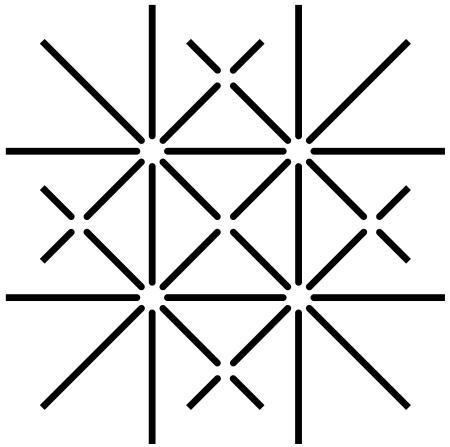
The version of the BigDFT package for (partially)periodic systems.

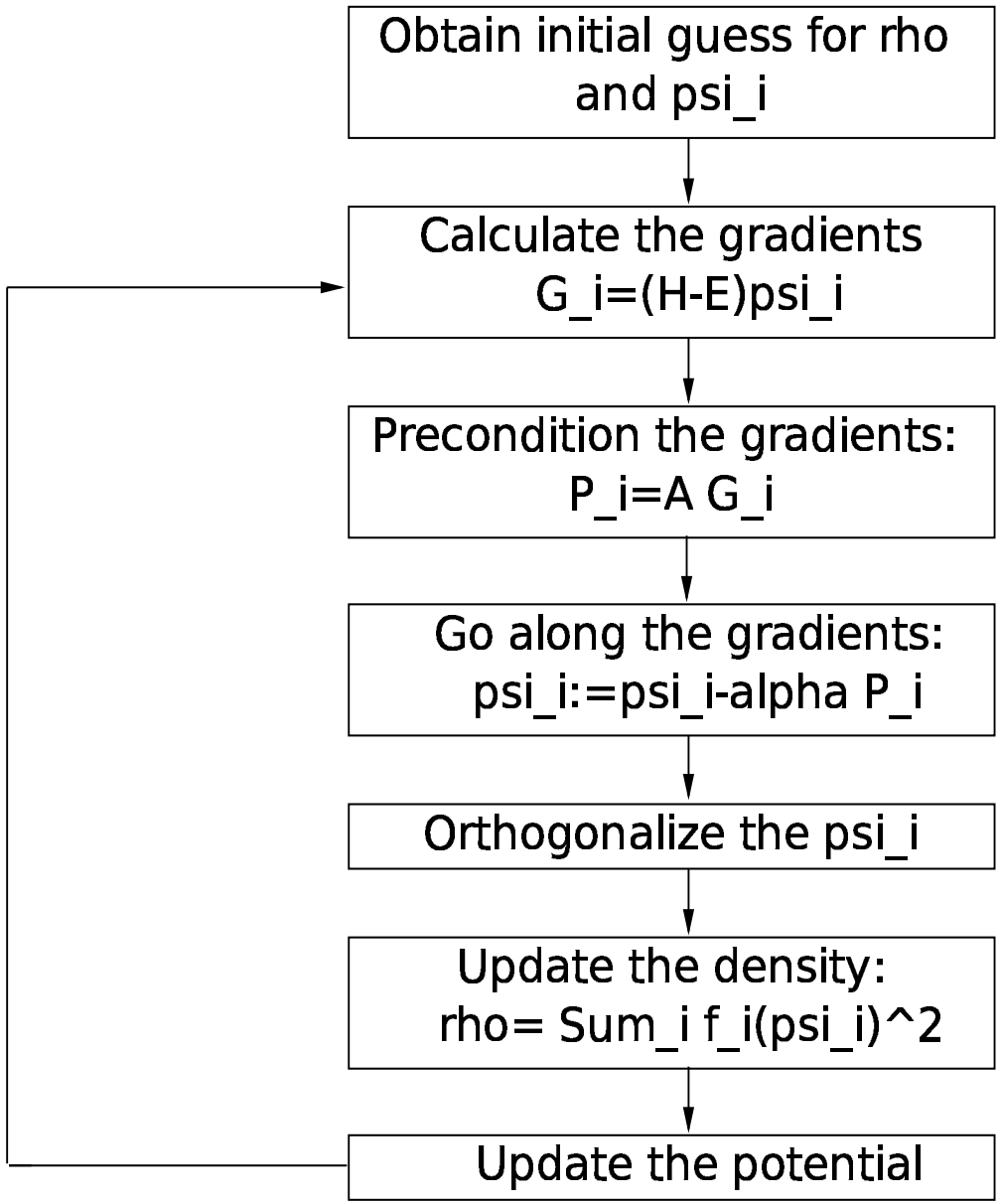
Alexey Neelov
Alexey.Neelov@unibas.ch



UNI
BASEL

The plan of the talk:

- The 3d wavelet basis
- The data structures for wavelet storage
- Periodic systems
- Code optimization (loop unrolling)
- Preconditioning
- Performance



The BigDFT self-consitency cycle

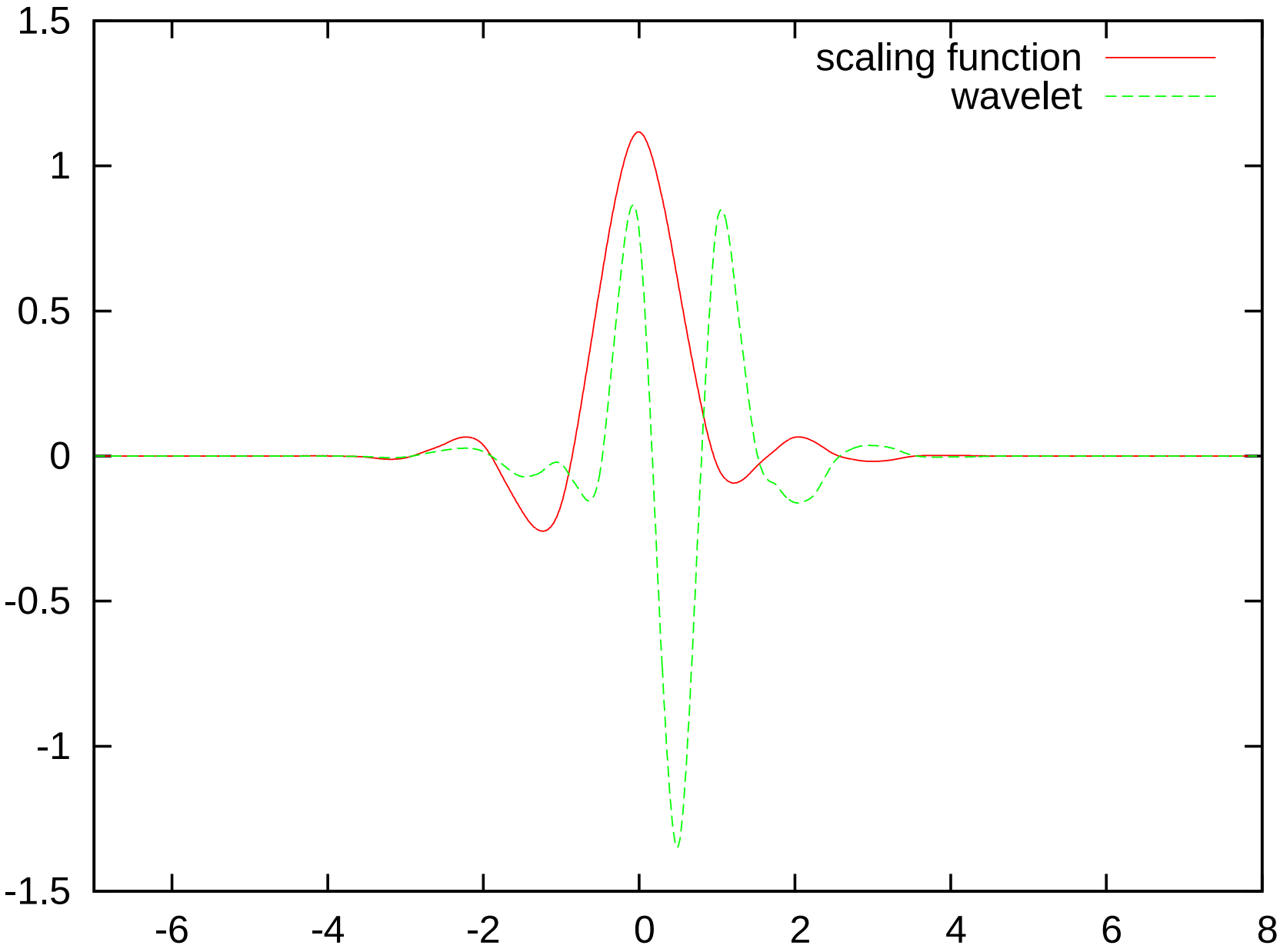


Figure 1: Daubechies-16 scaling function and wavelet

Wavelet basis sets in three dimensions

1 scaling function 7 wavelets

all are products of 1-dim scaling functions and wavelets:

$$
\begin{aligned}
\phi_{i,j,k}(x,y,z) &= \phi(x-i)\phi(y-j)\phi(z-k) \\
\psi^1_{i,j,k}(x,y,z) &= \phi(x-i)\phi(y-j)\psi(z-k) \\
\psi^2_{i,j,k}(x,y,z) &= \phi(x-i)\psi(y-j)\phi(z-k) \\
\psi^3_{i,j,k}(x,y,z) &= \phi(x-i)\psi(y-j)\psi(z-k) \\
\psi^4_{i,j,k}(x,y,z) &= \psi(x-i)\phi(y-j)\phi(z-k) \\
\psi^5_{i,j,k}(x,y,z) &= \psi(x-i)\phi(y-j)\psi(z-k) \\
\psi^6_{i,j,k}(x,y,z) &= \psi(x-i)\psi(y-j)\phi(z-k) \\
\psi^7_{i,j,k}(x,y,z) &= \psi(x-i)\psi(y-j)\psi(z-k)
\end{aligned}
$$

- Our basis contains only elements with values of indices i_1, i_2, i_3 inside the localization region.
- The localization region is not rectangular.

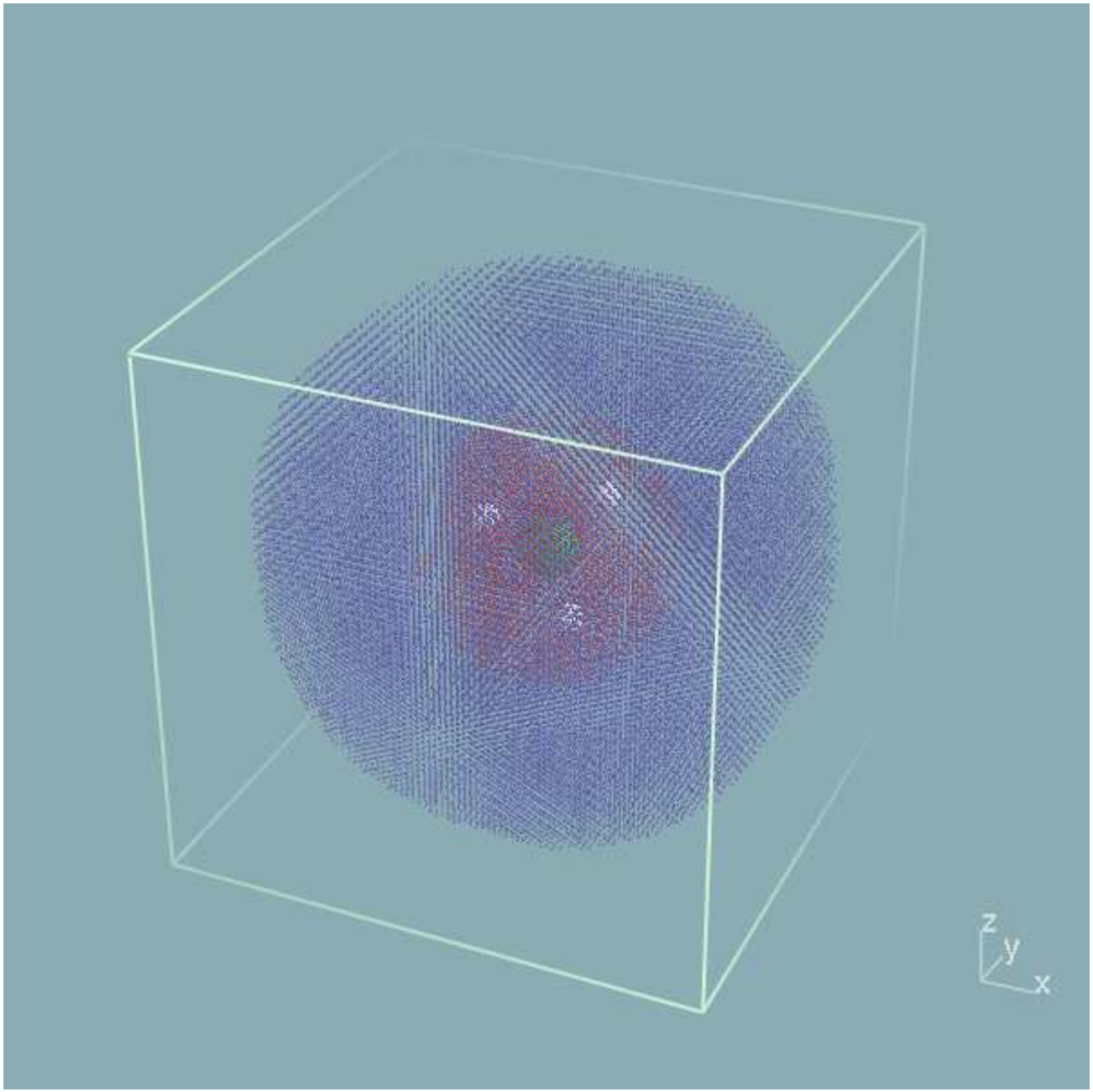


Figure 2: SiH4 molecule, free boundary conditions

Therefore, *two* forms of storage of *the same* wavefunction:

- Compressed form: only the expansion coefficients inside the localization region are stored. Used for the storage of Kohn-Sham orbitals in general and for their orthogonalization in particular.
- Uncompressed form: all the expansion coefficients in the cell are stored. Used for convolutions and solution of Poisson equation.

More details: the scheme on the blackboard.

Periodic systems.

- Fully periodic systems (e.g. crystals).
- "Periodized" clusters: speed comparable with the free BC.
- Impurities and defects: ?
- Partially periodic: surfaces and slabs (in progress)

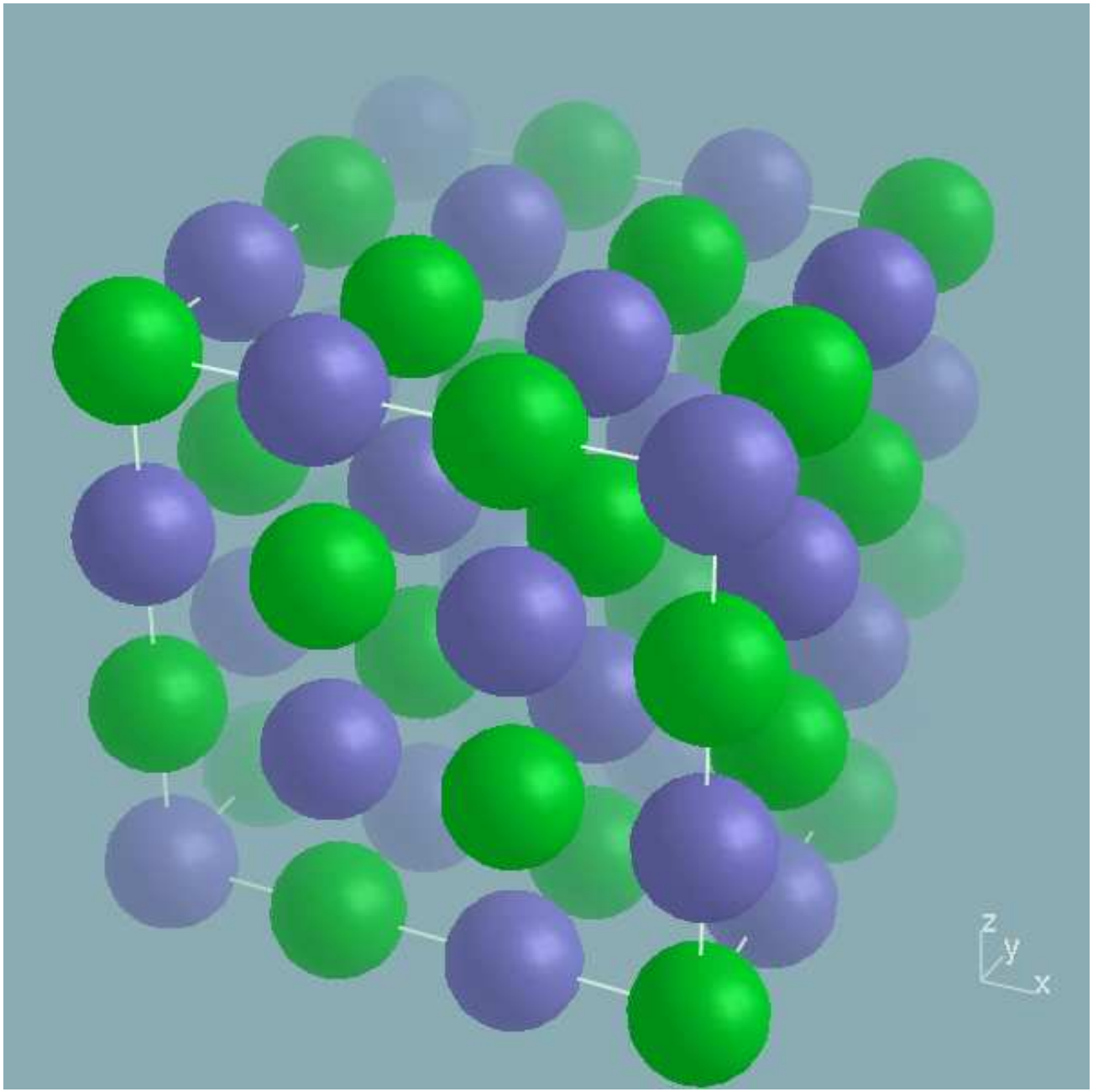


Figure 3: Rock salt cristal, 64 atoms

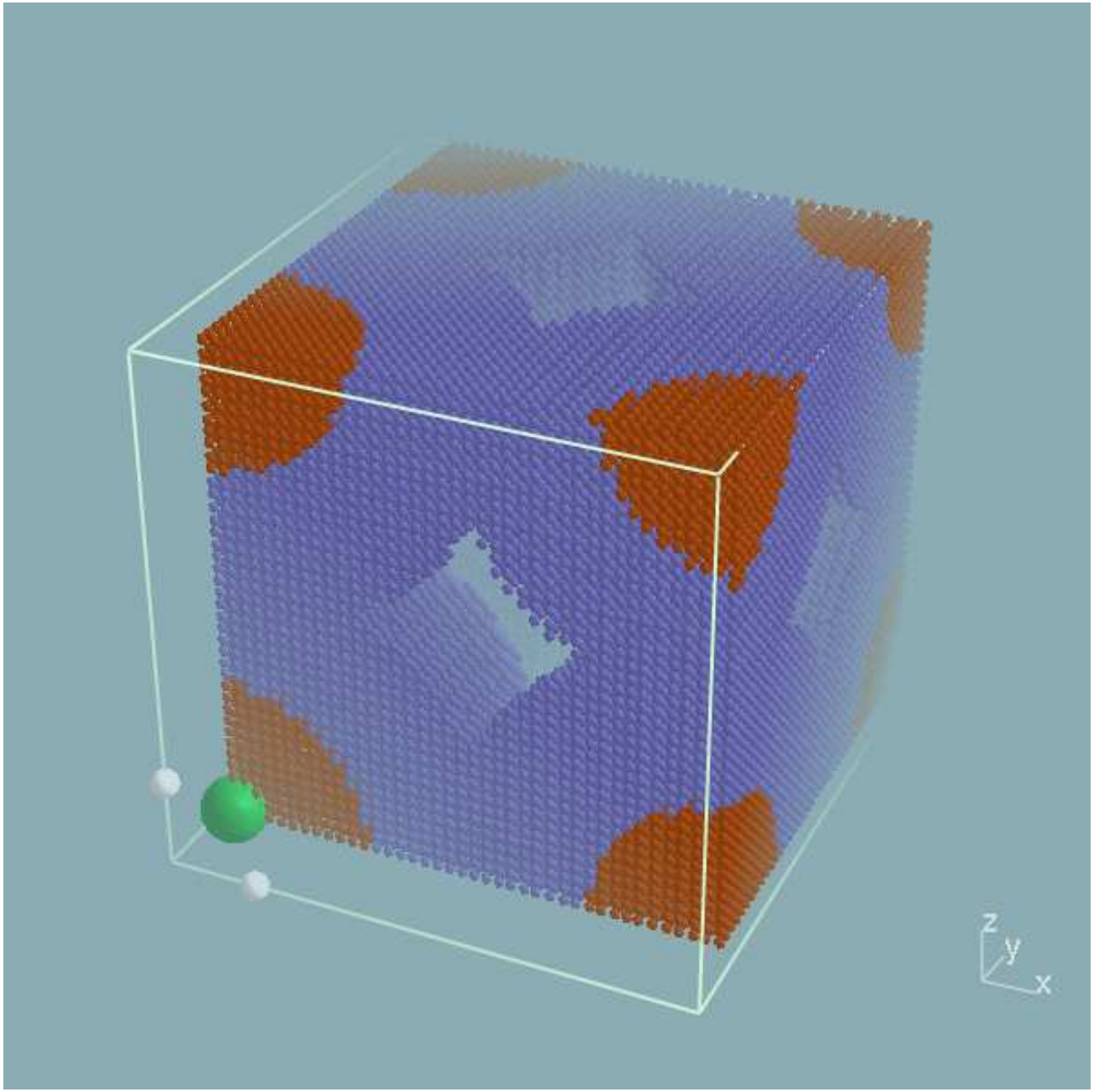


Figure 4: SiH4 molecule, periodic case

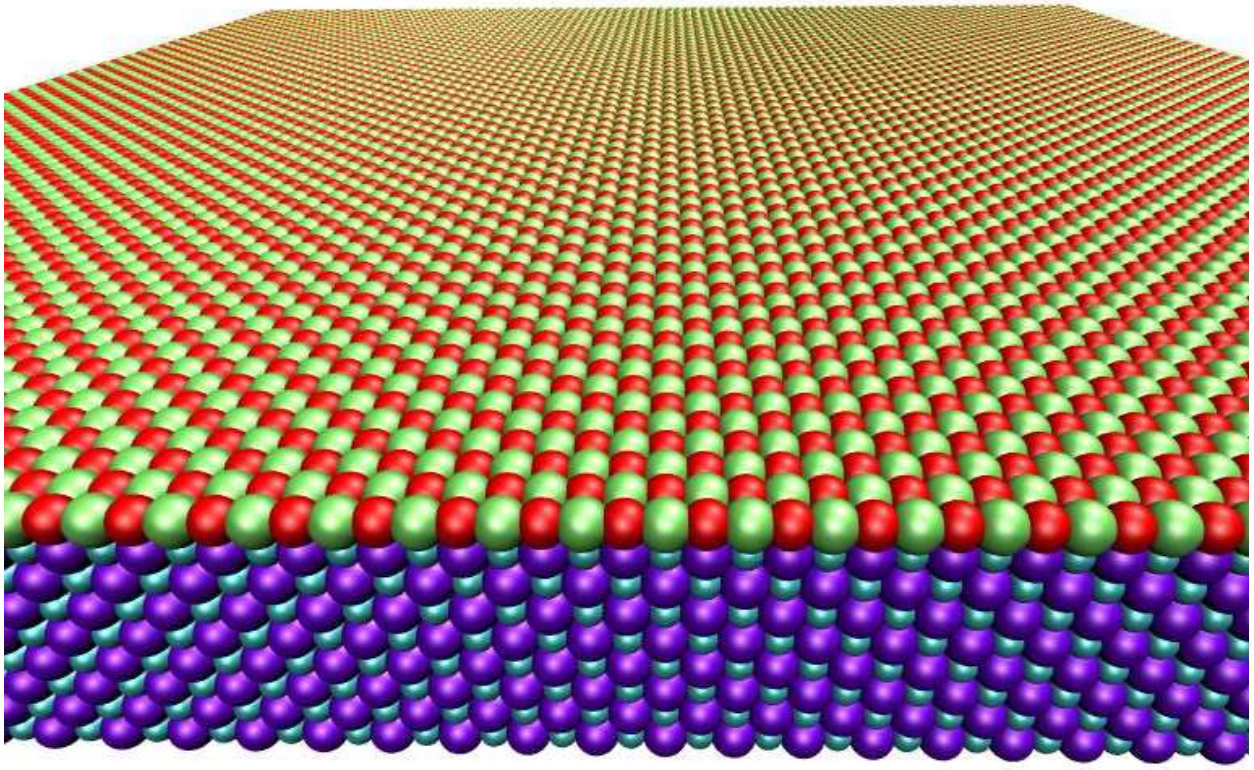


Figure 5: An example of a slab-like system (a NaCl monolayer on KBr)

What changes in BigDFT if one has a periodic system instead of a free one?

- The Poisson Solver changes (the talk of L. Genovese)
- The implementation of the nonlocal pseudopotential is slightly changed
- The convolutions are periodic, which means in particular:

- Rectangular localization regions are more suitable for convolutions
- The arrays do not grow or shrink after a convolution
- Fourier-based preconditioning is more efficient
- Calculation of tail is not needed

Periodic convolutions

Convolutions in the BigDFT, periodic version:

- Wavelet transform: wavelets and scaling functions $\Leftrightarrow$ fine scaling functions
- Application of the Laplacian (the kinetic energy term)
- Magic filter application: fine scaling functions $\Leftrightarrow$ real space grid

Consider the example of the Laplacian in detail (only the $\partial^2/\partial x^2$ part, for brevity)

In the simplest case, use the `modulo` function:

```
do i3=0,n3
   do i2=0,n2
      do i1=0,n1
         tt=0.d0
         do l=lowfil,lupfil
            j=modulo(i1+l,n1+1)
            tt=tt+x(j   ,i2,i3)*fil(l,1)
         enddo
         y(i1,i2,i3)=tt
      enddo
   enddo
enddo
```

In the code, we unroll such loop by the factors of 8 or 12. For demonstration: unrolling by four. We divide the array onto blocks with “transverse” dimension four. The aim is to get four independent streams.

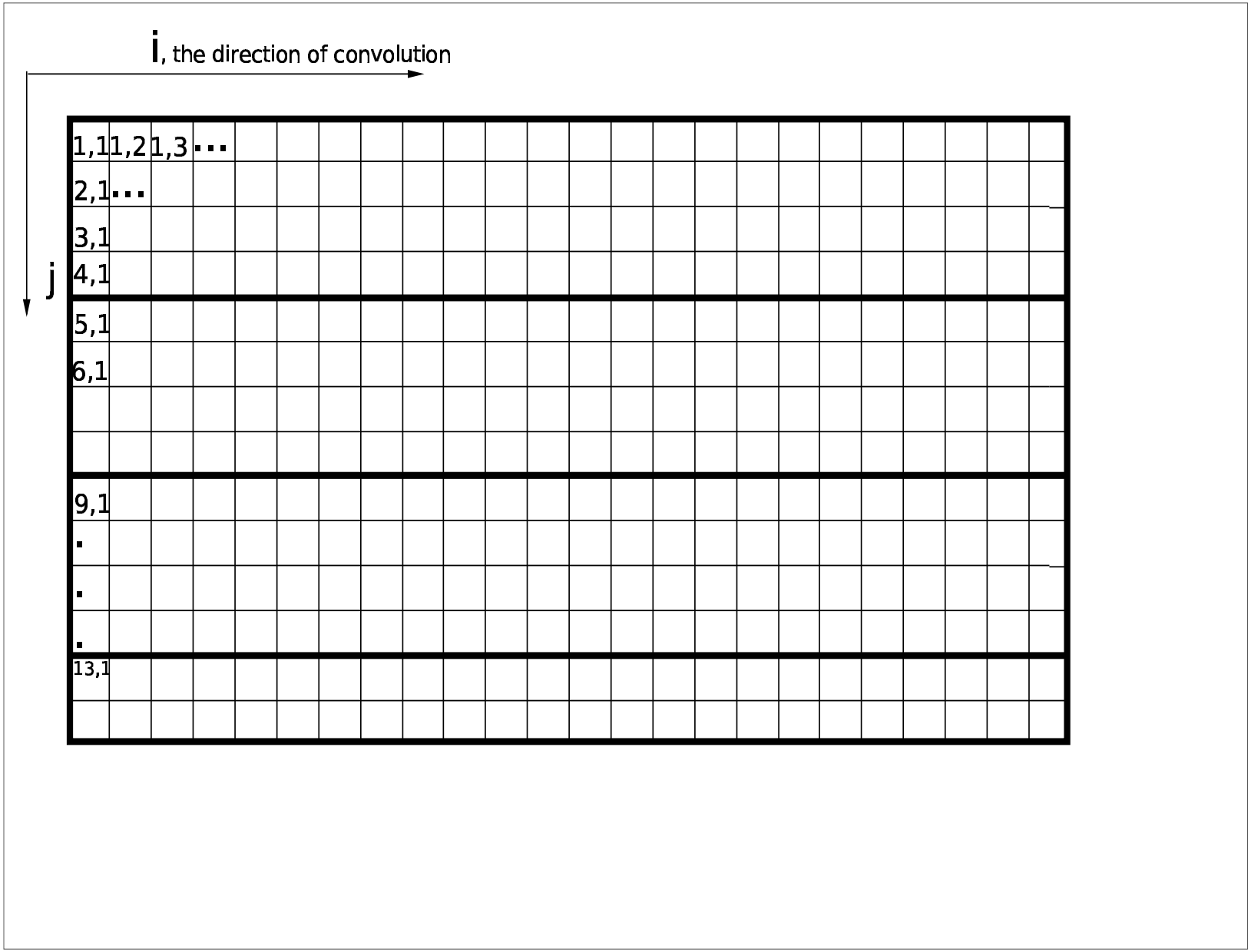


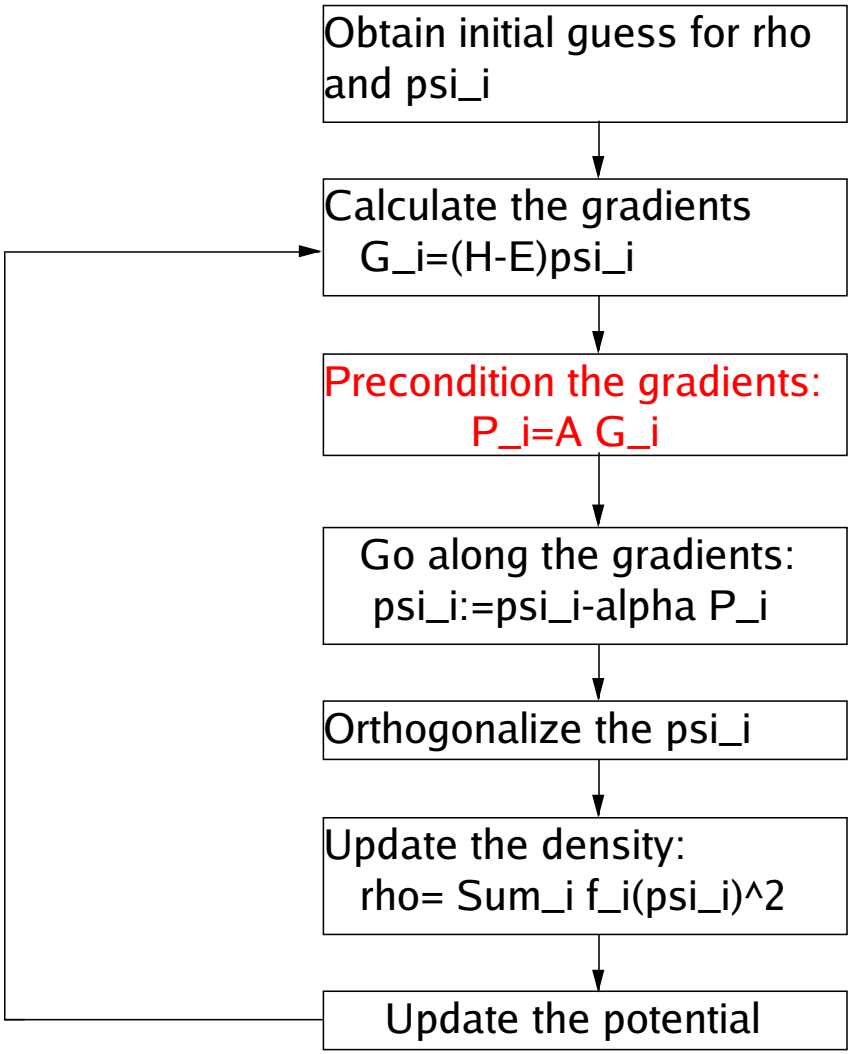
Figure 6: Loop unrolling for the array y(j,i)

```
do i=0,ndat/4-1
    do i1=0,n1
       tt1 =0.d0
       tt2 =0.d0
       tt3 =0.d0
       tt4 =0.d0
       do l=lowfil,lupfil
          j=mod_arr1(i1+l)
          tt1=tt1+x(j,i*4+1)*fil(l,1)
          tt2=tt2+x(j,i*4+2)*fil(l,1)
          tt3=tt3+x(j,i*4+3)*fil(l,1)
          tt4=tt4+x(j,i*4+4)*fil(l,1)
       enddo
       y(i1,i*4+1)=tt1
```

```
            y(i1,i*4+2)=tt2
            y(i1,i*4+3)=tt3
            y(i1,i*4+4)=tt4
        enddo
enddo
```

In general we need to take care of the remaining part of the data that doesn't fit into a block of size 4.

```
do i=(ndat/4)*4+1,ndat
  do i1=0,n1
     tt=0.d0
     do l=lowfil,lupfil
        j=mod_arr1(i1+l)
        tt=tt+x(j   ,i)*fil(l,1)
     enddo
     y(i1,i)=tt
  enddo
enddo
```


Obtain initial guess for rho and psi_i
Calculate the gradients G_i=(H-E)psi_i
Precondition the gradients: P_i=A G_i
Go along the gradients: psi_i:=psi_i-alpha P_i
Orthogonalize the psi_i
Update the density: rho= Sum_i f_i(psi_i)^2
Update the potential

Preconditioning

$$P_i = AG_i \approx (H - E_i)^{-1} G_i$$

Applying the Hamiltonian for preconditioning: takes too much flops.

Therefore, assume:

$$P_i \approx (-\frac{1}{2}\Delta + C)^{-1} G_i$$

How to invert $-\frac{1}{2}\Delta + C$?

- Diagonal preconditional for wavelets
- Fourier-based preconditioning for scaling functions
- Iterative preconditioning via conjugate gradient (optional)

The diagonal approximation for wavelets:

$$P_i \approx (-\frac{1}{2}\Delta + C)^{-1} G_i \approx \left(\text{diag}(-\frac{1}{2}\Delta + C)\right)^{-1}$$

The Galerkin matrix of the (shifted) Laplacian in real space for the scaling functions:

$$
\begin{aligned}
T_{i_1,i_2,i_3,j_1,j_2,j_3} &= \frac{1}{2h_1^2} a_{i_1-j_1} \delta_{i_2-j_2} \delta_{i_3-j_3} + \\
+ \quad & \frac{1}{2h_2^2} \delta_{i_1-j_1} a_{i_2-j_2} \delta_{i_3-j_3} + \frac{1}{2h_3^2} \delta_{i_1-j_1} \delta_{i_2-j_2} a_{i_3-j_3} + \\
+ \quad & C \delta_{i_1-j_1} \delta_{i_2-j_2} \delta_{i_3-j_3}
\end{aligned}
$$

FFT in 3 dimensions: the result is,

$$
\begin{aligned}
& T_{k_1,k_2,k_3,k_1',k_2',k_3'} = \\
= & \left[\frac{\bar{a}(k_1)}{2h_1^2} + \frac{\bar{a}(k_2)}{2h_2^2} + \frac{\bar{a}(k_3)}{2h_3^2} + C\right] \delta_{k_1-k_1'} \delta_{k_2-k_2'} \delta_{k_3-k_3'}
\end{aligned}
$$

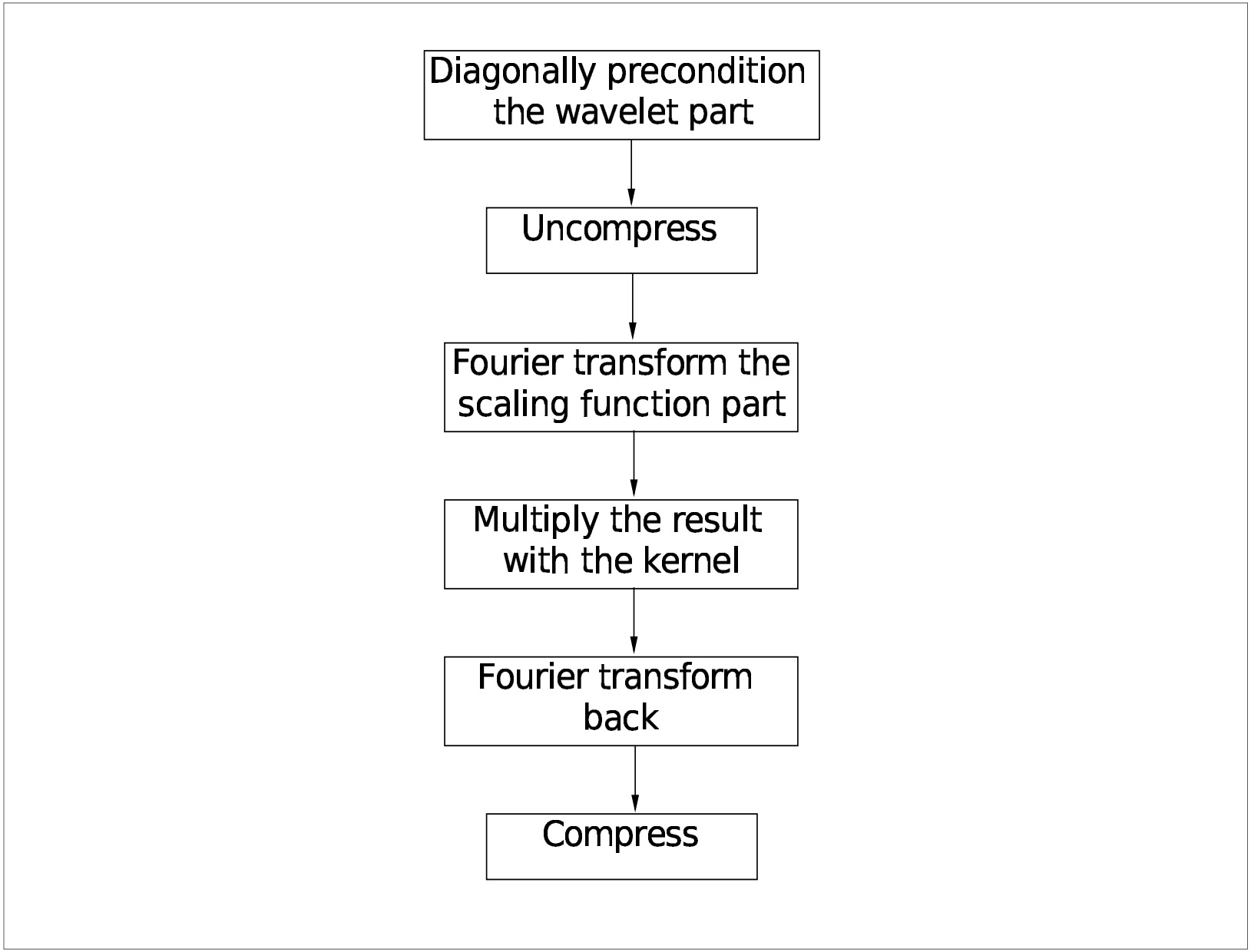


Figure 7: The mixed diagonal-Fourier preconditioning, periodic boundary conditions

Surface boundary conditions.

The Galerkin matrix of the (shifted) Laplacian in real space for the scaling functions looks similar to the fully periodic case. Its periodicity is, of course, different now.

$$
\begin{aligned}
T_{i_1,i_2,i_3,j_1,j_2,j_3} &= \frac{1}{2h_1^2} a_{i_1-j_1} \delta_{i_2-j_2} \delta_{i_3-j_3} + \\
+ \quad & \frac{1}{2h_2^2} \delta_{i_1-j_1} a_{i_2-j_2} \delta_{i_3-j_3} + \frac{1}{2h_3^2} \delta_{i_1-j_1} \delta_{i_2-j_2} a_{i_3-j_3} + \\
+ \quad & C \delta_{i_1-j_1} \delta_{i_2-j_2} \delta_{i_3-j_3}
\end{aligned}
$$

FFT in the x and z direction: the result is,

$$T_{k_1,i_2,k_3,k_1',j_2,k_3'} =$$

$$= \left[\frac{a_{i_2-j_2}}{2h_2^2} + \left(\frac{\bar{a}(k_1)}{2h_1^2} + \frac{\bar{a}(k_3)}{2h_3^2} + C\right)\delta_{i_2-j_2}\right]\delta_{k_1-k_1'}\delta_{k_3-k_3'}$$

This is a sparse (banded) matrix in i_2, j_2 that can be inverted. The inversion time is proportional to $n2$, the dimension of the simulation box in the y direction.

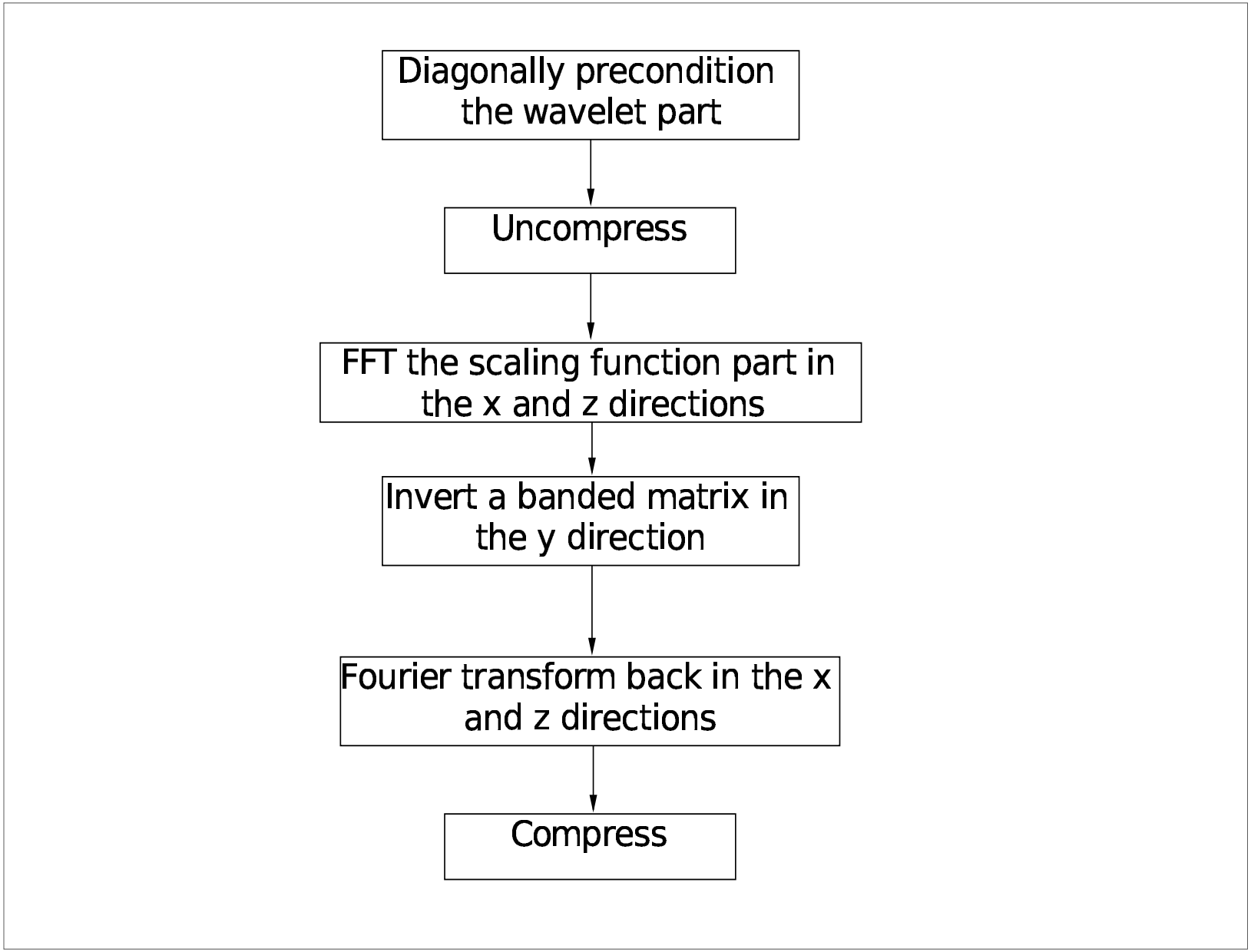


Figure 8: The mixed diagonal-Fourier preconditioning, surface boundary conditions

Iterative preconditioning

This time, iteratively compute

$$P_i \approx (-\frac{1}{2}\Delta + C)^{-1} G_i$$

This is

- more precise
- more slowly

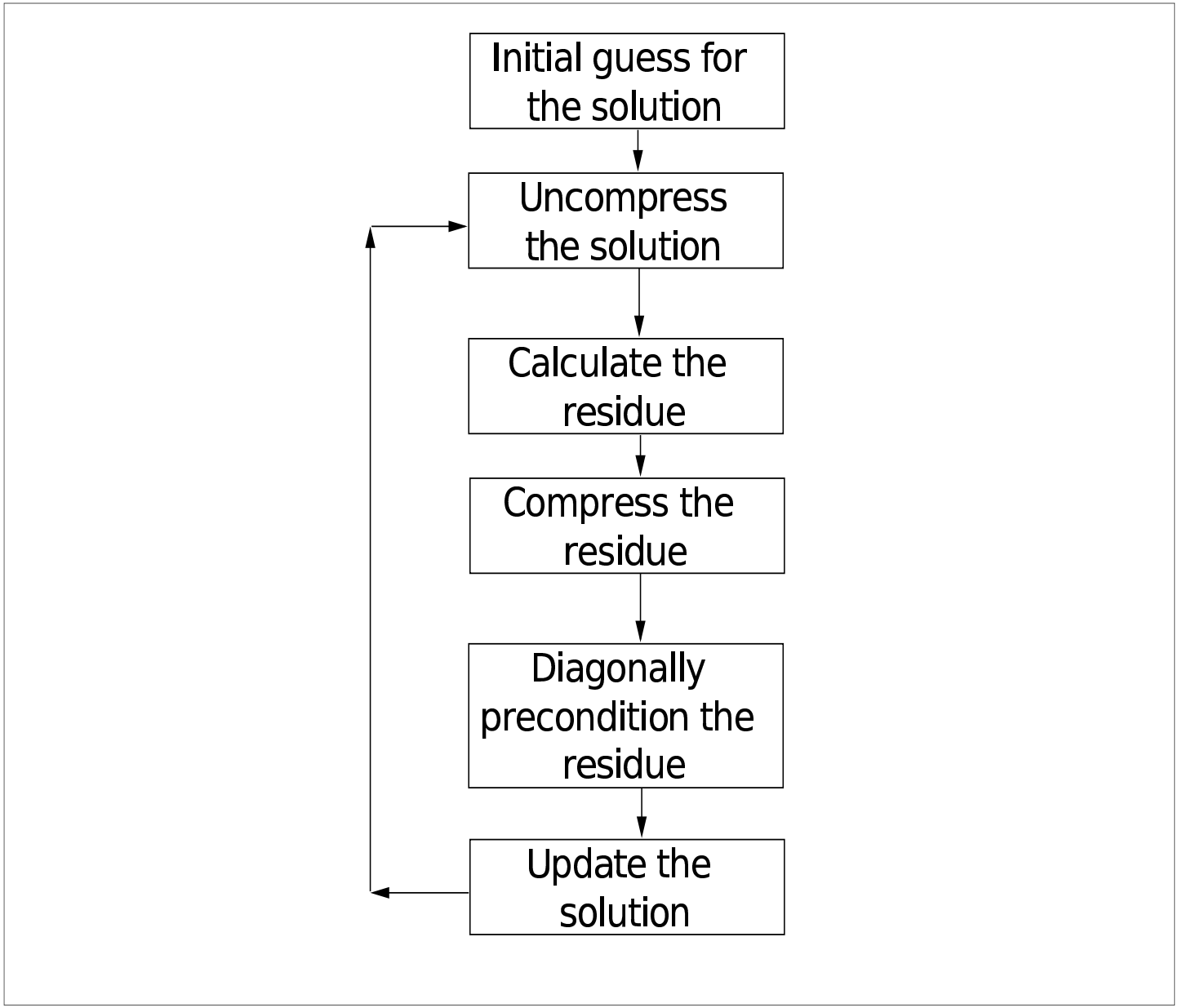


Figure 9: The iterative solution of the preconditioning equation

Performance

Tests on SiH4

- Free boundary conditions: 6 iterations give the fastest convergence
- Periodic BC: 1 iteration gives the fastest convergence

Periodic cell of size 18 a.u. gives roughly the same energy as in the free case. The CPU time spent is also similar.

FFT will be accelerated in near future.

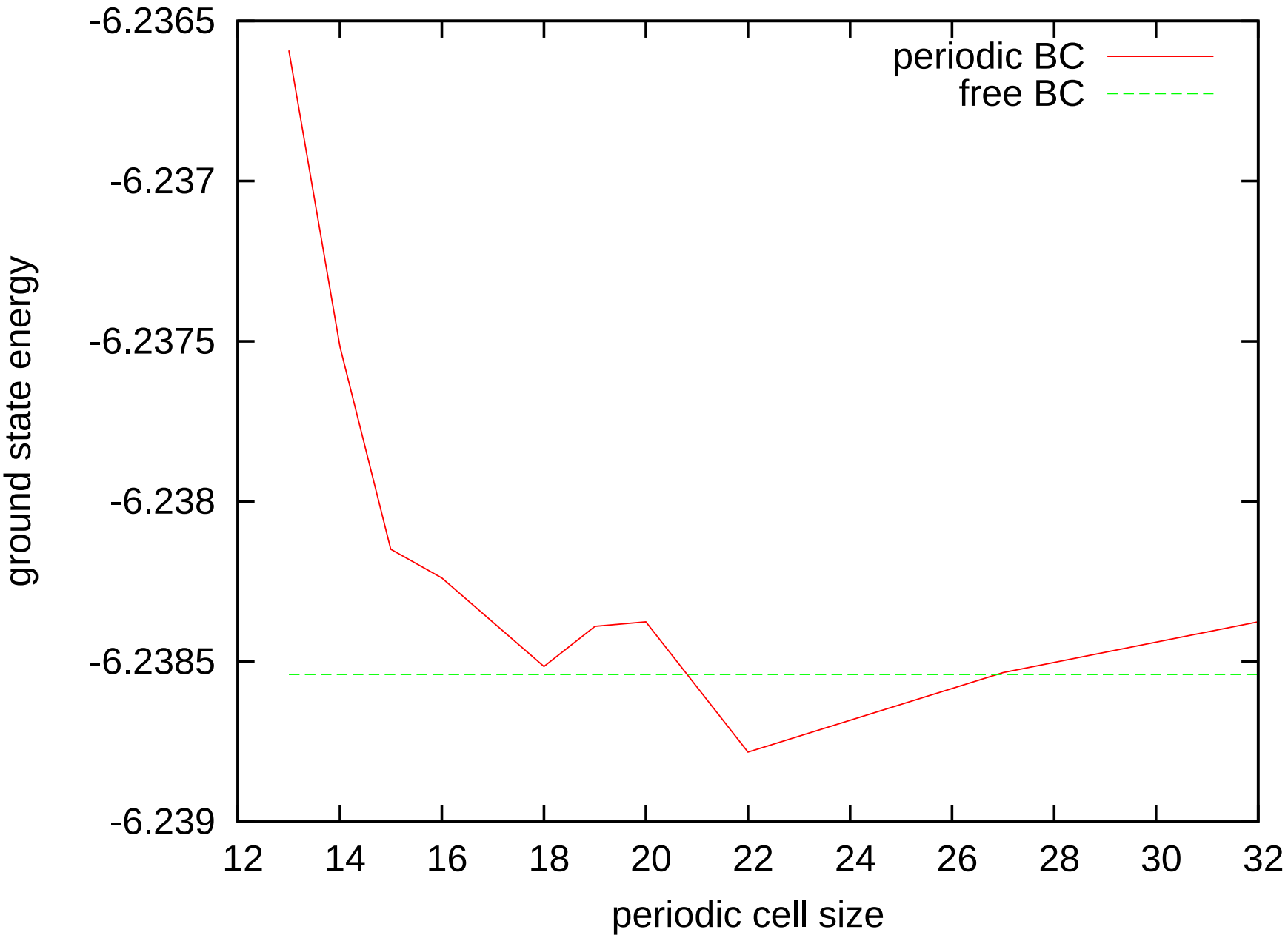


Figure 10: Ground state energy of the SiH4 molecule vs. the periodic cell size

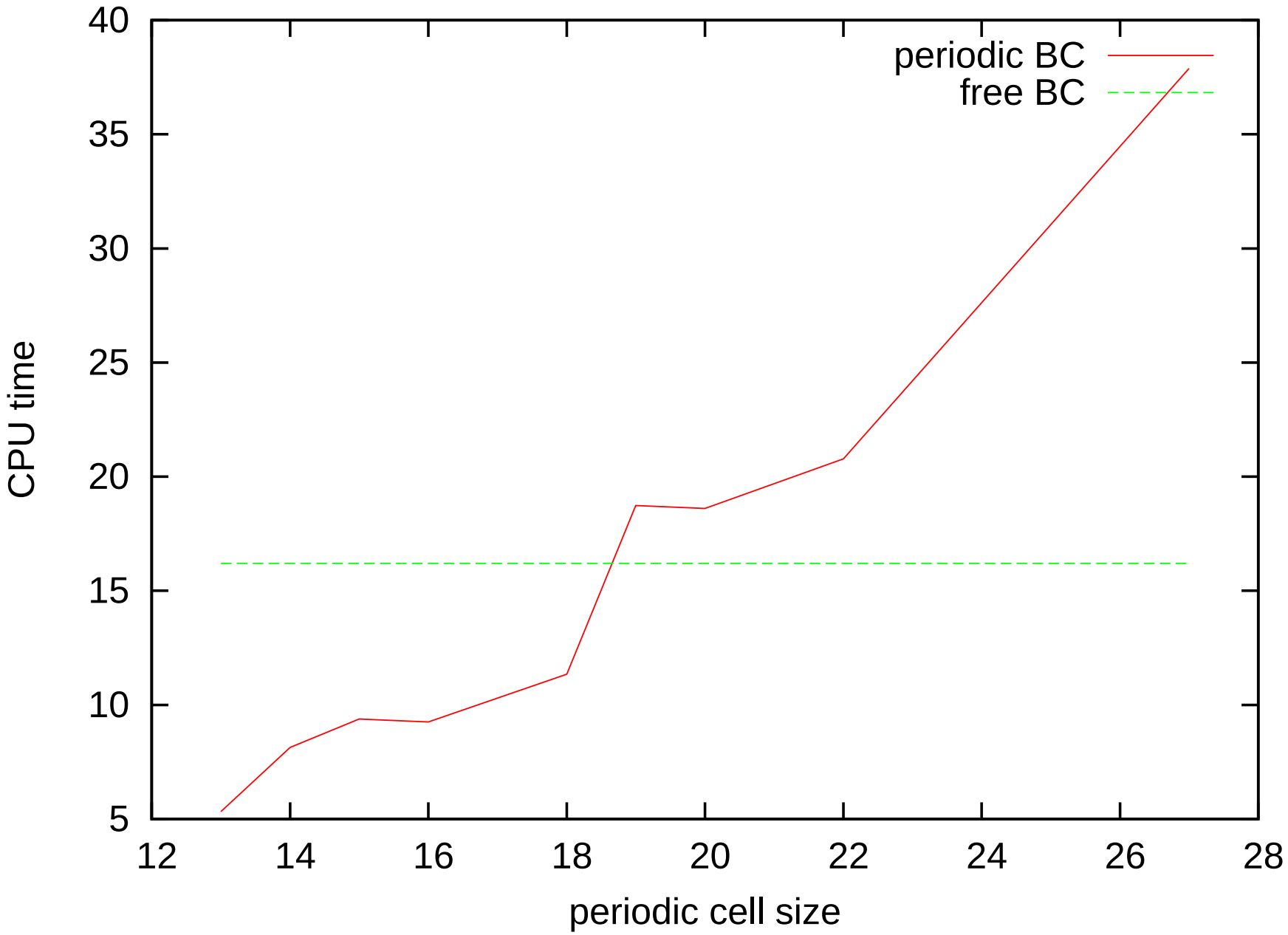


Figure 11: CPU time spent by the BigDFT on the SiH4 molecule vs. the periodic cell size