

Adaptive Multi-Mesh hp-FEM in Electronic Structure Calculations

Ondřej Čertík^{1,2}, Pavel Šolín¹, Jiří Vackář²

¹ University of Nevada, Reno, USA

² Institute of Physics, Academy of Sciences of the Czech Republic

March 6, 2009

Introduction and overview:

- Electronic Structure and Density Functional Theory (DFT)
- Spherically symmetric problems, examples
- Mixing schemes

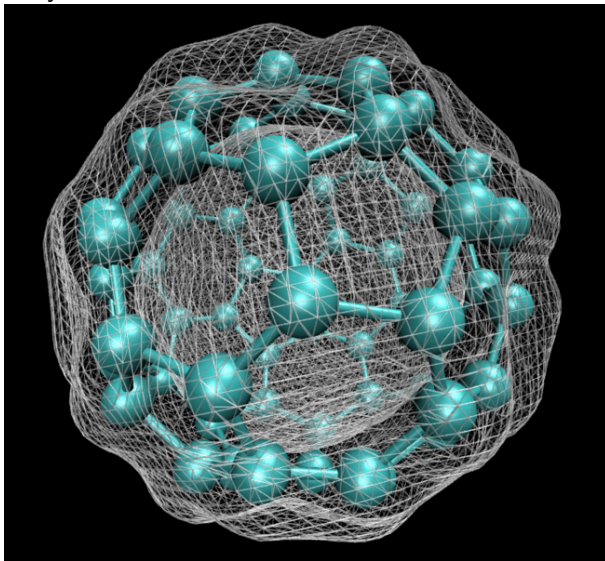
Main part:

- Non-symmetric 2D and 3D problems:
 - low order Finite Element Method (FEM)
 - high order adaptive FEM

Conclusion

Why Electronic Structure

Why Electronic Structure?



(source: Wikipedia)



Schrödinger equation

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

where

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

$$\hat{U} = \sum_{i<j} U(\mathbf{r}_i, \mathbf{r}_j) = \frac{1}{2} \sum_{i,j} U(\mathbf{r}_i, \mathbf{r}_j)$$

$$U(\mathbf{r}_i, \mathbf{r}_j) = U(\mathbf{r}_j, \mathbf{r}_i) = \frac{1}{|\mathbf{r}_i - \mathbf{r}_j|}$$

$$\hat{V} = \sum_i^N v(\mathbf{r}_i)$$

$$v(\mathbf{r}_i) = \sum_k -\frac{Z_k}{|\mathbf{r}_i - \mathbf{R}_k|}$$

Density Functional Theory

We solve the Kohn-Sham equations:

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

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

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

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

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

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

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

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:

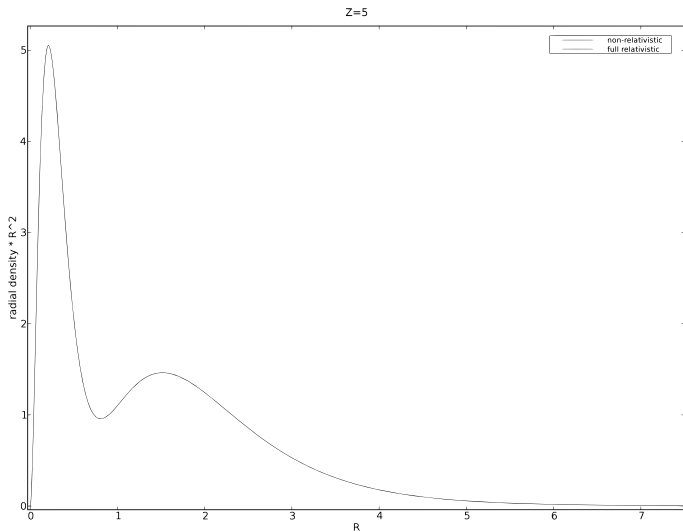
$$g_\kappa'' + \left(\frac{2}{r} + \frac{V'}{2Mc^2} \right) g_\kappa' + \left[(E - V) - \frac{\kappa(\kappa + 1)}{2Mr^2} + \frac{\kappa + 1}{4M^2c^2r} V' \right] 2Mg_\kappa = 0$$

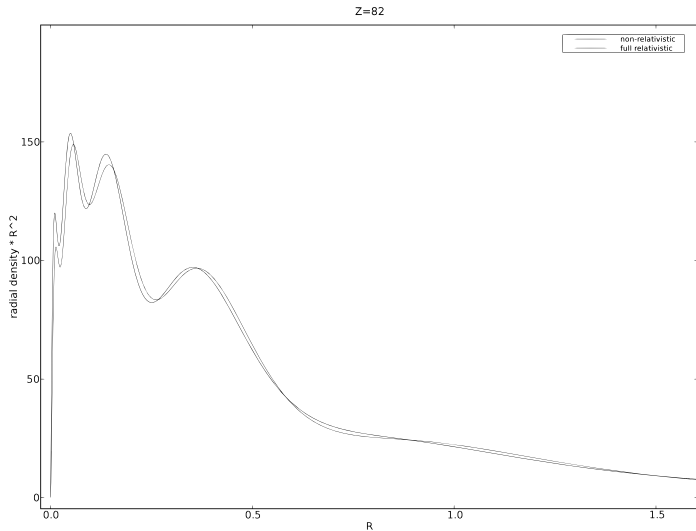
$$f_\kappa = \frac{g_\kappa'}{2Mc} + \frac{\kappa + 1}{r} \frac{g_\kappa}{2Mc}$$

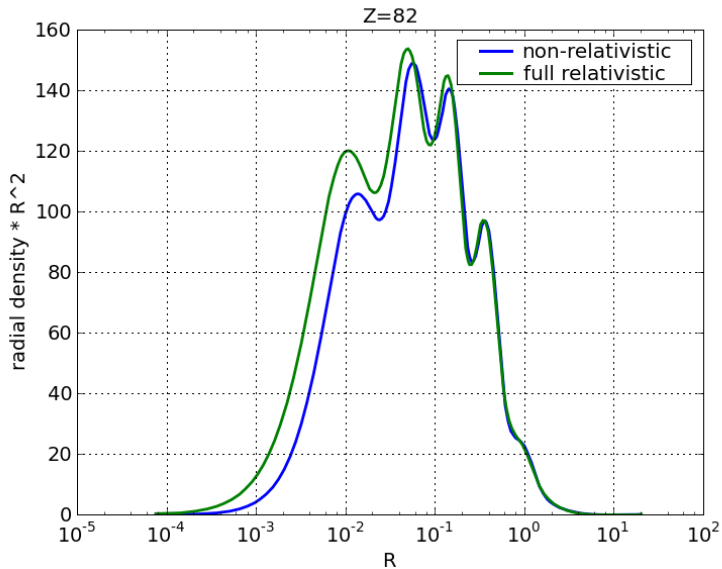
$$R^2 = f^2 + g^2$$

Radial Schrödinger equation:

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







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

4s(2) j=1+1/2: -31.15015728
4p(6) j=1-1/2: -26.73281564
4p(6) j=1+1/2: -22.38230707
4d(10) j=1-1/2: -15.1647618
4d(10) j=1+1/2: -14.3484973
5s(2) j=1+1/2: -5.225938506
4f(14) j=1-1/2: -4.960490099
4f(14) j=1+1/2: -4.775660273
5p(6) j=1-1/2: -3.710458943
5p(6) j=1+1/2: -2.889127431
5d(10) j=1-1/2: -0.8020049565
5d(10) j=1+1/2: -0.7070299184
6s(2) j=1+1/2: -0.4209603386
6p(2) j=1-1/2: -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 $\mathbf{J}_0, \mathbf{J}_1, \mathbf{J}_2, \dots$

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

with

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

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

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

start with $\beta_1 = \beta_2 = \beta_3 = \dots = \alpha$ and at every iteration adjust the parameters β_i according to this very simple algorithm: if $R_i(\mathbf{x}_{M-1})R_i(\mathbf{x}_M) > 0$ then increase β_i by α 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:

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

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

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

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

SciPy

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

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
```

One particle Schrödinger equation:

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

Truncation boundary conditions (zero Dirichlet far away).

Weak formulation

$$\int_{\Omega} \frac{\hbar^2}{2m} \nabla \psi \cdot \nabla v + V \psi v \, dV = \int_{\Omega} E \psi v \, dV + \underbrace{\oint_{\Gamma} \frac{\hbar^2}{2m} (\nabla \psi) v \cdot \mathbf{n} \, dS}_0 .$$

Discrete generalized eigenvalue problem

$$(K_{ij} + V_{ij}) q_j = E M_{ij} q_j .$$

Finite element formulation

One particle Schrödinger equation:

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

FEM:

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

$$V_{ij} = \int \phi_i V \phi_j dV ,$$

$$M_{ij} = \int \phi_i \phi_j dV ,$$

$$K_{ij} = \frac{\hbar^2}{2m} \int \nabla \phi_i \cdot \nabla \phi_j dV ,$$

$$F_i = \frac{\hbar^2}{2m} \oint \frac{d\psi}{dn} \phi_i dS .$$

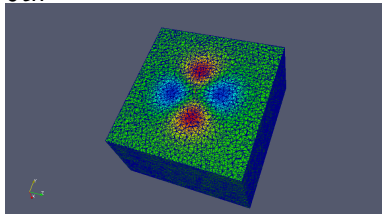
Usually we set $F_i = 0$.

- SfePy = low order FEM software (non adaptive)
- BSD open-source license
- Many applications in engineering and science
- available at <http://sfepy.org>

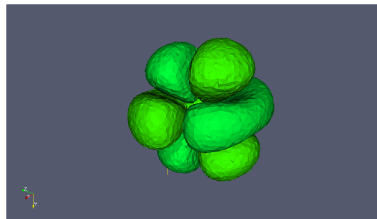
3D Harmonic oscillator

Eigenvectors:

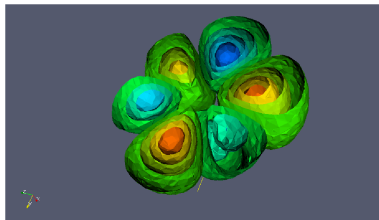
0th



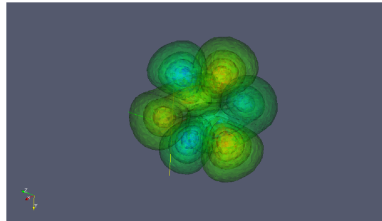
12th



10th



12th



Hydrogen atom

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

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

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

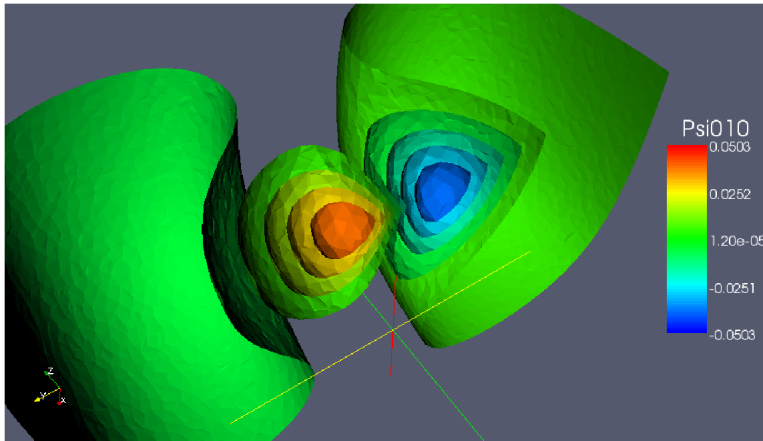
where $n = 1, 2, 3, \dots$. Degeneracy is n^2 , so: $E_1 = -\frac{1}{2} = -0.5$,
 $E_2 = -\frac{1}{8} = -0.125$, $E_3 = -\frac{1}{18} = -0.055$, $E_4 = -\frac{1}{32} = -0.031$.

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	...

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:

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

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

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

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

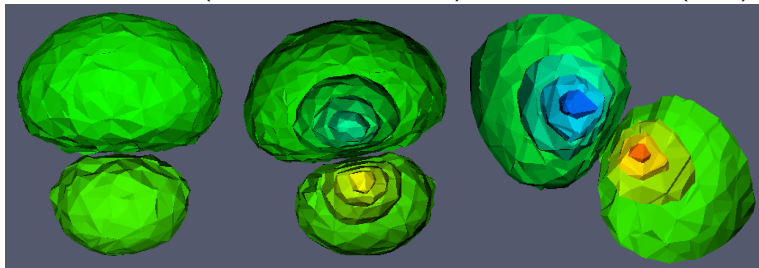
$$v(\mathbf{r}) = \sum_k \frac{Z_k}{|\mathbf{r} - \mathbf{R}_k|}$$

Boron (5 electrons)

Uniform tetrahedral mesh, 50 000 nodes, 5 lowest eigenvalues

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)



Second attempt: Hermes

- Hermes = higher order adaptive FEM (hp-FEM) software
- GPL open-source license
- Many applications in engineering and science
- Developed at University of Nevada, Reno and Institute of Thermomechanics, Prague
- available at <http://hpfem.math.unr.edu>

Uniform vs Adapted Mesh

2D hydrogen atom, square box, 6 times uniformly refined:

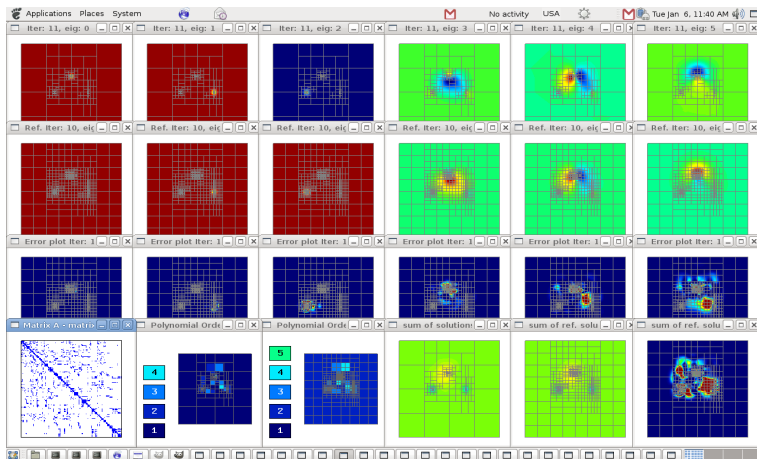
- matrix: 16129×16129 , error: 12.6%

adaptive hp-FEM refinement:

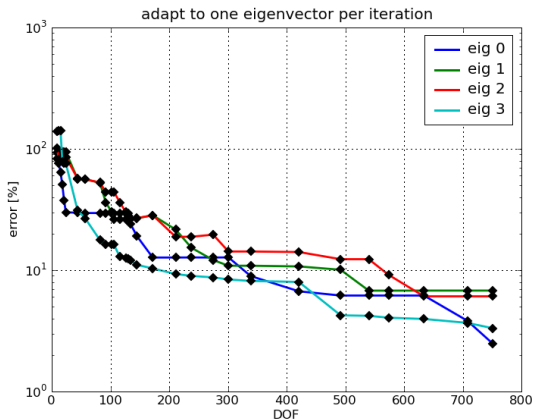
- matrix: 141×141 , error: 7.5%

How to converge the eigenvectors?

hp-FEM on a single mesh

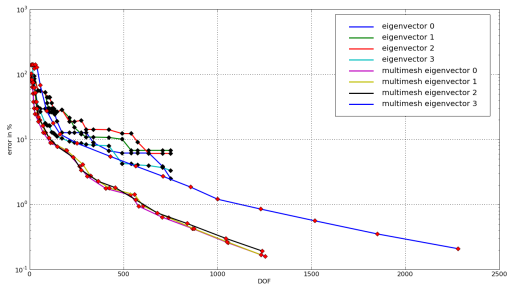


Converging one eigenvector per iteration:

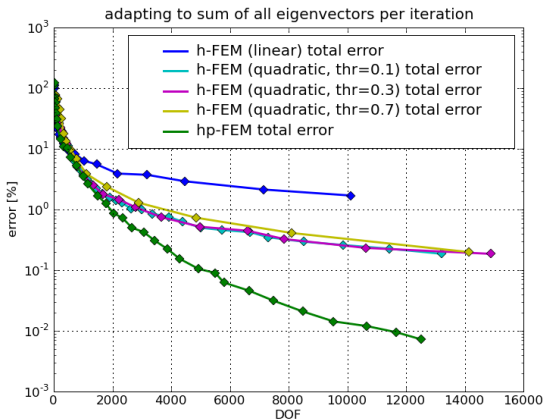


Multimesh Comparison

eigenvector 0..4: converging one eigenvector at a time
multimesh eigenvector 0..4: converging each eigenvector on its own mesh

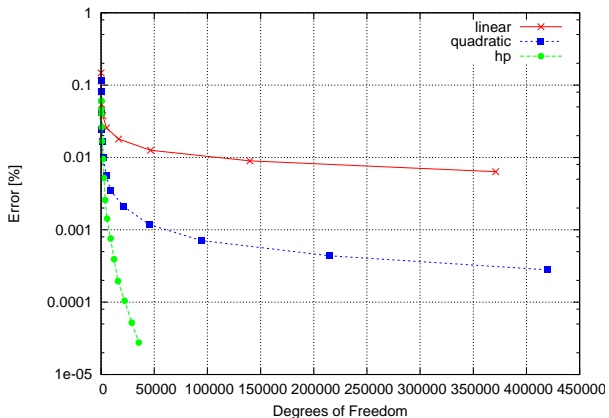


Adapting to the Sum of Eigenvectors



hp-FEM convergence in 3D

Laplace equation (note: on the y-scale, 1 means 100%):



- hp-FEM treatment is necessary
- implement the DFT self-consistency cycle in 2D, then go to 3D
- 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.