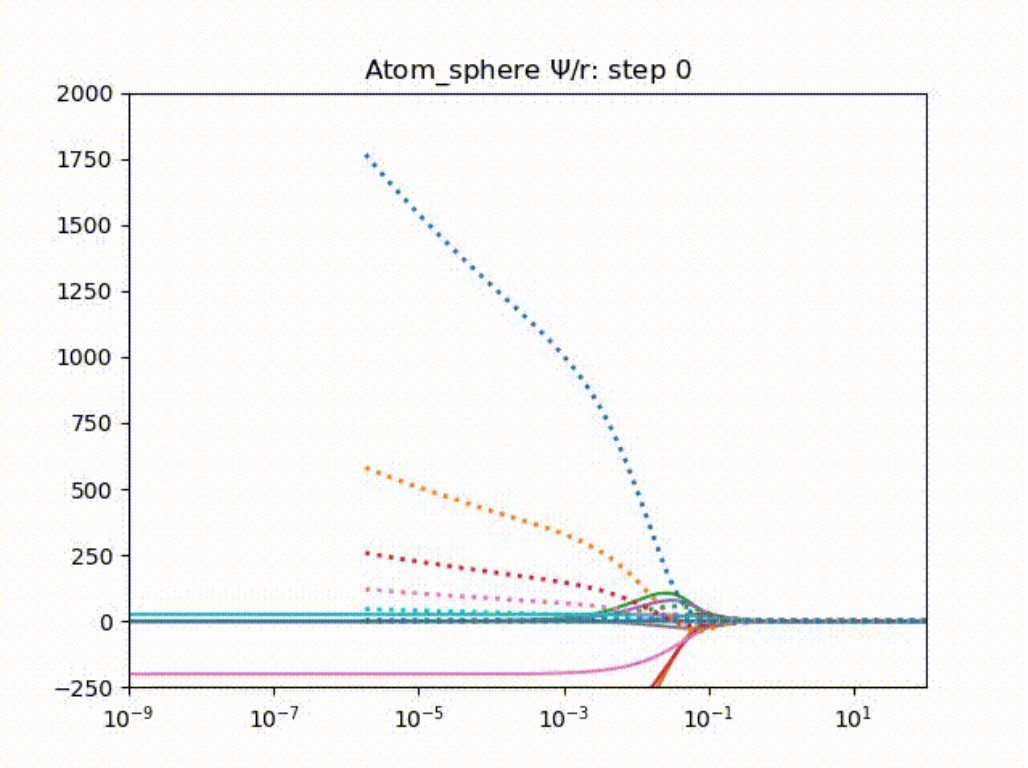


- **It seems to work!**
  - but only for PBE XC potential
  - and for selected elements
- For hybrid PBE0 AtomSphere gives up to 20 eV higher energy than SRATOM foe heavy elements

Putting SR WFs into AS

- AtomSphere wavefunctions (compared to SRATOM) for Xe atom; starting point $\phi_0(r) = r^l e^{-0.9 Z_{eff} r}$

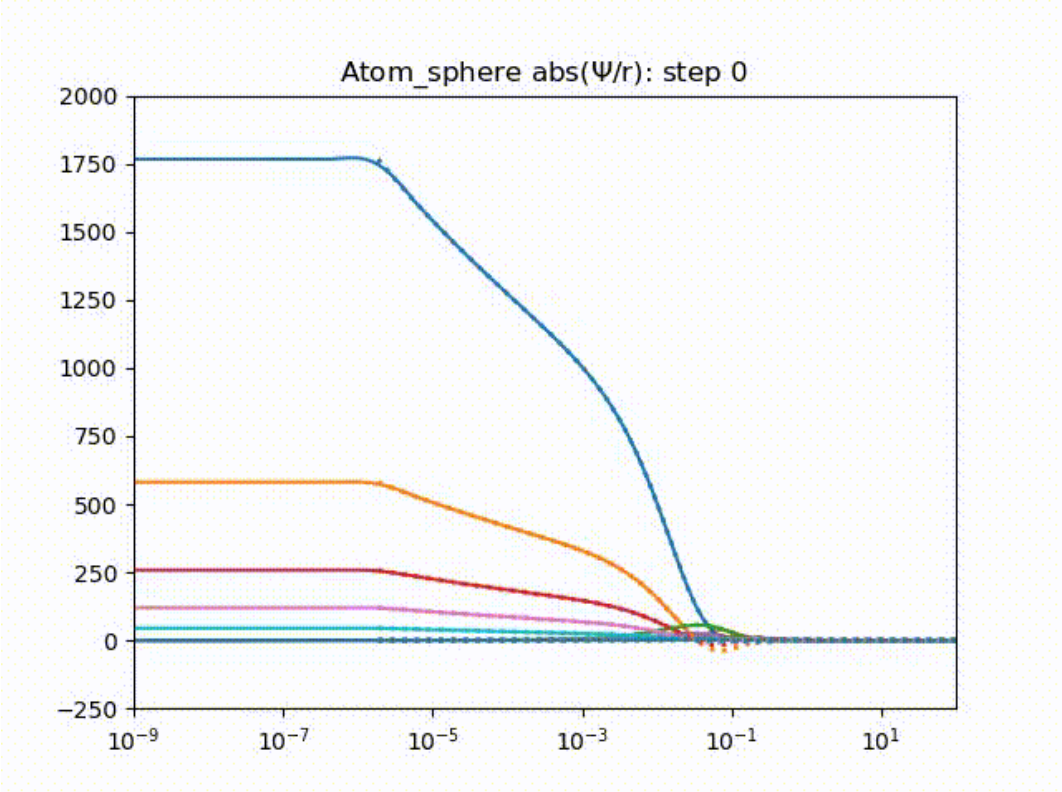
- The wavefunctions converge to the SRATOM counterparts



Putting SR WFs into AS

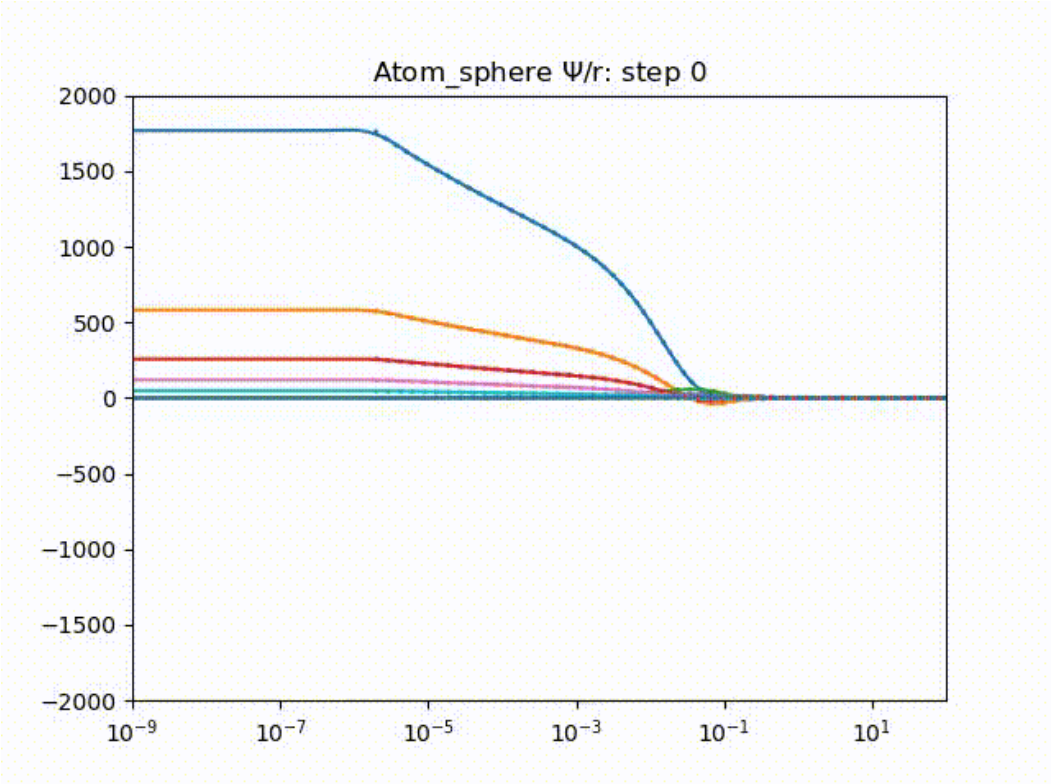
- AtomSphere wavefunctions (compared to SRATOM) for Xe atom; starting point: SRATOM wavefunctions

- The wavefunctions *do not* converge to the SRATOM counterparts



Putting SR WFs into AS

- AtomSphere *ionic* wavefunctions (compared to SRATOM) for Xe atom; starting point: SRATOM wavefunctions
- The wavefunctions *diverge*



Let's have a glimpse at the code

Language	files	blank	comment	code
Fortran 90	71	4953	8632	335697
Fortran 77	21	230	3496	8041
CMake	5	7	15	102
C/C++ Header	2	5	0	53
SUM:	99	5195	12143	343893

- **~344k lines of code**
  - 8k in F77 | 336k in F90
- **12k lines of comments**
  - 3.5k in F77 | 8.5k in F90
- **Comment ratio: 44% in F77 | 2.5% in F90**

Why writing documentation is important?

For you

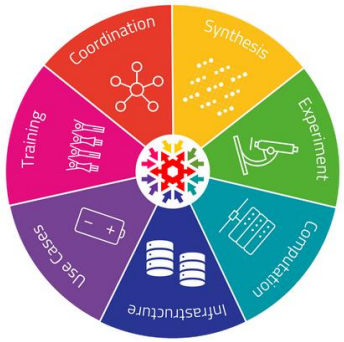
- You will be using your code in 6 months
- You want people to use your code and give you credit

For others

- Others will use your code and build upon it
- Others will be encouraged to contribute to your code

For science

- Encourage open science
- Allow reproducibility and transparency



Good practices when commenting the code

- Make comments meaningful

```
x += 1;  // increment x
END;  #  END
x = x * 1.1221432243293; /* Joes fiddle factor don't know what it is
but it doesnt work without it */
a = (INT64)b * (INT64)c; /* INT64 = INT32 * INT32 */
```

- Try to answer questions: "*What has been done?*", "*What needs to be done*"; "*Why it needs to be done?*"

```
1 //Close the gate
2 gate.close()
```

Bad

```
1 //Enter isolation mode: (Close gates, shutdown power, activate beacon)
2 //Step 1: Close all gates but the emergency spill gate.
3 gate.close()
```


Good

Good practices when commenting the code

- Good code still needs documenting
  - And it's almost impossible to write a good code in FORTRAN
- Use **`implicit none`**
- Write the equation (or the link) and the numerical method that is used to solve it

```
      if (nang.eq.12) then
! A. H. Stroud, Prentice Hall 1971, page 296
! integrates up to order 5 (.eq. up to x^5 or x^1*y^2*z^2)
```

- Code and comments should be written at the same time

```
            r1=rnum1/rdenom1
            call detnp(nrad,rr,r1,nmod(iprin,l+1,isp))
            !! UNCOMMENT TO SET VIRTUAL ORBITALS
         else
            if (.not.cut_atom.and.rprb.gt.0.d0) then
               r2=rprb*rnum2
               call detnp(nrad,rr,r2,nmod(iprin,l+1,isp))
```

Back to AtomSphere

The comments:

```
!! SETTING UP DIFFERENT SEGMENTS
! Finite nucleus Coulomb potential
!! confinement potential
! Add parabolic confining potential
!!!!!FIXME: Building -1/2 d^2/dr^2 -
Z/r + l(l+1)/2r^2
!GS Orthogonalization of psi
```

Setting up

```
!! SOLVING SCF CYCLE
! if there is no convergence within 50
steps of the 2000 loop the shift are
presumably not optimal and they will be
set equal to the eigenvalues of the
previous 3000 loop
! calculate charge densities
! check normalization
! calculate gradient
! calculate unconstrained gradient
! hh (kinetic energy + local potential)
times psi
! Add hartree contribution
! Add XC terms
```

Energy and potential calculation

```
! Since the Hartree potential is
calculated we can now add the core
charge
! This (vxc) is needed only for
checking purposes
! Add Hartree Fock terms
! Add hartree and XC contributions to
gradient
! calculate Lagrange multiplier matrix


!! THIS PART DECIDES THE VALUE OF NMOD
FOR EACH ORBITAL
! transform psi, grad and grads to
canonical ones
! add constraints to gradient
! residues
! precondition gradient
! Update wavefunction
! set up DIIS matrix (lower triangle)
! calculate new line
! copy to work array, right hand side,
boundary elements
! solve linear system
! update wavefunction
! orthogonalize wavefunctions
!! End of SCF
```

WTF?

Advance SCF

Look at the precondition gradient

```
! precondition gradient
   if (ids.eq.1) then ! Since preconditioner
is fixed, it has to be calculated only in
first iteration
     do l=0,lmax
       do iprin=1, max(nprin(l+1,1),
nprin(l+1,2))
         do j=1, nrad
           pot0(j,iprin,l+1)=shift(iprin,l+1)
 !use constant local potential
           if (cut_atom) then
             pot0(j,iprin,l+1)=pot0(j,iprin,
l+1) + cutoff_pot( rr(j), cutoff_type, r_cut,
w_cutoff, scale_cutoff )
           else if (rprb.gt.0.d0) then
             pot0(j,iprin,l+1)=pot0(j,iprin,l+
1)+.5d0*(rr(j)/rprb**2)**2
           end if
         enddo
         enddo
         enddo
         z=0.d0 ! local potential entirely
contained in potloc
         call crthhp(nspol,nprinx,nprin,
nrad,lmax,rr,z,pot0,hhp)
       endif
```

```
!** TRIDIAGONAL SYMMETRIC
PRECONDITIONING HAMILTONIAN MATRIX
*****************!
 subroutine crthhp(nspol,nprinx,
nprin,nrad,lmax,rr,znuc,pot0,hhp)
!  Calculates a tridiagonal
symmetric preconditioning
hamiltonian matrix hhp in a basis
of linear finite elements
 implicit real*8 (a-h, o-z)
  dimension  rr(nrad),hhp(2,nrad,
nprinx,lmax+1),pot0(nrad,nprinx,
lmax+1),nprin(lmax+1,nspol)
```

Preconditioner

- Solving Schrödinger equation for orthogonal basis is equivalent to minimizing the constrained gradient

$$\vec{g} = H\vec{u} - \epsilon\vec{u}$$

- Eigenvalues converge faster then eigenvectors; let $\vec{u}$ be approximate eigenvector and $\vec{u} + \vec{p}$ be the true eigenvector; then $\vec{p}$ is found as

$$\vec{p} = (H - \epsilon I)^{-1}\vec{g}$$

and at each step $\vec{u}_i$ can be found as

$$\vec{u}_i = \vec{u}_{i-1} - t\vec{p},$$

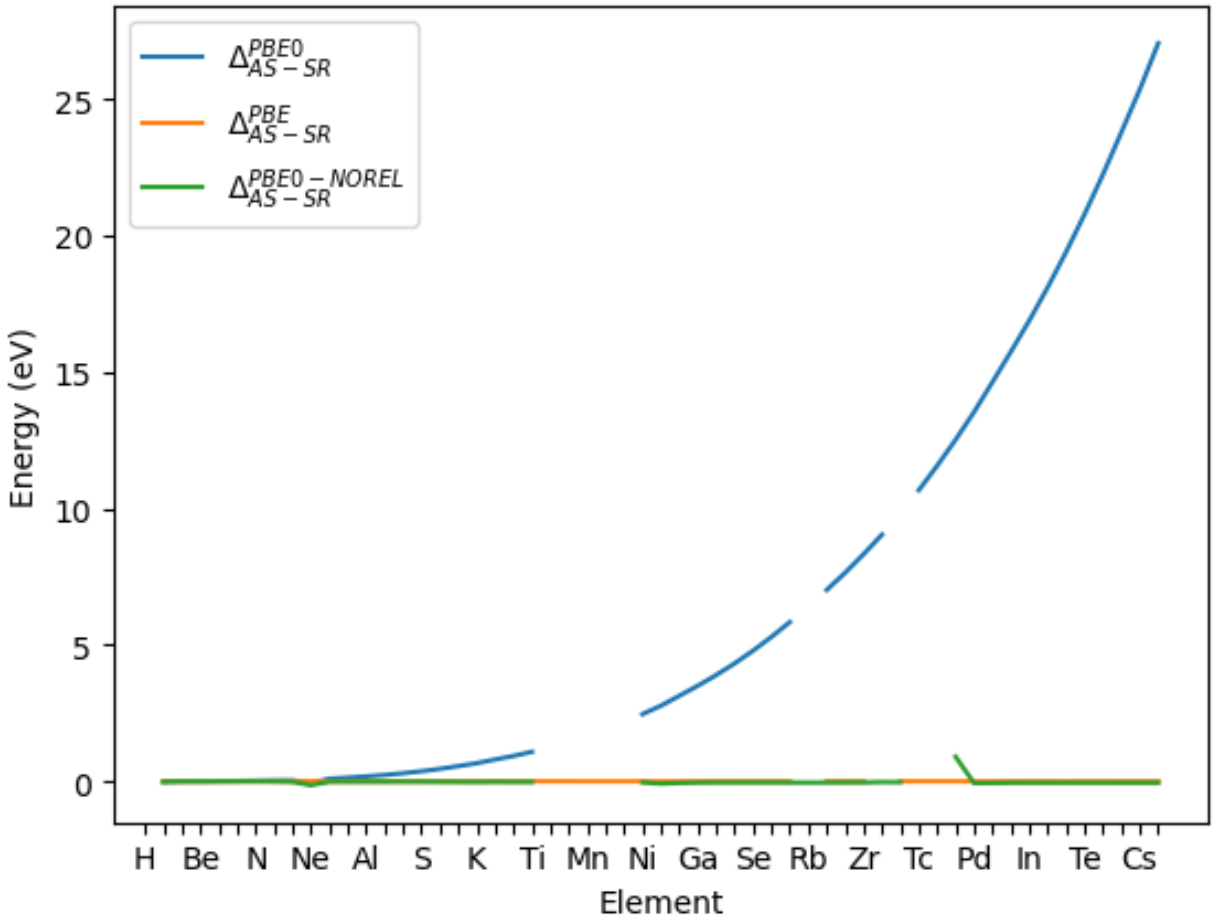
where t is of the order of 1.

Solution to the AtomSphere problem

- Change the Hamiltonian preconditioner matrix to include ZORA terms

```
2394  2394          do 3092,i=1,nrad-1
2395        -       aa=const*(rr(i+1)-rr(i))
      2395  +       ! aa=const*(rr(i+1)-rr(i))
      2396  +       aa=damprel(i+1)*l*(l+1)*.333333333333333d0*(rr(i+1)-rr(i))
2396  2397          r1=r2
2397  2398          r2=rr(i+1)
2398  2399          dr=r2-r1
...   ...     @@ -2401,7 +2402,8 @@ end subroutine detnp
2401  2402          r1r2=r1*r2
2402  2403          p=q
2403  2404          q=pot0(i+1,iprin,l+1)
2404        -       tt=(r1q + r1r2 + r2q)/dr*.16666666666666d0
      2405  +       ! tt=(r1q + r1r2 + r2q)/dr*.16666666666666d0
      2406  +       tt=damprel(i+1)*(r1q + r1r2 + r2q)/dr*.33333333333333d0
2405  2407          ss=dr*(10.d0*p*r1q + 2.d0*q*r1q + 4.d0*p*r1r2 +   &
```

Results



Hybrids still do not work correctly (reason – nonlocal free atom potential)
Also, several elements did not converge

Conclusions

- **AtomSphere works now (sort of…)**
  - Hybrids are not working as the free-atom potential is non-local
  - MetaGGAs still need some time to be implemented

- **My advice to you: Comment your code**
  - But comment wisely ☺

PS

- Just a reminder, we have GIMS which is also being actively developed
  - and we desperately need the feedback!

https://gims-dev.ms1p.org