# Solving Many-Body Schrödinger Equation Using Density Functional Theory and Finite Elements

Ondřej Čertík, Jiří Vackář, Miroslav Tůma

Institute of Physics, Academy of Sciences of the Czech Republic

June 21, 2008

# Introduction

Contens

- Density Functional Theory
- Spherically symmetric problems, examples
- Mixing schemes
- Non symmetric 3D problems, Finite Element Method

# Schrödinger equation

$$\hat{H}\,|\Psi\rangle = (\hat{T} + \hat{U} + \hat{V})\,|\Psi\rangle = E\,|\Psi\rangle$$

where

$$\hat{T} = \sum_{i}^{N} -\frac{1}{2}\nabla_i^2$$

$$\hat{U} = \sum_{i<j} U(\mathbf{r_i}, \mathbf{r_j}) = \frac{1}{2}\sum_{i,j} U(\mathbf{r_i}, \mathbf{r_j})$$

$$U(\mathbf{r_i}, \mathbf{r_j}) = U(\mathbf{r_j}, \mathbf{r_i}) = \frac{1}{|\mathbf{r_i} - \mathbf{r_j}|}$$

$$\hat{V} = \sum_{i}^{N} v(\mathbf{r_i})$$

$$v(\mathbf{r_i}) = \sum_{k} -\frac{Z_k}{|\mathbf{r_i} - \mathbf{R_k}|}$$

We solve the Kohn-Sham equations:

$$(-\tfrac{1}{2}\nabla^2 + V_H(\mathbf{r}) + V_{xc}(\mathbf{r}) + v(\mathbf{r}))\psi_i(\mathbf{r}) = \epsilon_i\psi(\mathbf{r})$$

that yield the orbitals $\psi_i$ that reproduce the density $n(\mathbf{r})$ of the original interacting system

$$n(\mathbf{r}) = \sum_i^N |\psi_i(\mathbf{r})|^2$$

$$V_H(\mathbf{r}) = \frac{\delta E_H}{\delta n(\mathbf{r})} = \tfrac{1}{2}\int \frac{n(\mathbf{r}')}{|\mathbf{r}-\mathbf{r}'|}\mathrm{d}^3 r' \iff \nabla^2 V_H = n(\mathbf{r})$$

$$E_{xc}[n] = (T+U)[n] - E_H[n] - T_S[n]$$

$$V_{xc}(\mathbf{r}) = \frac{\delta E_{xc}[n]}{\delta n(\mathbf{r})}$$

$$v(\mathbf{r}) = \sum_k -\frac{Z_k}{|\mathbf{r}-\mathbf{R_k}|}$$

# Atomic wavefunctions

Spherically symmetric potential:

$$V(\mathbf{x}) = V(r)$$

$$\psi_{nlm}(\mathbf{x}) = R_{nl}(r)\, Y_{lm}\left(\frac{\mathbf{x}}{r}\right)$$

Radial Schrödinger equation:

$$R''_{nl} + \frac{2}{r} R'_{nl} + \frac{2M}{\hbar^2}(E - V)R_{nl} - \frac{l(l+1)}{r^2} R_{nl} = 0$$

# Relativistic atomic wavefunctions

Dirac equation:

$$(ic\gamma^\mu D_\mu - mc^2)\psi = 0$$

$$D_\mu = \partial_\mu + ieA_\mu$$

Radial Dirac equation:

$$g_\kappa'' + \left(\frac{2}{r} + \frac{V'}{2Mc^2}\right) g_\kappa' + \left[(E - V) - \frac{\kappa(\kappa+1)}{2Mr^2} + \frac{\kappa+1}{4M^2c^2r}V'\right] 2Mg_\kappa = 0$$

$$f_\kappa = \frac{g_\kappa'}{2Mc} + \frac{\kappa+1}{r}\frac{g_\kappa}{2Mc}$$

$$R^2 = f^2 + g^2$$

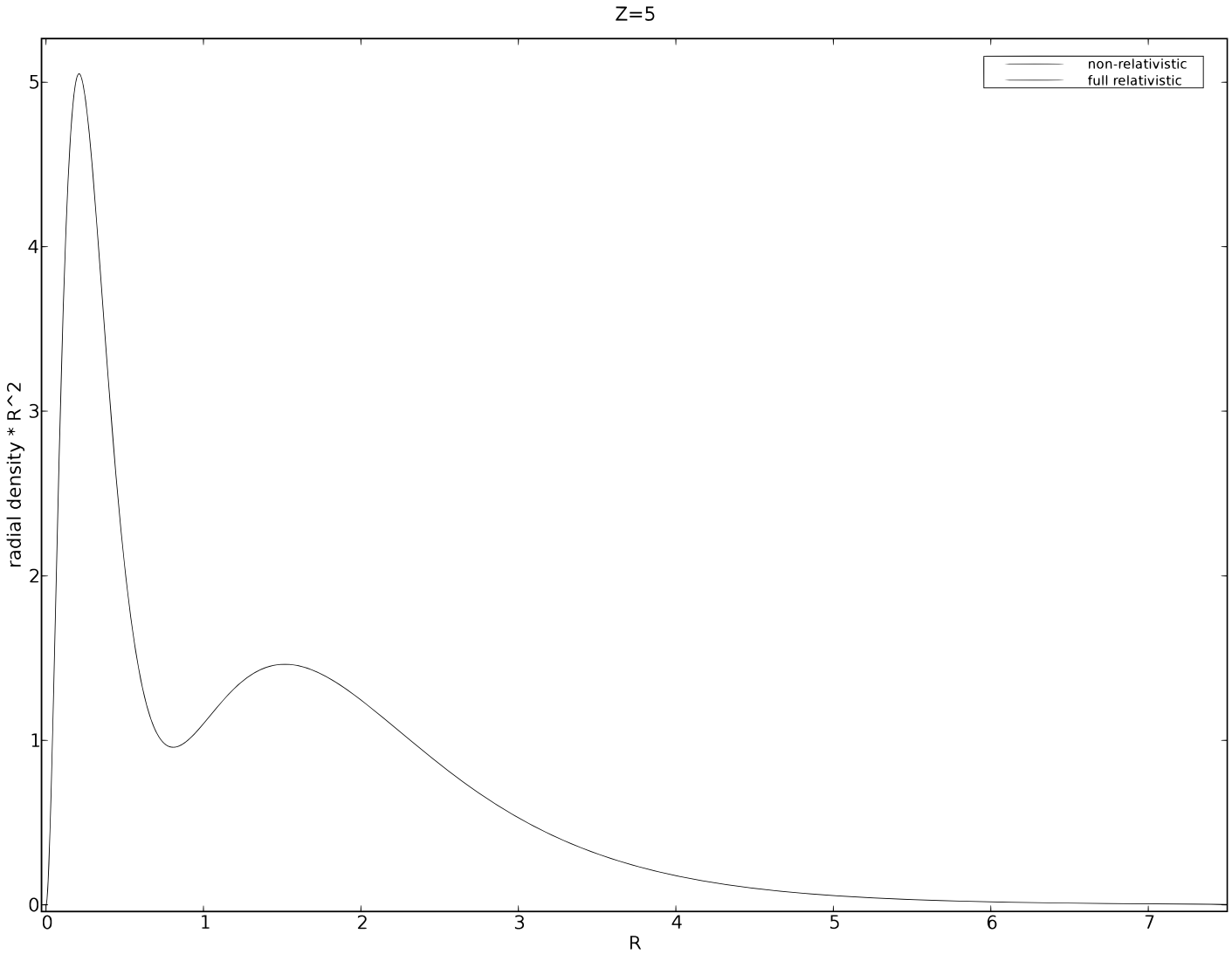
Radial Schrödinger equation:

$$R'' + \frac{2}{r}R' + \left[(E - V) - \frac{l(l+1)}{2Mr^2}\right] 2MR = 0$$

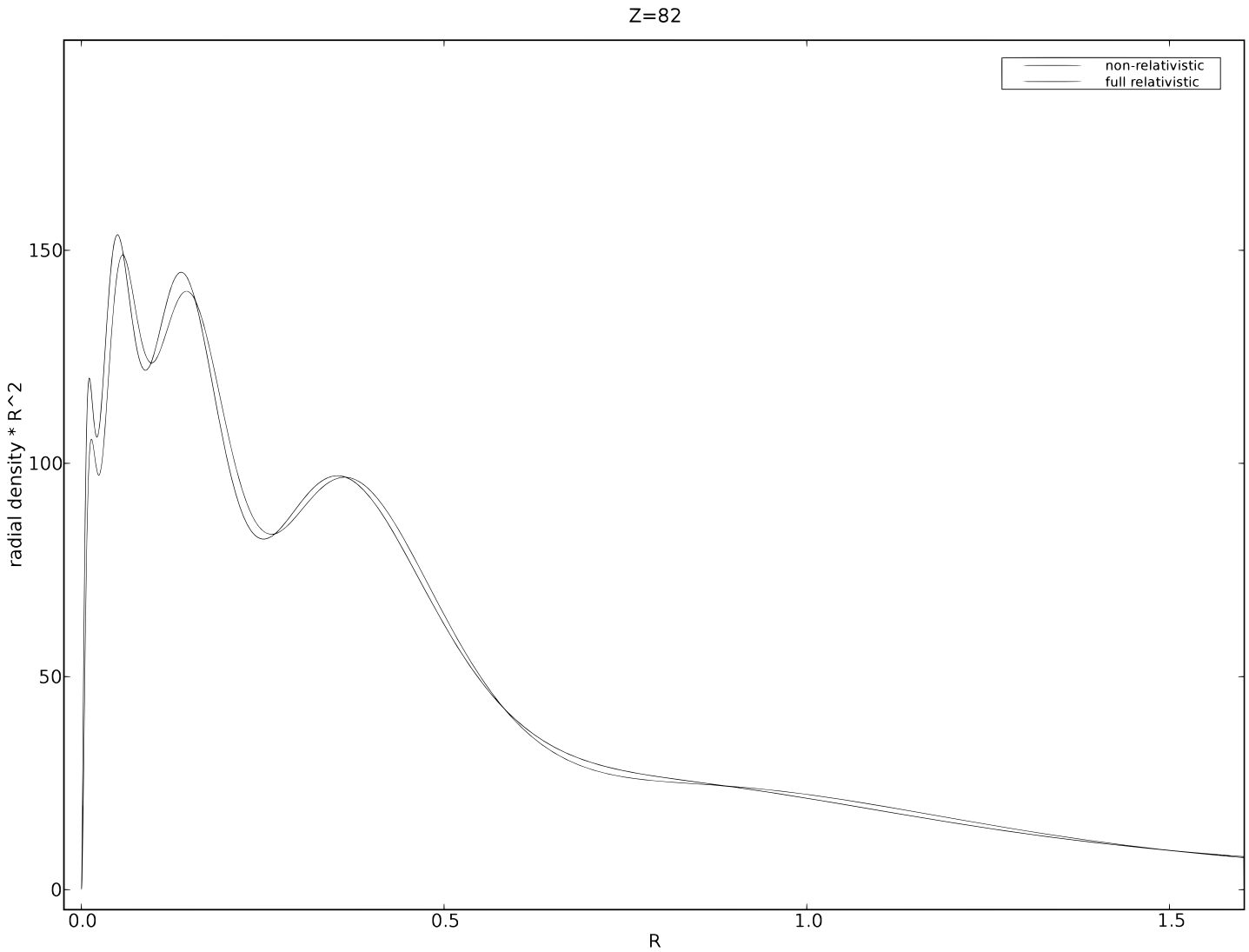
# Code (300 lines in Python, 800 in Fortran)

```
from atom import atom, show
import radial
import utils
def do(Z):
    R = radial.create_log_grid(Z)
    s = atom(Z,alpha=0.3,iter=20,relat=0,grid=R)
    n5_lda = radial.KS_construct_density(s,R,Z)*R*R
    s = atom(Z,alpha=0.3,iter=20,relat=2,grid=R)
    n5_rlda = radial.KS_construct_density(s,R,Z)*R*R
    utils.makeplot(R,[
        (n5_lda,"b-","non-relativistic"),
        (n5_rlda,"g-","full relativistic"),
        ],title="Z=%d"%(Z),xleg="R",
        yleg="radial density * R^2")
do(5)
#do(82)
```

# Boron


Z=5
non-relativistic
full relativistic
radial density * R^2
R
0
1
2
3
4
5
6
7

# Lead


Z=82
non-relativistic
full relativistic
radial density * R^2
R
0
50
100
150
0.0
0.5
1.0
1.5

# Lead - nonrelativistic calculation

Iterations: 20
$|F(x)| = 0.00003516$
Agrees with NIST:
http://physics.nist.gov/

```
1s( 2): -2901.078061
2s( 2): -488.8433352
2p( 6): -470.8777849
3s( 2): -116.526852
3p( 6): -107.950391
3d(10): -91.88992429
4s( 2): -25.75333021
4p( 6): -21.99056413
4d(10): -15.03002657
4f(14): -5.592531664
5s( 2): -4.206797624
5p( 6): -2.941656967
5d(10): -0.9023926829
6s( 2): -0.3571868295
6p( 2): -0.1418313263
```

# Lead - relativistic calculation

Iterations: 20
$|F(x)| = 0.00000584$

```
1s( 2) j=l+1/2: -3209.51946
2s( 2) j=l+1/2: -574.1825655
2p( 6) j=l-1/2: -551.7234408
2p( 6) j=l+1/2: -472.3716103
3s( 2) j=l+1/2: -137.8642241
3p( 6) j=l-1/2: -127.6789451
3p( 6) j=l+1/2: -109.9540395
3d(10) j=l-1/2: -93.15817605
3d(10) j=l+1/2: -89.36399096
```

```
4s( 2) j=l+1/2: -31.15015728
4p( 6) j=l-1/2: -26.73281564
4p( 6) j=l+1/2: -22.38230707
4d(10) j=l-1/2: -15.1647618
4d(10) j=l+1/2: -14.3484973
5s( 2) j=l+1/2: -5.225938506
4f(14) j=l-1/2: -4.960490099
4f(14) j=l+1/2: -4.775660273
5p( 6) j=l-1/2: -3.710458943
5p( 6) j=l+1/2: -2.889127431
5d(10) j=l-1/2: -0.8020049565
5d(10) j=l+1/2: -0.7070299184
6s( 2) j=l+1/2: -0.4209603386
6p( 2) j=l-1/2: -0.1549640727
```

# Iteration to self-consistency

The problem:

$$\mathbf{F}(\mathbf{x}) = \mathbf{x}$$

equivalently

$$\mathbf{R}(\mathbf{x}) = 0$$

for $\mathbf{R}(\mathbf{x}) = \mathbf{F}(\mathbf{x}) - \mathbf{x}$. We approximate

$$\mathbf{R}(\mathbf{x}_{M+1}) - \mathbf{R}(\mathbf{x}_M) \approx \mathbf{J} \cdot (\mathbf{x}_{M+1} - \mathbf{x}_M)$$

with the Jacobian

$$J_{ij} = \frac{\partial R_i}{\partial x_j}$$

We want $\mathbf{R}(\mathbf{x}_{M+1}) = 0$:

$$\mathbf{x}_{M+1} \approx \mathbf{x}_M - \mathbf{J}^{-1} \cdot \mathbf{R}(\mathbf{x}_M)$$

$\mathbf{J}$ is approximated by a sequence of $\mathbf{J}_0$, $\mathbf{J}_1$, $\mathbf{J}_2$, ...

# Linear mixing

$$\mathbf{x}_{M+1} \approx \mathbf{x}_M - \mathbf{J}_M^{-1} \cdot \mathbf{R}(\mathbf{x}_M)$$

with

$$\mathbf{J}_M^{-1} = -\alpha \mathbb{1}$$

so

$$\mathbf{x}_{M+1} = \mathbf{x}_M + \alpha \mathbf{R}(\mathbf{x}_M) = \mathbf{x}_M + \alpha(\mathbf{F}(\mathbf{x}_M) - \mathbf{x}_M)$$

**SciPy**

```
from scipy.optimize.nonlin import linearmixing
```

# "exciting" mixing

Used in the FP-LAPW DFT code (http://exciting.sourceforge.net/)

$$\mathbf{x}_{M+1} \approx \mathbf{x}_M - \mathbf{J}_M^{-1} \cdot \mathbf{R}(\mathbf{x}_M)$$

with

$$\mathbf{J}_M^{-1} = -\mathrm{diag}(\beta_1, \beta_2, \beta_3, \dots)$$

start with $\beta_1 = \beta_2 = \beta_3 = \cdots = \alpha$ and at every iteration adjust the parameters $\beta_i$ according to this very simple algorithm: if $R_i(\mathbf{x}_{M-1})R_i(\mathbf{x}_M) > 0$ then increase $\beta_i$ by $\alpha$ otherwise set $\beta_i = \alpha$ (if $\beta_i > \alpha_{max}$, set $\beta_i = \alpha_{max}$).

## SciPy

```
from scipy.optimize.nonlin import excitingmixing
```

# Broyden update

The *first Broyden method*:

$$\mathbf{J}_{M+1} = \mathbf{J}_M - \frac{(\Delta\mathbf{R}(\mathbf{x}_M) + \mathbf{J}_M \cdot \Delta\mathbf{x}_M)\Delta\mathbf{x}_M^T}{|\Delta\mathbf{x}_M|^2}$$

The *second Broyden method*:

$$\mathbf{J}_{M+1}^{-1} = \mathbf{J}_M^{-1} + \frac{(\Delta\mathbf{x}_M - \mathbf{J}_M^{-1} \cdot \Delta\mathbf{R}(\mathbf{x}_M))\Delta\mathbf{R}(\mathbf{x}_M)^T}{|\Delta\mathbf{R}(\mathbf{x}_M)|^2}$$

starting with the linear mixing:

$$\mathbf{J}_0^{-1} = -\alpha\mathbb{1}$$

**SciPy**

```
from scipy.optimize import broyden1, broyden2
```

# low memory second Broyden update

The *second Broyden method*
$(\mathbf{J}_{M+1}^{-1} = \mathbf{J}_M^{-1} + \frac{(\Delta\mathbf{x}_M - \mathbf{J}_M^{-1}\cdot\Delta\mathbf{R}(\mathbf{x}_M))\Delta\mathbf{R}(\mathbf{x}_M)^T}{|\Delta\mathbf{R}(\mathbf{x}_M)|^2})$ can be written as

$$\mathbf{J}_{M+1}^{-1} = \mathbf{J}_M^{-1} + \mathbf{u}\mathbf{v}^T$$

with

$$\mathbf{u} = \Delta\mathbf{x}_M - \mathbf{J}_M^{-1}\cdot\Delta\mathbf{R}(\mathbf{x}_M)$$

$$\mathbf{v} = \frac{\Delta\mathbf{R}(\mathbf{x}_M)}{|\Delta\mathbf{R}(\mathbf{x}_M)|^2}$$

so the whole inverse Jacobian can be written as

$$\mathbf{J}_M^{-1} = -\alpha\mathbb{1} + \mathbf{u}_1\mathbf{v}_1^T + \mathbf{u}_2\mathbf{v}_2^T + \mathbf{u}_3\mathbf{v}_3^T + \cdots$$

$$\mathbf{J}_M^{-1}\cdot\mathbf{y} = -\alpha\mathbf{y} + \mathbf{u}_1(\mathbf{v}_1^T\mathbf{y}) + \mathbf{u}_2(\mathbf{v}_2^T\mathbf{y}) + \mathbf{u}_3(\mathbf{v}_3^T\mathbf{y}) + \cdots$$

**SciPy**

```
from scipy.optimize import broyden3
```

# Other methods

The *generalized Broyden method* (modified Broyden method):

$$\sum_{p=M-k}^{M-1} (1+\omega_0^2\delta_{pn})\Delta\mathbf{R}(\mathbf{x}_n)^T\Delta\mathbf{R}(\mathbf{x}_p)\gamma_p = \Delta\mathbf{R}(\mathbf{x}_n)^T\mathbf{R}(\mathbf{x}_M)$$

$$\mathbf{x}_{M+1} = \mathbf{x}_M + \beta_M\mathbf{R}(\mathbf{x}_M) - \sum_{p=M-k}^{M-1} \gamma_p(\Delta\mathbf{x}_p + \beta_M\Delta\mathbf{R}(\mathbf{x}_p))$$

other methods: Anderson, extended Anderson

**SciPy**

```
from scipy.optimize import broyden_generalized,
    anderson, anderson2
```

# Finite element formulation

One particle Schrödinger equation:

$$\left(-\frac{\hbar^2}{2m}\nabla^2 + V\right)\psi = E\psi\,.$$

FEM:

$$(K_{ij} + V_{ij})\,q_j = EM_{ij}q_j + F_i\,,$$

$$\begin{aligned} V_{ij} &= \int \phi_i V \phi_j \,\mathrm{d}V\,, \\ M_{ij} &= \int \phi_i \phi_j \,\mathrm{d}V\,, \\ K_{ij} &= \frac{\hbar^2}{2m}\int \nabla\phi_i \cdot \nabla\phi_j \,\mathrm{d}V\,, \\ F_i &= \frac{\hbar^2}{2m}\oint \frac{\mathrm{d}\psi}{\mathrm{d}n}\phi_i \,\mathrm{d}S\,. \end{aligned}$$

Usually we set $F_i = 0$.

# SfePy

- SfePy = general finite element analysis software
- BSD open-source license
- available at
  - `http://sfepy.org` (developers)
    - mailing lists, issue (bug) tracking
    - we encourage and support everyone who joins!
  - `http://sfepy.kme.zcu.cz` (project information)
- selected applications:
  - *homogenization of porous media* (parallel flows in a deformable porous medium)
  - *acoustic band gaps* (homogenization of a strongly heterogenous elastic structure: phononic materials)
  - *shape optimization* in incompressible flow problems

# Particle in the box

$$V(x) = \begin{cases} 0, & \text{inside the box} \quad a \times a \times a \\ \infty, & \text{outside} \end{cases}$$

Analytic solution:

$$E_{n_1 n_2 n_3} = \frac{\pi^2}{2a^2}\left(n_1^2 + n_2^2 + n_3^2\right)$$

where $n_i = 1, 2, 3, \ldots$ are independent quantum numbers. We chose $a = 1$, i.e.: $E_{111} = 14.804$, $E_{211} = E_{121} = E_{112} = 29.608$, $E_{122} = E_{212} = E_{221} = 44.413$, $E_{311} = E_{131} = E_{113} = 54.282$ $E_{222} = 59.217$, $E_{123} = E_{\text{perm.}} = 69.087$.

Numerical solution ($a = 1$, 24702 nodes):

| E | 1 | 2-4 | 5-7 | 8-10 | 11 | 12- |
|---|---|---|---|---|---|---|
| theory | 14.804 | 29.608 | 44.413 | 54.282 | 59.217 | 69.087 |
| FEM | 14.861 | 29.833 | 44.919 | 55.035 | 60.123 | 70.305 |
| | | 29.834 | 44.920 | 55.042 | | 70.310 |
| | | 29.836 | 44.925 | 55.047 | | ... |

# 3D Harmonic oscillator

$$V(r) = \begin{cases} \frac{1}{2}\omega^2 r^2, & \text{inside the box} \quad a \times a \times a \\ \infty, & \text{outside} \end{cases}$$

Analytic solution in the limit $a \to \infty$:

$$E_{nl} = \left(2n + l + \frac{3}{2}\right)\omega$$

where $n, l = 0, 1, 2, \ldots$. Degeneracy is $2l + 1$, so: $E_{00} = \frac{3}{2}$, triple $E_{01} = \frac{5}{2}$, $E_{10} = \frac{7}{2}$, quintuple $E_{02} = \frac{7}{2}$ triple $E_{11} = \frac{9}{2}$, quintuple $E_{12} = \frac{11}{2}$:

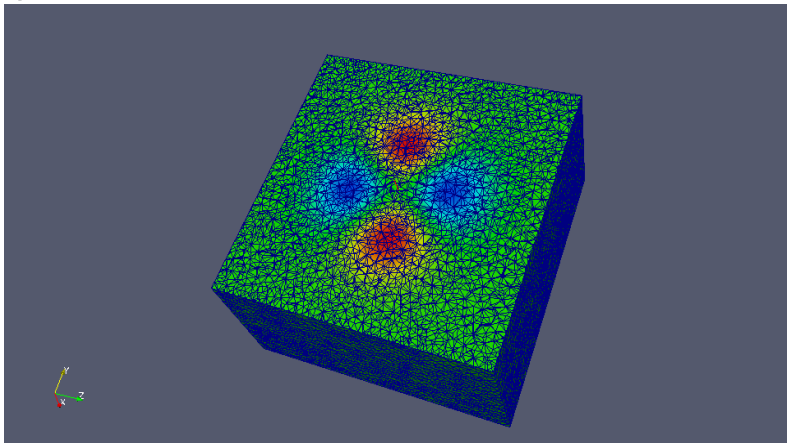
Numerical solution ($a = 15$, $\omega = 1$, 290620 nodes):

| E | 1 | 2-4 | 5-10 | 11- |
|---|---|---|---|---|
| theory | 1.5 | 2.5 | 3.5 | 4.5 |
| FEM | 1.522 | 2.535 | 3.554 | 4.578 |
| | | 2.536 | 3.555 | 4.579 |
| | | 2.536 | 3.555 | 4.579 |
| | | | 3.555 | ... |
| | | | 3.556 | |
| | | | 3.556 | |

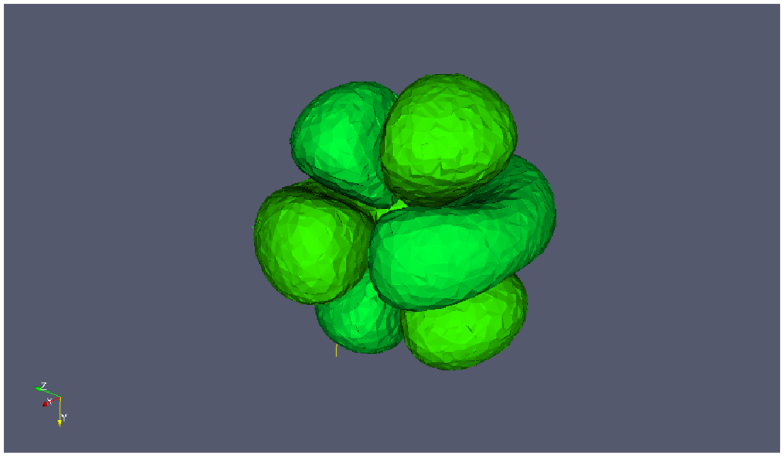
# 3D Harmonic oscillator

Eigenvectors:

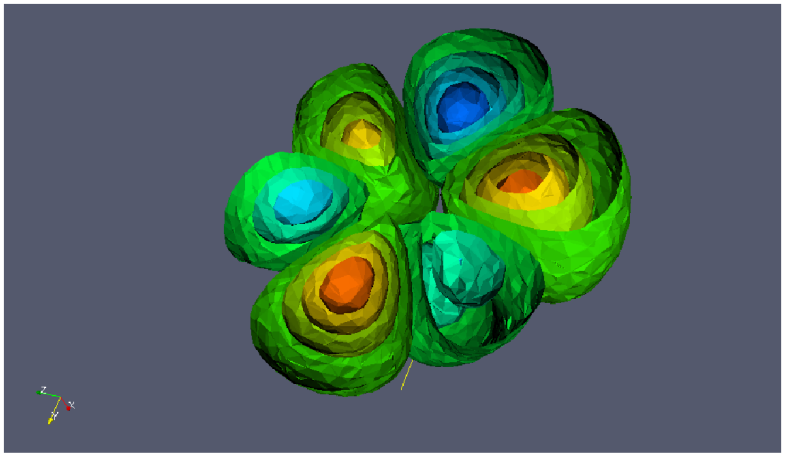
0th



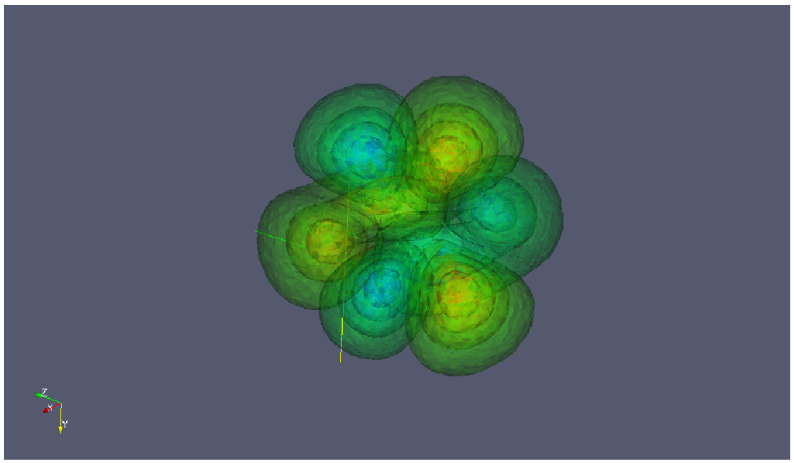
12th



10th



12th



# Hydrogen atom

$$V(r) = \begin{cases} -\frac{1}{r}, & \text{inside the box} \quad a \times a \times a \\ \infty, & \text{outside} \end{cases}$$

Analytic solution in the limit $a \to \infty$:

$$E_n = -\frac{1}{2n^2}$$

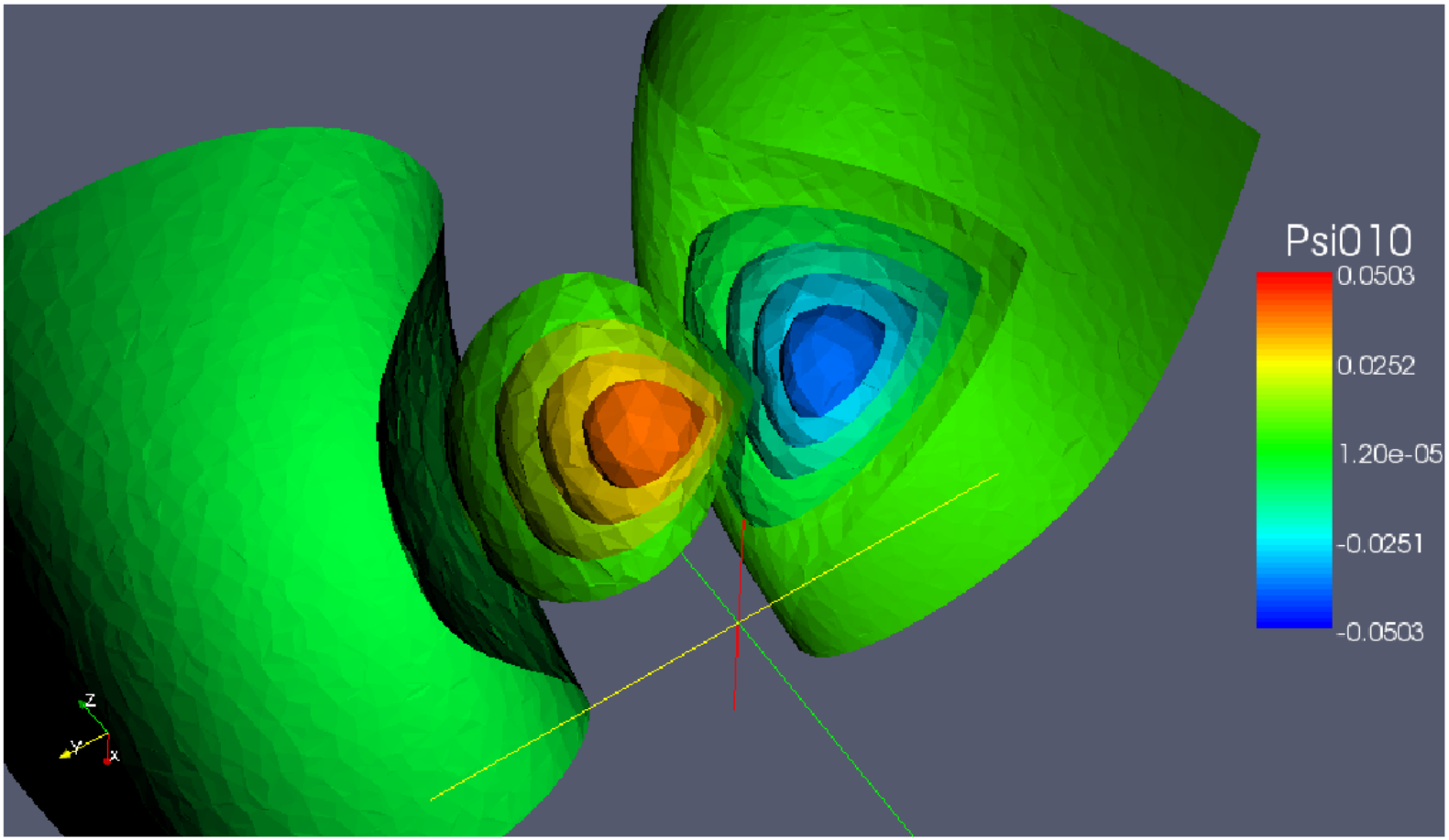
where $n = 1, 2, 3, \ldots$. Degeneracy is $n^2$, so: $E_1 = -\frac{1}{2} = -0.5$, $E_2 = -\frac{1}{8} = -0.125$, $E_3 = -\frac{1}{18} = -0.055$, $E_4 = -\frac{1}{32} = -0.031$.

Numerical solution ($a = 15$, 160000 nodes):

| E | 1 | 2-5 | 6-14 | 15- |
|---|---|---|---|---|
| theory | -0.5 | -0.125 | -0.055 | -0.031 |
| FEM | -0.481 | -0.118 | -0.006 | $\cdots$ |

# Hydrogen atom

11th eigenvalue (calculated: -0.04398532, exact: -0.056), on the mesh with 976 691 tetrahedrons and 163 666 nodes, for the hydrogen atom (V=-1/r).



We solve the Kohn-Sham equations using FEM:

$$(-\tfrac{1}{2}\nabla^2 + V_H(\mathbf{r}) + V_{xc}(\mathbf{r}) + v(\mathbf{r}))\psi_i(\mathbf{r}) = \epsilon_i\psi(\mathbf{r})$$

that yield the orbitals $\psi_i$ that reproduce the density $n(\mathbf{r})$ of the original interacting system

$$n(\mathbf{r}) = \sum_i^N |\psi_i(\mathbf{r})|^2$$

$$\nabla^2 V_H = n(\mathbf{r})$$

$$v(\mathbf{r}) = \sum_k -\frac{Z_k}{|\mathbf{r} - \mathbf{R_k}|}$$

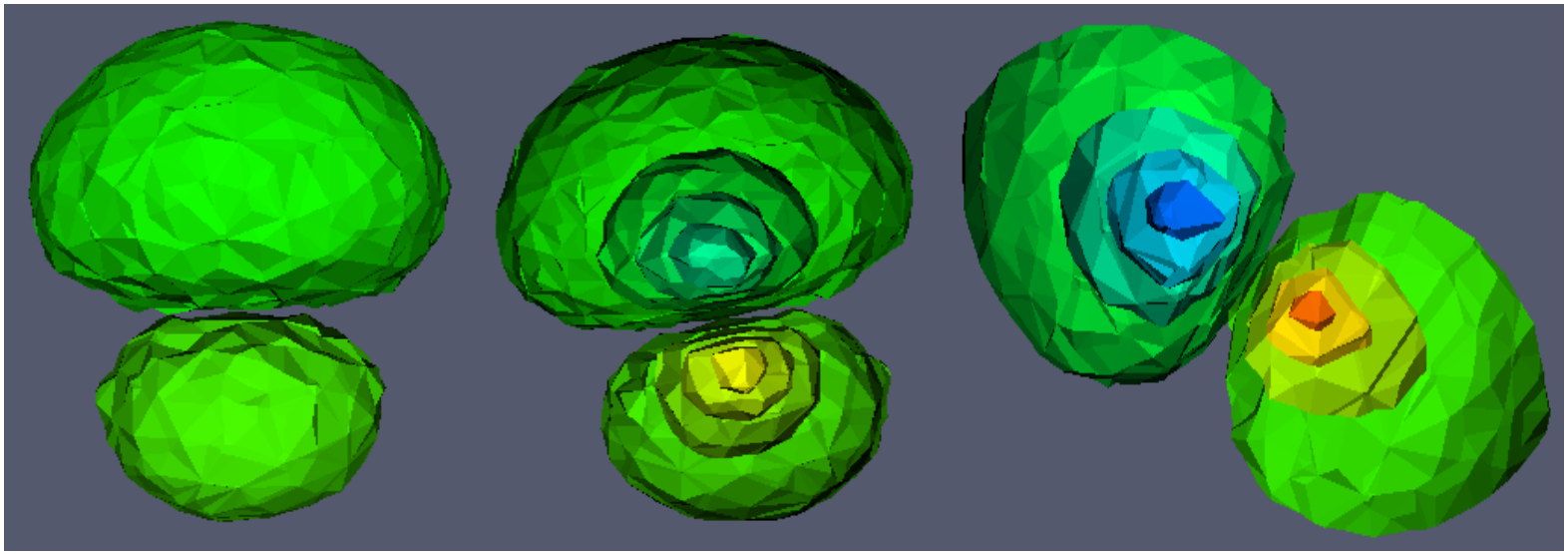
# Boron

Current status (uniform tetrahedral mesh, 50 000 nodes):

| | | | | | |
|---|---|---|---|---|---|
| radial | -6.564449519 | -6.564449519 | -0.3447644413 | -0.3447644413 | -0.1366622746 |
| FEM | -3.18675417 | -0.68091886 | -0.65252624 | -0.63762163 | -0.58488204 |

- Bad convergence should greatly improve with a better mesh

2th eigenvector (contours and a slice), 3th eigenvector (slice)



# Conclusion, Future work

- automatic mesh generation and refining
- use pseudopotentials (reduces the number of electrons to solve for)
- only depend on open source (free software) solutions


## Acknowledgements

This research was partly supported by the LC06040 research center project and the GACR grant no. IAA100100637.