

Sparse matrix based implementation of the DFTB method

B. Aradi, B. Hourahine, C. Köhler, T. Frauenheim

Overview

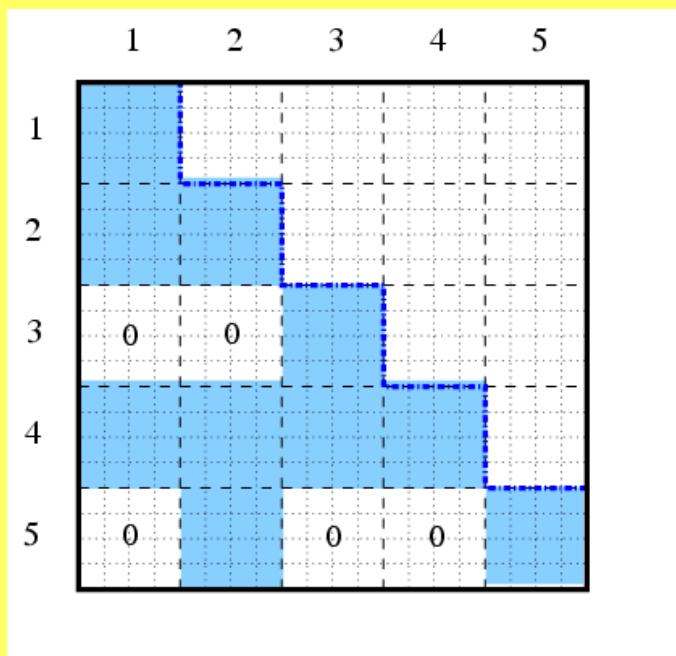
- Sparsity of the DFTB matrices
- Packed matrix format
- Analysis in packed format: Mullikan charges
- The DFTB⁺ Implementation
- Current features
- Planned features
- Look inside

Vanishing matrix elements

Hamiltonian matrix element for spin polarised DFTB calculation:

$$H_{\mu\nu\sigma} = \langle \phi_\mu | H_0 | \phi_\nu \rangle + \frac{1}{2} S_{\mu\nu} \sum_C \sum_{l'' \in C} (\gamma_{A(\mu)l(\mu), Cl''} + \gamma_{B(\nu)l(\nu), Cl''}) \Delta q_{Cl''}$$

$$\pm S_{\mu\nu} \left(\sum_{l' \in B(\mu)} W_{B(\mu)l(\mu)l'} p_{B(\mu)l'} + \sum_{l' \in B(\nu)} W_{B(\nu)l(\nu)l'} p_{B(\nu)l'} \right)$$



Matrix elements vanish, if

$$\langle \phi_\mu | H_0 | \phi_\nu \rangle = 0 \quad S_{\mu\nu} = \langle \phi_\mu | \phi_\nu \rangle = 0$$

Cutoff radius independent of system size.

Nr. of nonzero matrix elements scales as $O(N)$.

No need to calculate and store zeros.

Sparse storage: Non-periodic case

Storage order:

$1s1s, 1s1p_x, 1s1p_y, \dots, 1p_x1s, 1p_x1p_x, \dots, 1p_z1p_z$

$1s2s, 1s2p_x, 1s2p_y, \dots, 1p_x2s, 1p_x2p_x, \dots, 1p_z1p_z$

...

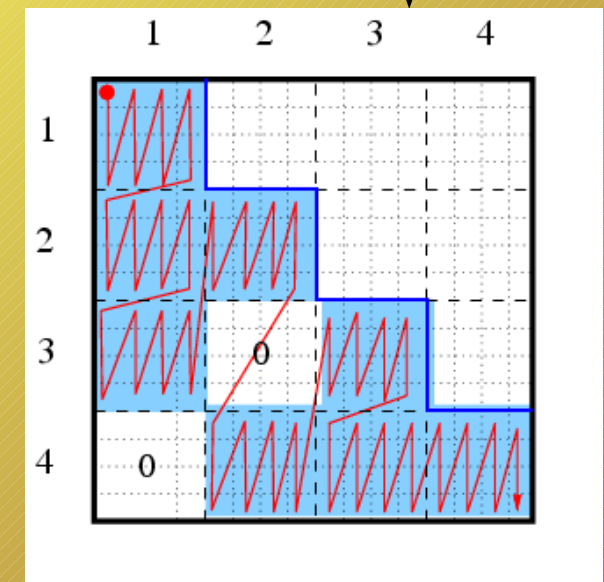
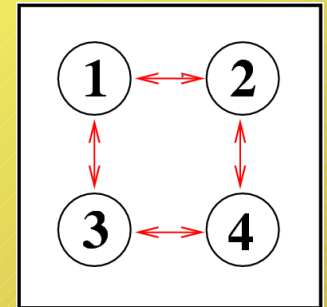
$4s4s, 4s4p_x, 4s4p_y, \dots, 4p_x4s, 4p_x4p_x, \dots, 4p_z4p_z$

Due to the symmetry of H, S, etc., only one triangle has to be stored.

For on-site blocks both triangles are stored

Accessing: Neighbor map and indexing arrays

```
do iAt1 = 1, nAtom
  do iNeigh = 1, nNeighbors(iAt1)
    iAt2 = iNeighbors(iNeigh, iAt1)
    ind = iPair(iAt2, iAt1)
    over(ind) = someFunc(iAt2, iAt1)
  end do
end do
```



Periodic case

Bloch functions:

$$\beta_{\mu}^k(\mathbf{r}) = \frac{1}{\sqrt{N}} \sum_{\mathbf{R}} \phi_{\mu}(\mathbf{r}) e^{i\mathbf{k}\mathbf{R}}$$

$$\langle \beta_{\mu}^k | O | \beta_{\nu}^k \rangle = O_{\mu\nu}(\mathbf{k}) = \sum_{\mathbf{R}} e^{i\mathbf{k}\mathbf{R}} O_{\mu\nu}(\mathbf{R})$$

$$O_{\mu\nu}(\mathbf{R}) = \langle \phi_{\mu}(0) | O | \phi_{\nu}(\mathbf{R}) \rangle$$

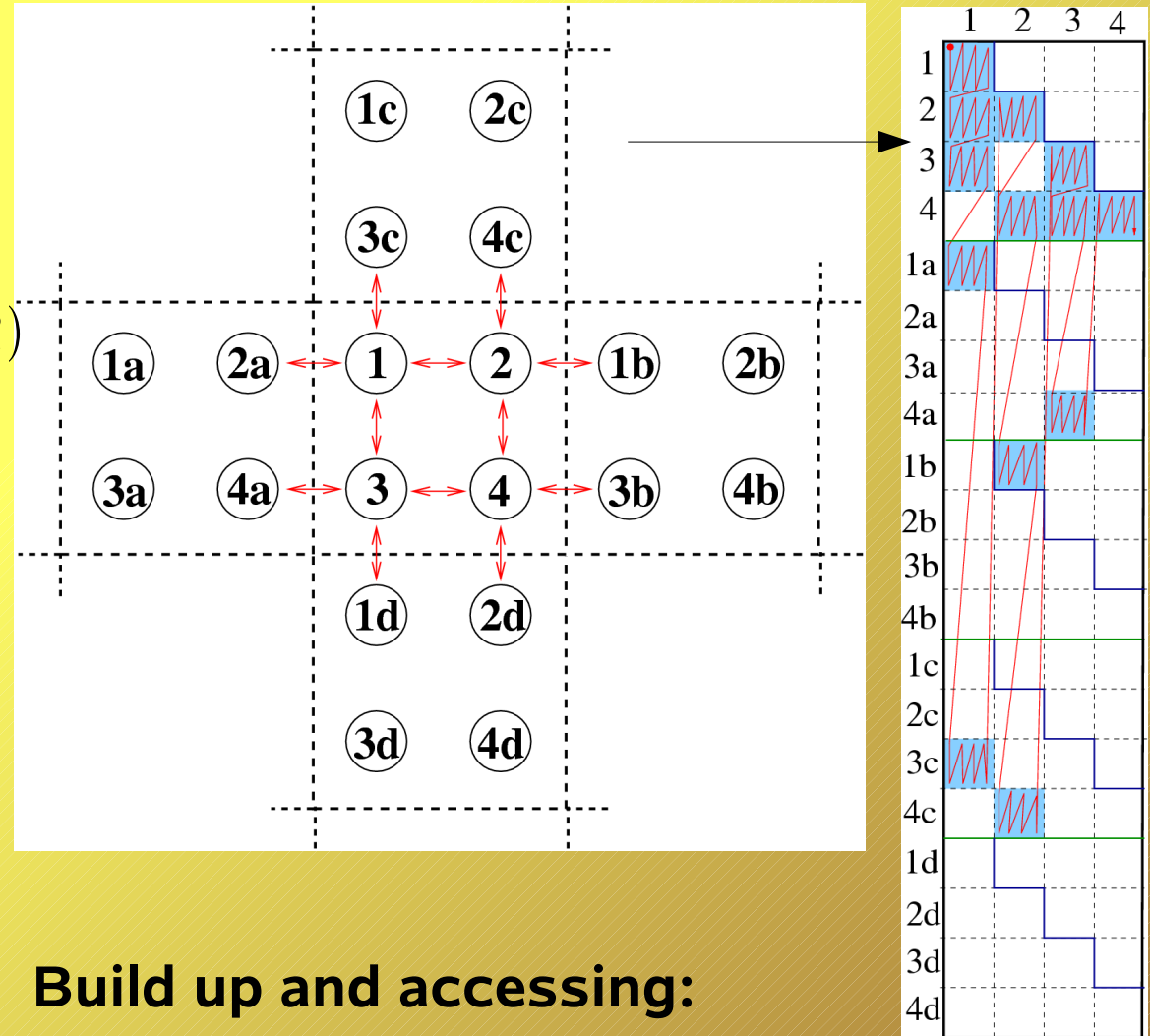
Real space rectangular matrix

Symmetry of the interaction:

$$\langle \phi_2 | \phi_{1b} \rangle = \langle \phi_{1b} | \phi_2 \rangle^* = \langle \phi_1 | \phi_{2a} \rangle^*$$



Only lower triangle of every block must be considered



Build up and accessing:

Same as for non-periodic systems

Periodic case (cont.)

Transformation between real and k-space:

Real rectangular \rightarrow complex square:

$$O_{\mu\nu}(\mathbf{k}) = \sum_{\mathbf{R}} e^{i\mathbf{k}\mathbf{R}} O_{\mu\nu}(\mathbf{R}) \qquad O_{\mu\nu}(\mathbf{R}) = \langle \phi_{\mu}(0) | O | \phi_{\nu}(\mathbf{R}) \rangle$$

Complex square \rightarrow real rectangular:

$$O_{\mu\nu}(\mathbf{R}) = \sum_l \omega_l e^{-i\mathbf{k}_l\mathbf{R}} O_{\mu\nu}(\mathbf{k}_l)$$

(e.g. transforming density matrix to rectangular form)

If k-point sampling is good, forward then back transformation = Identity

Workflow

0. Determining neighbor map.
1. Building sparse matrixes H and S.
2. Converting sparse H and S to square form.
3. Solving dense eigenproblem.
- 4a. Creating dense density matrix and transforming to sparse form
or
- 4b. Creating sparse density matrix directly
5. Analysis in sparse form (Mullikan, forces, etc.)

Mullikan analysis in real space

Periodic case: Basis functions = Bloch functions

$$\psi_i^{\mathbf{k}}(\mathbf{r}) = \sum_{\mu} c_{i\mu}^{\mathbf{k}} \beta_{\mu}^{\mathbf{k}}(\mathbf{r}) \quad \beta_{\mu}^{\mathbf{k}}(\mathbf{r}) = \frac{1}{\sqrt{N}} \sum_{\mathbf{R}} \phi_{\mu}(\mathbf{r}) e^{i\mathbf{k}\mathbf{R}}$$

Total number of electrons: Weighted sum of Mullikan charges

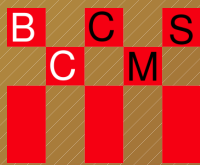
$$q_i^{\mathbf{k}} = n_i^{\mathbf{k}} |\psi_i^{\mathbf{k}}|^2 = n_i^{\mathbf{k}} \sum_{\mu} \sum_{\nu} c_{i\mu}^{\mathbf{k}*} c_{i\nu}^{\mathbf{k}} \langle \beta_{\mu}^{\mathbf{k}} | \beta_{\nu}^{\mathbf{k}} \rangle \quad \langle \beta_{\mu}^{\mathbf{k}} | \beta_{\nu}^{\mathbf{k}} \rangle = \sum_{\mathbf{R}} e^{i\mathbf{k}\mathbf{R}} S_{\mu\nu}(\mathbf{R})$$

$$N_e = \sum_l \omega_l \sum_i^{\text{occ}} q_i^{\mathbf{k}_l} = \sum_i \sum_l \omega_l n_i^{\mathbf{k}_l} \sum_{\mu} \sum_{\nu} c_{i\mu}^{\mathbf{k}_l*} c_{i\nu}^{\mathbf{k}_l} \sum_{\mathbf{R}} e^{i\mathbf{k}_l\mathbf{R}} S_{\mu\nu}(\mathbf{R})$$

Introducing density matrix: $P_{\mu\nu}$

$$P_{\mu\nu}(\mathbf{k}) = \sum_i^{\text{occ}} n_i^{\mathbf{k}} c_{i\mu}^{\mathbf{k}*} c_{i\nu}^{\mathbf{k}}$$

$$N_e = \sum_{\mathbf{R}} \sum_{\mu} \sum_{\nu} S_{\mu\nu}(\mathbf{R}) \sum_l \omega_l P_{\mu\nu}(\mathbf{k}_l) e^{i\mathbf{k}_l\mathbf{R}}$$



Mullikan analysis in real space (cont.)

$$N_e = \sum_{\mathbf{R}} \sum_{\mu} \sum_{\nu} S_{\mu\nu}(\mathbf{R}) \sum_l \omega_l P_{\mu\nu}(\mathbf{k}_l) e^{i\mathbf{k}_l \mathbf{R}}$$

Introducing packed (real space) density matrix:

$$P_{\mu\nu}(\mathbf{R}) = \sum_l \omega_l P_{\mu\nu}(\mathbf{k}_l) e^{-i\mathbf{k}_l \mathbf{R}} = \sum_l \omega_l P_{\mu\nu}(\mathbf{k}_l) e^{i\mathbf{k}_l \mathbf{R}}$$

Due to time reversal symmetry

Total charge / orbital charge (pointwise multiplication and summation)

$$N_e = \sum_{\mathbf{R}} \sum_{\mu} \sum_{\nu} S_{\mu\nu}(\mathbf{R}) P_{\mu\nu}(\mathbf{R}) = \sum_{\nu} q_{\nu}$$

$$q_{\nu} = \sum_{\mathbf{R}} \sum_{\nu} S_{\mu\nu}(\mathbf{R}) P_{\mu\nu}(\mathbf{R})$$

Similar for non-scc forces, non-scc energy

Scaling properties

- **Parts, not scaling $O(N)$**
 - SCC-DFTB: long range part of γ : $O(N^2)$ summation
 - Could be replaced by alternative $O(N)/O(N \log N)$ scheme
 - Diagonalisation in dense form $O(N^3)$ – could be replaced by
 - Sparse diagonaliser (PETSc, M. Sternberg)
 - $O(N)$ scheme (e.g. Divide and Conquer, W. Yang)
- **Advantages of the sparse matrix form**
 - Reduced storage
 - Important, if many copies must be stored (e.g. DFTB+U mixes density matrices)
 - Faster operation for big systems (Matrix construction, Mullikan-analysis, force calculation)

DFTB⁺ 1.0

- **All features from the old Paderborn DFTB code:**
 - Non-scc/scc calculations
 - Geometry optimisation
 - Steepest descent
 - Conjugate gradient
 - Partial geometry optimisation
 - Geometry optimisation constraints in xyz-coordinates
 - Molecular dynamics (Anderson thermostat)
 - Dispersion correction
 - Broyden charge mixer (very efficient)
 - Periodic boundary conditions (Γ point only)

DFTB⁺ 1.0 (cont.)

- **Additions to the functionality of the old Paderborn DFTB:**
 - Optionally l-shell resolved
 - K-point sampling and band structure
 - Spin polarisation (see C. Köhler)
 - Variational energy (see B. Hourahine)
 - Ability to treat f-electrons (see S. Sanna and B. Hourahine)
 - LDA+U extension (see B. Hourahine)
 - Modified γ for H (see M. Elstner)
 - External charges (with smoothing function for clusters)
 - User choice of LAPACK dense diagonalisers
 - standard, divide-and-conquer, relatively robust

DFTB⁺ 1.0 (cont. #2)

- **Additions to the functionality of the old Paderborn DFTB:**
 - New mixers
 - Simple, Andersen, DIIS
 - Improved molecular dynamics
 - Mermin free energy conserved
 - User choice of filling function
 - Fermi, Methfessel-Paxton (arbitrary order)
 - OpenMP parallelisation
 - Automatic code validation
 - Automatic testing and comparing against standard set of systems
 - Tool for plotting orbitals, charge densities etc. (Waveplot)
 - New user friendly, extensible input format

DFTB – Human-friendly Structured Data Input

Internal: XML DOM, External: XML or **Human-friendly Structured Data**

```
# Reading geometry from disc in gen format
```

```
Geometry = GenFormat {  
  <<< dna.gen  
}
```

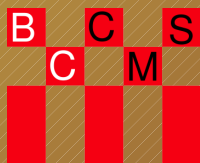
Practically
everything can be
specified in the input

```
# Defining properties of DFTB Hamiltonian
```

```
Hamiltonian = DFTB {  
  SCC = Yes # SCC calculation  
  Variational = Yes # Variational en.  
  Mixer = Broyden { # Broyden Mixer  
    MixingParameter = 0.2  
  }  
  Filling = Fermi { # Fermi filling  
    Temperature [Kelvin] = 300  
  }  
  Eigensolver = DivideAndConquer {} # Fast!  
  ...  
}
```

no magic constants

sensible defaults
for parameters



Benchmarks / Release

- **Should DFTB⁺ be slower than DFTB?**
 - I-shell resolved
 - Rewritten in a more complex language
 - Nearly object oriented approach
 - Modularised
 - No quick and dirty solutions
- **Good news: It is faster than old DFTB!**
 - Time per SCC cycle lower (with standard lapack diagonaliser)
 - 128 atoms, periodic, with forces: 5% faster
 - Default diagonaliser (zhegvd/dsygvd): at least 2 times faster.
- **Official release:** End of October.

