ASI: Atomic Simulation Interface

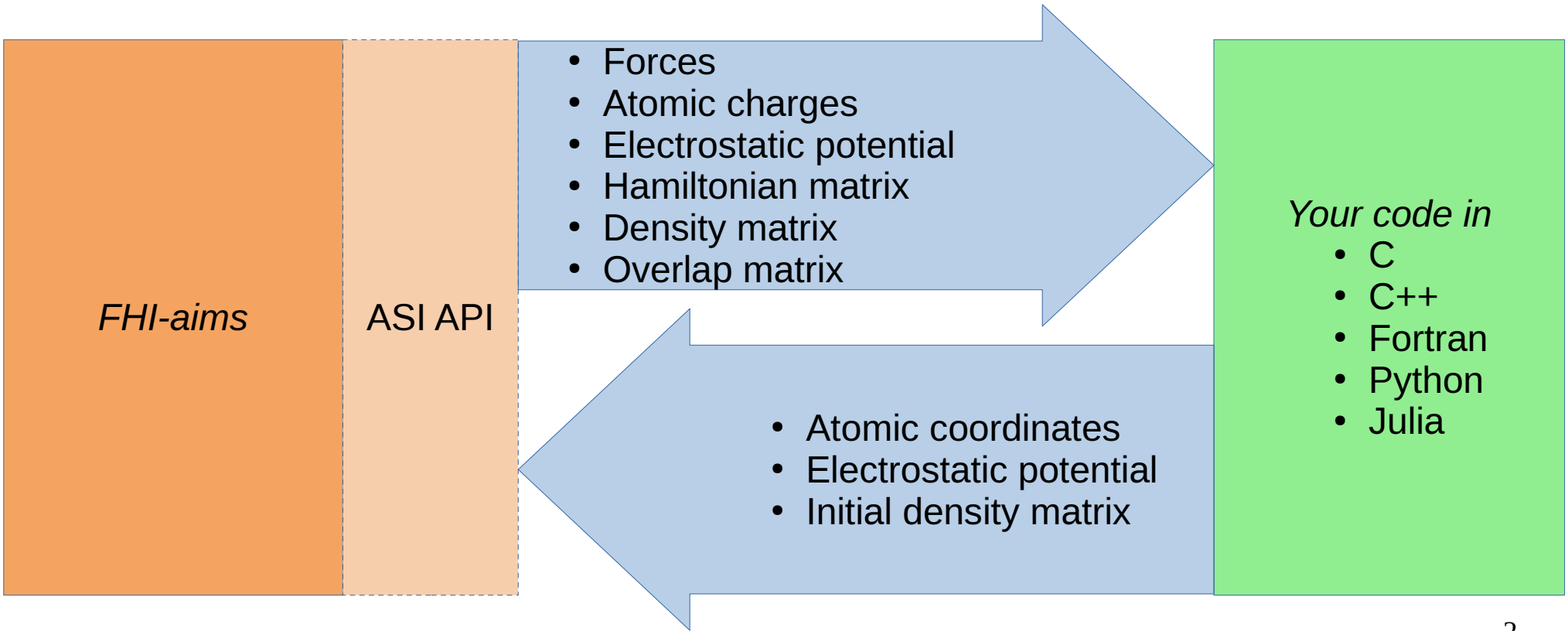
Pavel Stishenko, Benjamin Hourahine, Volker Blum,
Reinhard Maurer, Scott Woodley, Andrew J Logsdail
stishenkop@cardiff.ac.uk

FHI-aims Developers' and Users' Meeting 2023

August 2 – August 4, 2023

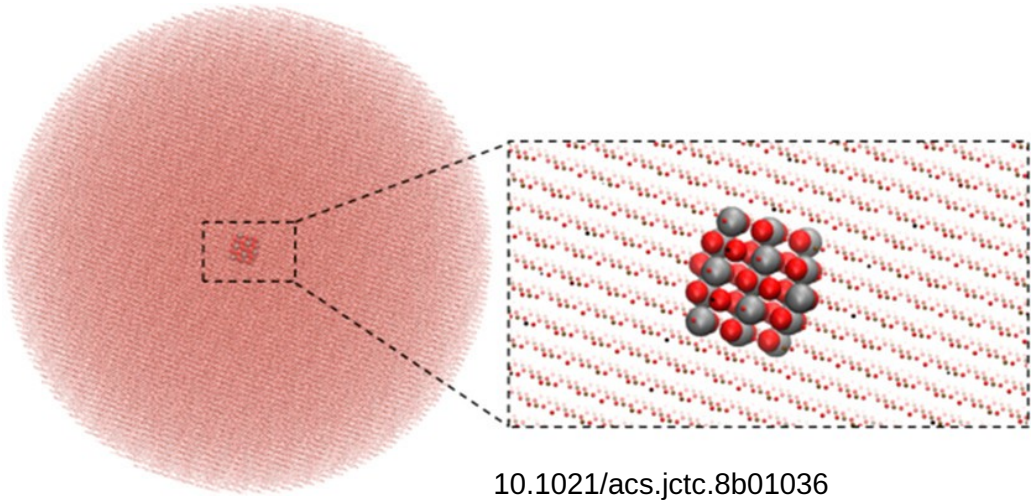
Center of Free Electron Laser Science (CFEL), Hamburg, Germany

ASI – **A**pplication **P**rogramming **I**nterface



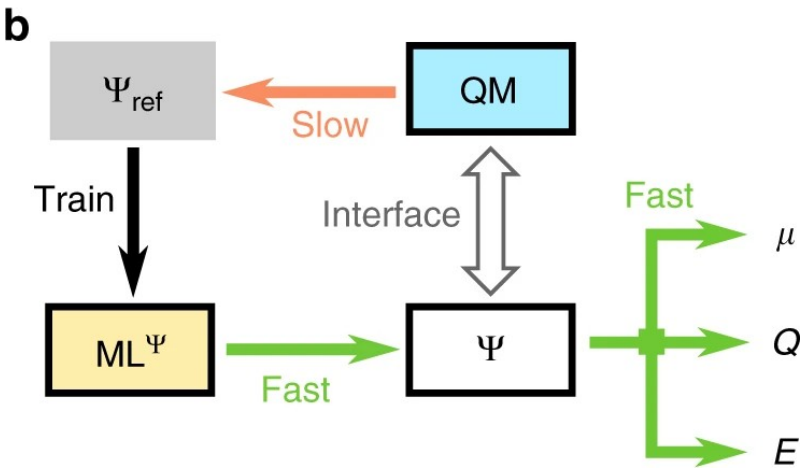
Motivation

QM/MM



10.1021/acs.jctc.8b01036

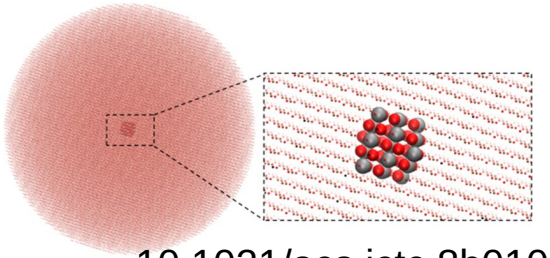
QM/ML



10.1038/s41467-019-12875-2

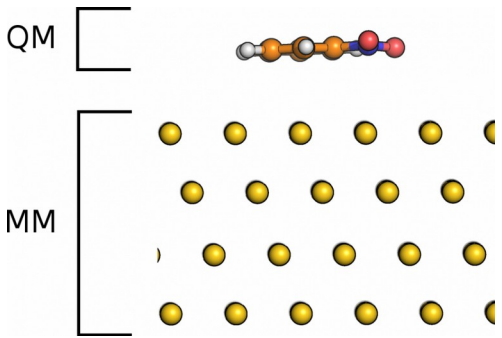
Electrostatic embedding

- ChemShell QM/MM
- Image charge augmented QM/MM (CP2K)



10.1021/acs.jctc.8b01036

ASI helps to avoid proxy charges ESP representation



10.1021/ct400698y

A review: 10.1039/D2CP04537K

ML models of electronic structure

- DFTB deep learning (small hydrocarbons, bulk aluminium)

A Density Functional Tight Binding Layer for Deep Learning of Chemical Hamiltonians

Haichen Li, Christopher Collins, Matteus Tanha, Geoffrey J. Gordon, and David J. Yaron*

Cite this: *J. Chem. Theory Comput.* 2018, 14, 11, 5764−5776
Publication Date: October 15, 2018
https://doi.org/10.1021/acs.jctc.8b00873

Article Views 2926 | Altmetric 17 | Citations 63

LEARN ABOUT THESE METRICS

Equivariant analytical mapping of first principles Hamiltonians to accurate and transferable materials models

Liwei Zhang, Berk Onat, Geneviève Dusson, Adam McSloy, G. Anand, Reinhard J. Maurer, Christoph Ortner & James R. Kermode

npj Computational Materials **8**, Article number: 158 (2022) | Cite this article

- SchNOrb (water, ethanol, malondialdehyde, uraci)

Unifying machine learning and quantum chemistry with a deep neural network for molecular wavefunctions

K. T. Schütt, M. Gastegger, A. Tkatchenko, K.-R. Müller & R. J. Maurer

Nature Communications **10**, Article number: 5024 (2019) | Cite this article

- SA-GPR (small hydrocarbons)

Transferable Machine-Learning Model of the Electron Density

Andrea Grisafi, Alberto Fabrizio, Benjamin Meyer, David M. Wilkins, Clemence Corminboeuf, and Michele Ceriotti*

Cite this: *ACS Cent. Sci.* 2019, 5, 1, 57−64
Publication Date: December 26, 2018
https://doi.org/10.1021/acscentsci.8b00551

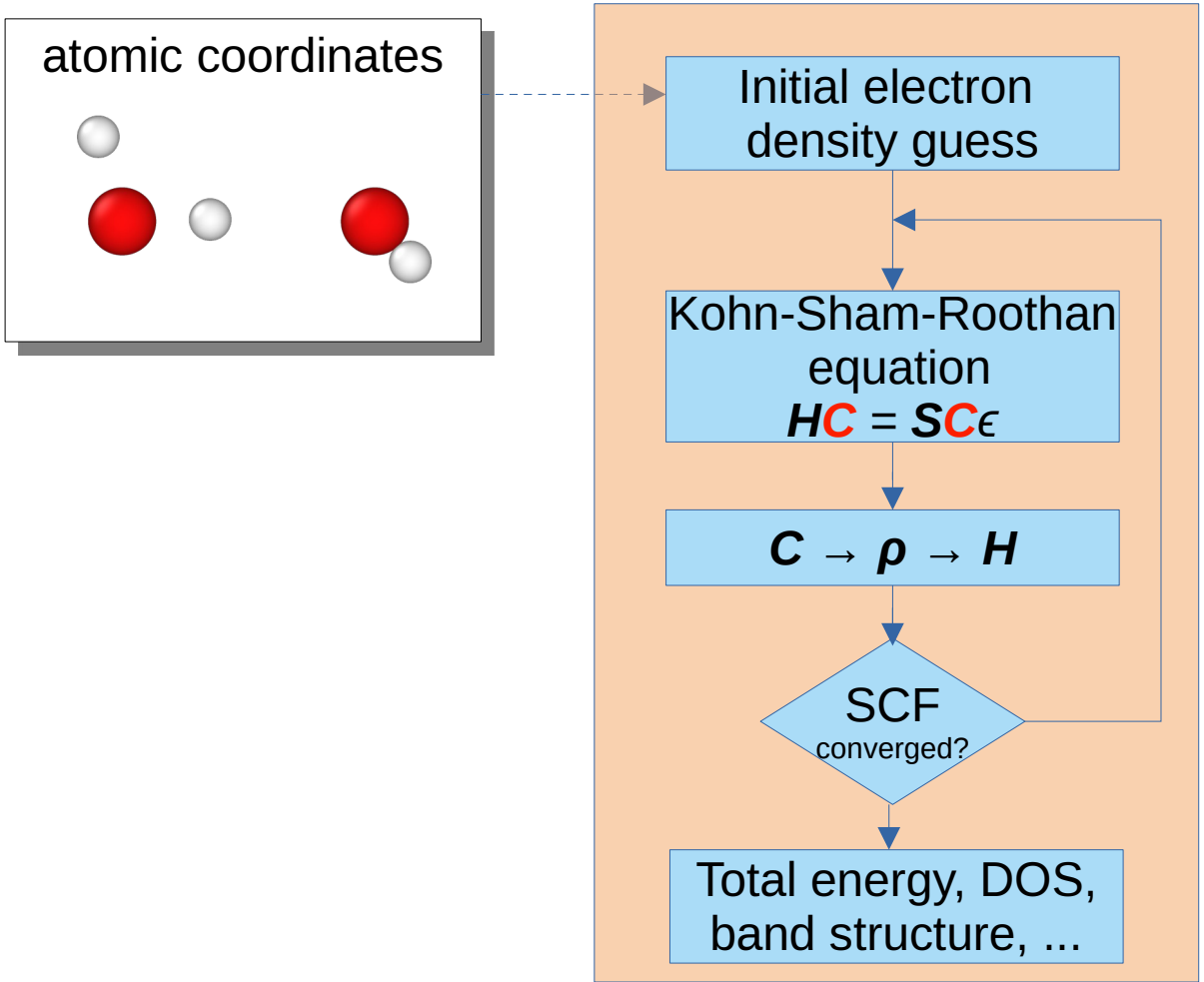
Article Views 11048 | Altmetric 55 | Citations 140

LEARN ABOUT THESE METRICS

ML models of electronic structure

- DFTB deep learning (small hydrocarbons, bulk aluminium)
    - Predicted quantity: ***H***
- SchNOrb (water, ethanol, malondialdehyde, uraci)
    - Predicted quantities: ***H, S***
- SA-GPR (small hydrocarbons)
    - Predicted quantities: ***ρ***

Core DFT algorithm



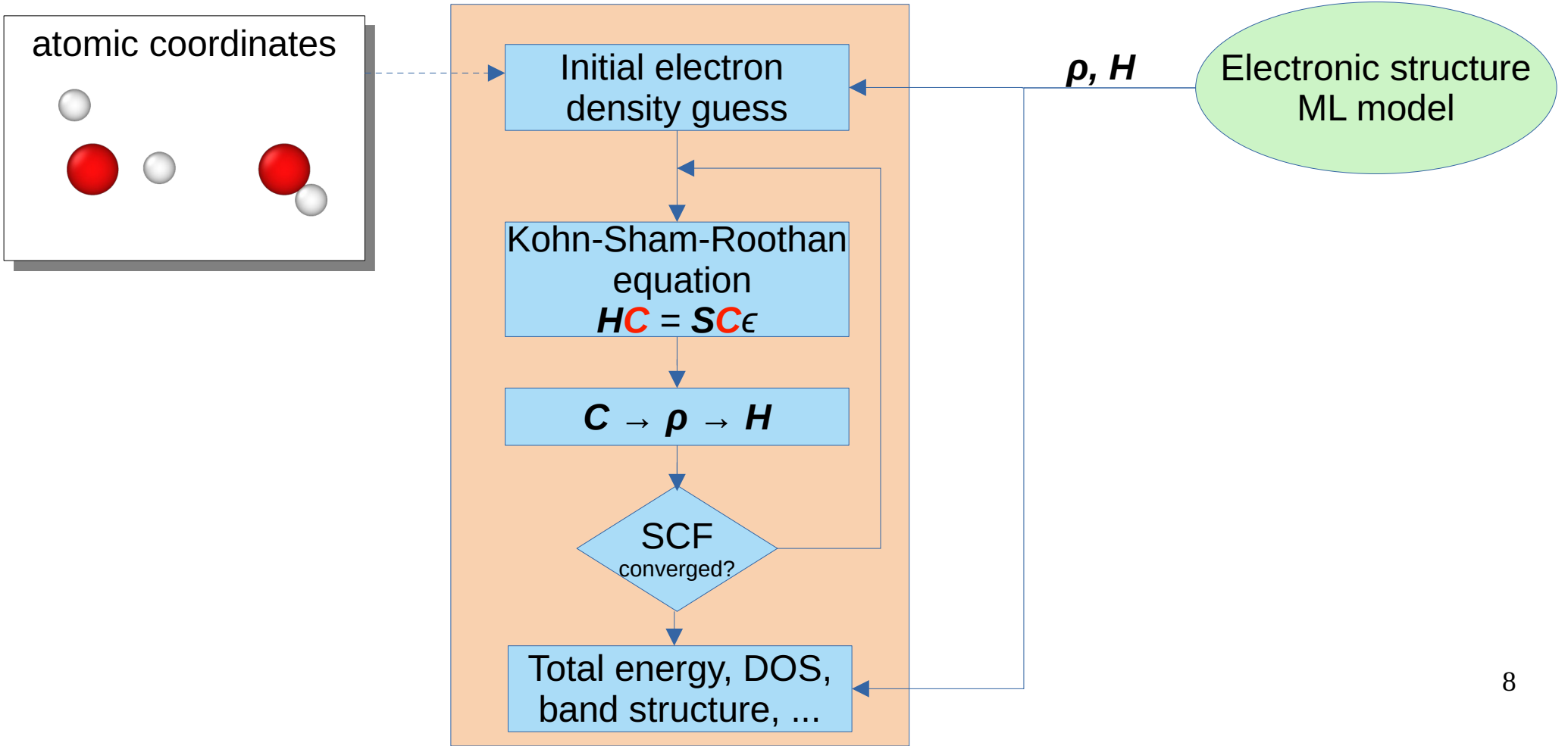
H – Hamiltonian operator

C – wave function expansion coefficients

S – overlap matrix

ρ – electronic density

Place of electronic structure ML



Difficulties of application of electronic structure ML models

- **Huge size** of predicted data arrays (matrices, fields): 200 MB per matrix for a 100 water molecules

- Lack of **efficient** and **convenient** ways for import and export such data

API’s of FHI-aims

- **i-PI**
  - widely adopted, **ASE** wrapper, **MolSSI** Driver Interface wrapper
  - classical quantities (small data)
- **Text I/O**
  - great ASE wrapper
  - classical quantities (small data)
- **File I/O (elsi_restart)**
  - need of conversion, synchronization
  - performance is tricky
  - only density matrix currently
- **CFFI (by Jan Hermann)**
  - Python-only
  - somewhat abandoned

ASI API Requirements

1. efficient for large data structures
   (high speed and low memory footprint)
2. easy-to-use
3. easy-to-implement
4. portable

ASI API is a plain C API

- Can be implemented efficiently
- No added complexity for deployment
- Portable
  - **Fortran** `iso_c_binding`
  - **Python** `ctypes, asi4py`
  - **Julia** `ccall`
- Compatible with
  - `MPI`
  - `BLAS`
  - `ScaLAPACK`

```
void ASI_init ( const char *  inputpath,
                const char *  outputfilename,
                int           mpiComm
              )
```

Key functionality of ASI API

- Control flow (minimal)
- Classical MD (for convenience)
- Electrostatic potential (QM/MM)
- Kohn-Sham-Roothaan matrices (QM/ML)

Total: ~24 functions

ASI control flow functions

Minimal set for the sake of simple and non-invasive implementation

```
void ASI_init ( const char *  inputpath,
                const char *  outputfilename,
                int           mpiComm
              )
```

```
void ASI_run ( )
```

```
void ASI_finalize ( )
```

ASI classical MD functions

void	**ASI_set_geometry** (const double *coords, int n_atoms=-1, const double *lattice=0)
int	**ASI_n_atoms** ()
double	**ASI_energy** ()
const double *	**ASI_forces** ()
const double *	**ASI_stress** ()
const double *	**ASI_atomic_charges** (int scheme=-1)

- Repeats i-PI functionality
- For the sake of convenience

ASI electrostatic potential functions

```
void   ASI_calc_esp (int n, const double *coords, double *potential, double *potential_grad)
void   ASI_set_external_potential (ASI_ext_pot_func_t callback, void *aux_ptr)
void   ASI_register_external_potential (ASI_ext_pot_func_t callback, void *aux_ptr)
```

```
typedef void(*  ASI_ext_pot_func_t) (void *aux_ptr, int n, const double *coords, double *potential, double
                *potential_grad)
```

- Use callbacks for setting ESP
- Two ways to set ESP (like in DFTB+):
  - before SCF loop
  - during SCF loop
- For QM/MM embedding

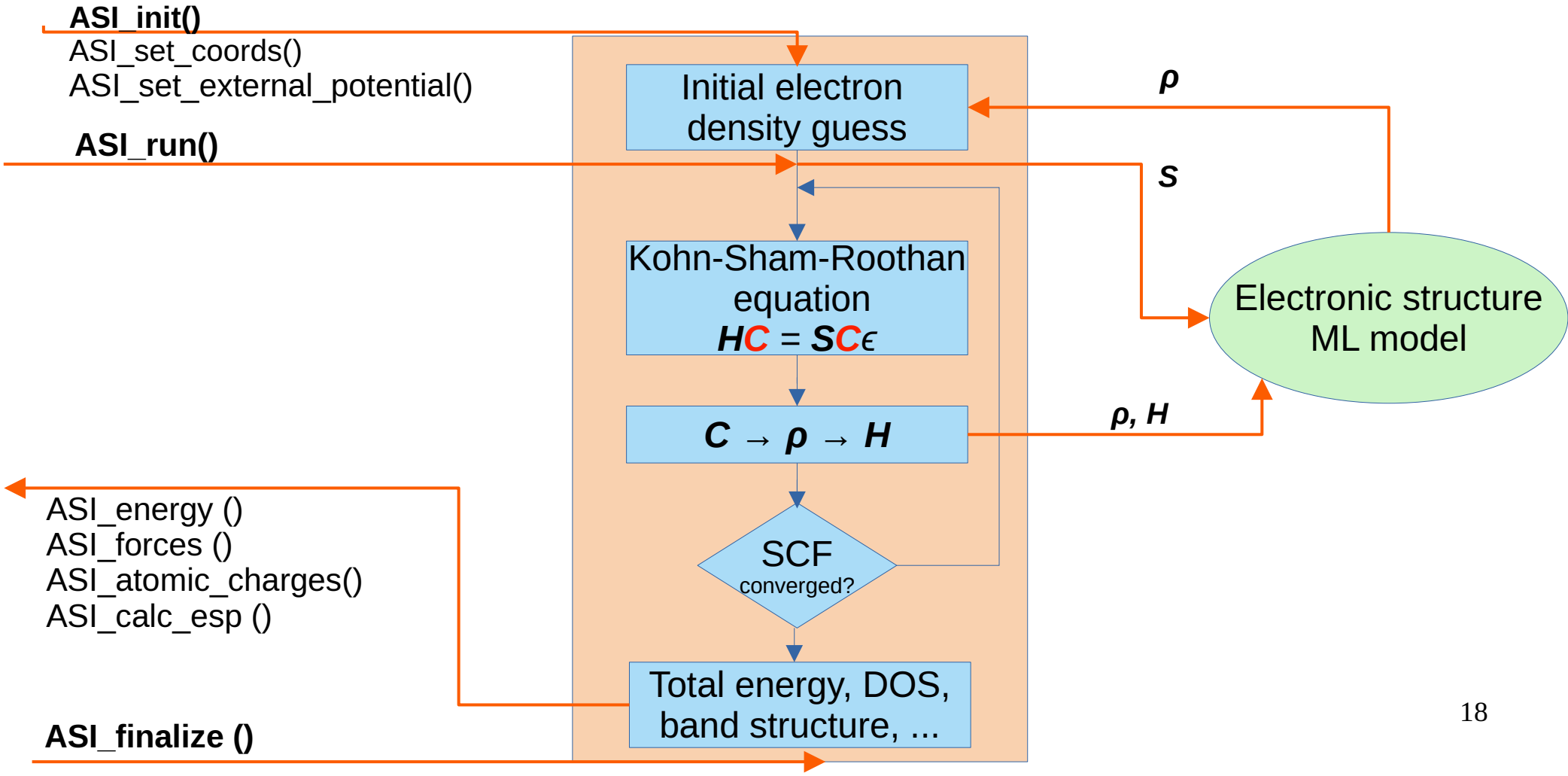
ASI for Kohn-Sham-Roothaan matrices

```
typedef void(*  ASI_dmhs_callback_t) (void *aux_ptr, int iK, int iS, int *blacs_descr, void *blacs_data)
void  ASI_register_dm_callback (ASI_dmhs_callback_t callback, void *aux_ptr)
void  ASI_register_overlap_callback (ASI_dmhs_callback_t, void *aux_ptr)
void  ASI_register_hamiltonian_callback (ASI_dmhs_callback_t, void *aux_ptr)
void  ASI_register_dm_init_callback (ASI_dmhs_callback_t, void *aux_ptr)
```

- Use callbacks for getting and setting matrices
- Use BLACS for distributed matrices
- Parallelization over **k**-points and spin channels

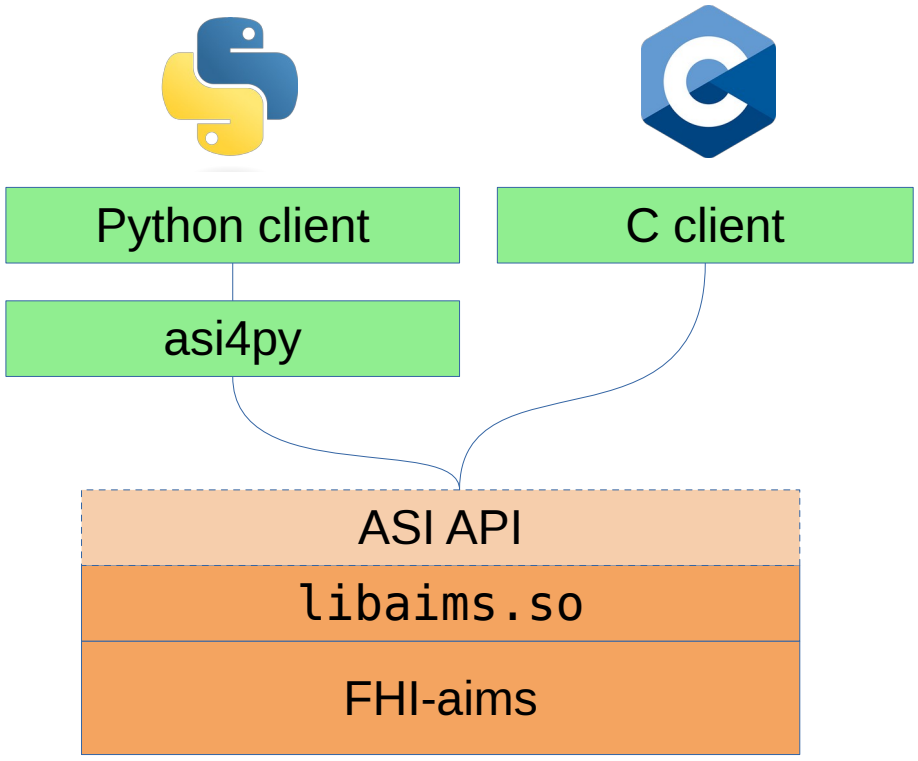
- For QM/ML methods

Place of ASI in FHI-aims



Implementation details

- Part of **FHI-aims**
  `set(BUILD_SHARED_LIBS ON CACHE STRING "")`
- Python wrapper **`asi4py`**
  `pip install asi4py`



Minimal C++ example

```cpp
int main(int argc, char *argv[])
{
  MPI_Init_thread(&argc, &argv, MPI_THREAD_FUNNELED, &mpi_provided_threading); // instea
  MPI_Comm_size(MPI_COMM_WORLD, &world_size);
  MPI_Comm_rank(MPI_COMM_WORLD, &world_rank);

  const MPI_Fint f_mpi_comm = MPI_Comm_c2f(MPI_COMM_WORLD);

  ASI_init("control.in", "asi.log", f_mpi_comm); // read geometry.in and control.in

  int  n_basis = ASI_get_basis_size();

  ASI_run();    // DO CALCULATIONS!

  auto E = ASI_energy();
  if (world_rank == 0)
    std::cout << "Energy == " << E  << " Ha = " << E * 27.2113845 << " eV" << std::endl;

  ASI_finalize();
  MPI_Finalize();
  return 0;
}
```

Python DM export callback

```
def default_saving_callback(aux, iK, iS, descr, data):
  try:
    asi, storage = cast(aux, py_object).value
    data = asi.scalapack.gather_numpy(descr, data, (asi.n_basis,asi.n_basis))
    if data is not None:
      storage[(iK, iS)] = data.copy()
  except Exception as eee:
    print (f"Something happened in ASI default_saving_callback : {eee}\nAborting...")
    MPI.COMM_WORLD.Abort(1)

storage = {}
atoms.calc = ASI_ASE_calculator(ASI_LIB_PATH, init_aims, MPI.COMM_WORLD, atoms)
atoms.calc.asi.register_dm_callback(default_saving_callback, (atoms.calc.asi, storage) )

parprint(f'E = {atoms.get_potential_energy():.6f}')

DM = storage.get((1,1), None)
```

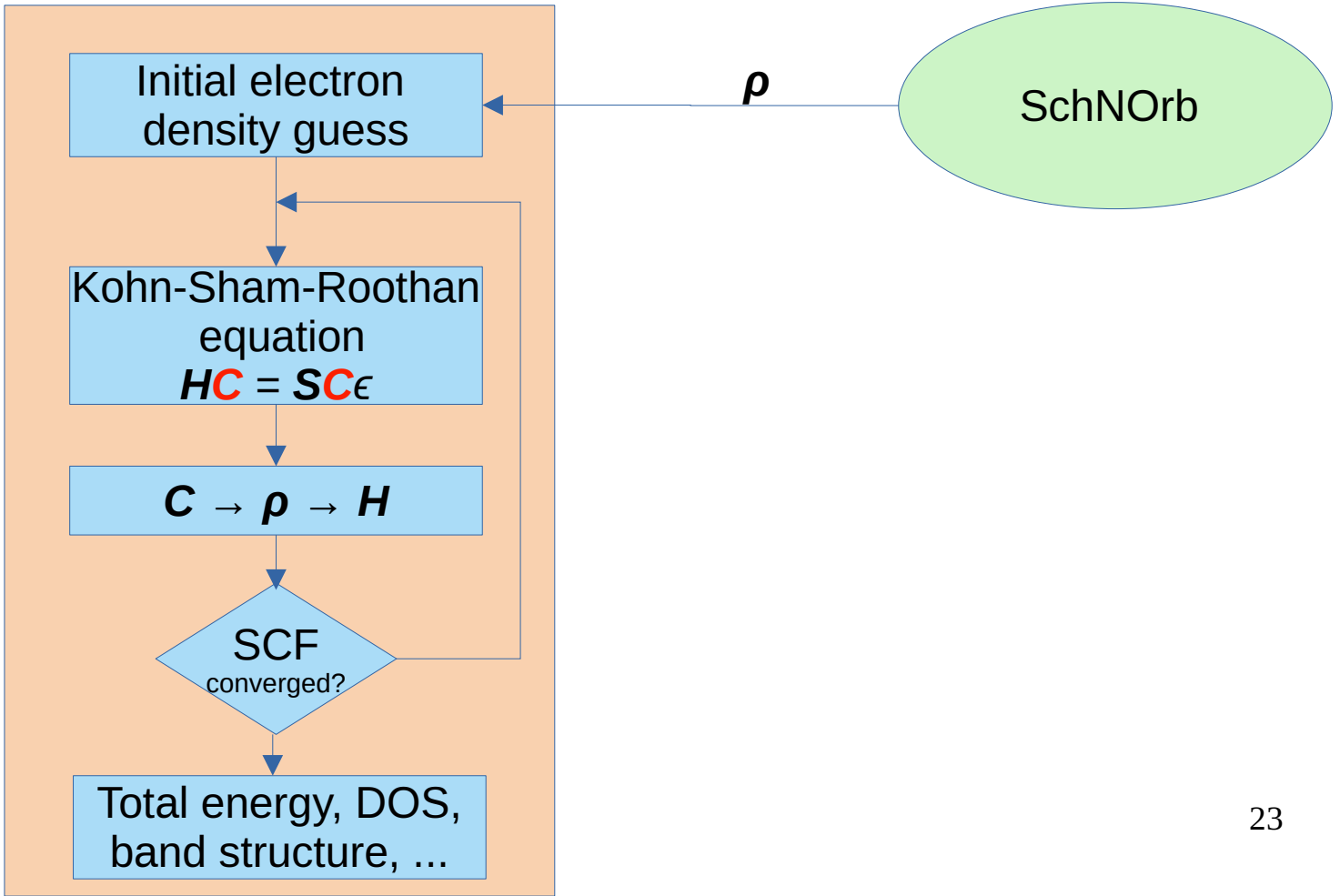
Matrices import/export

```
atoms.calc = ASI_ASE_calculator(ASI_LIB_PATH, init_aims, None, atoms)
atoms.calc.asi.keep_density_matrix = True
atoms.calc.asi.keep_hamiltonian = True
atoms.calc.asi.keep_overlap = True
atoms.calc.asi.init_density_matrix = {(1,1):predict_dm(atoms)}

parprint(f'E = {atoms.get_potential_energy():.6f}')

S = atoms.calc.asi.overlap_storage[(1,1)]
H = atoms.calc.asi.hamiltonian_storage[(1,1)]
DM = atoms.calc.asi.dm_storage.get((1,1), None)
S_cnt = atoms.calc.asi.overlap_calc_cnt[(1,1)]
H_cnt = atoms.calc.asi.hamiltonian_calc_cnt[(1,1)]
DM_cnt = atoms.calc.asi.dm_calc_cnt[(1,1)]
```

SCF acceleration via DM prediction



SchNOrb model for water

Unifying machine learning and quantum chemistry with a deep neural network for molecular wavefunctions

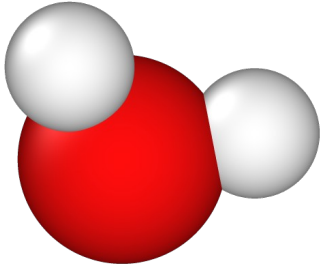
K. T. Schütt, M. Gastegger, A. Tkatchenko, K.-R. Müller & R. J. Maurer

Nature Communications **10**, Article number: 5024 (2019) | Cite this article

Julia Westermayr

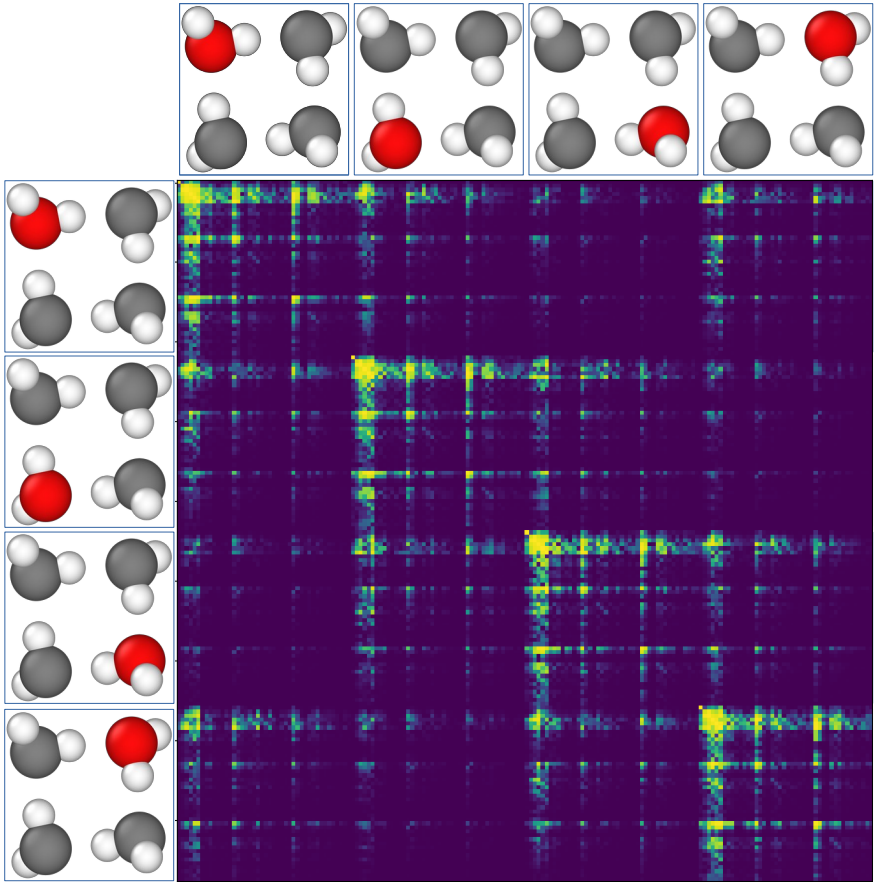
Reinhard Maurer

WARWICK
THE UNIVERSITY OF WARWICK



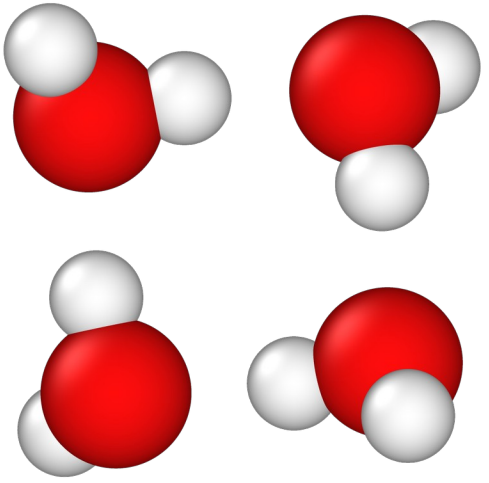
SchNOrb → ρ

Denstiy matrix in localized basis

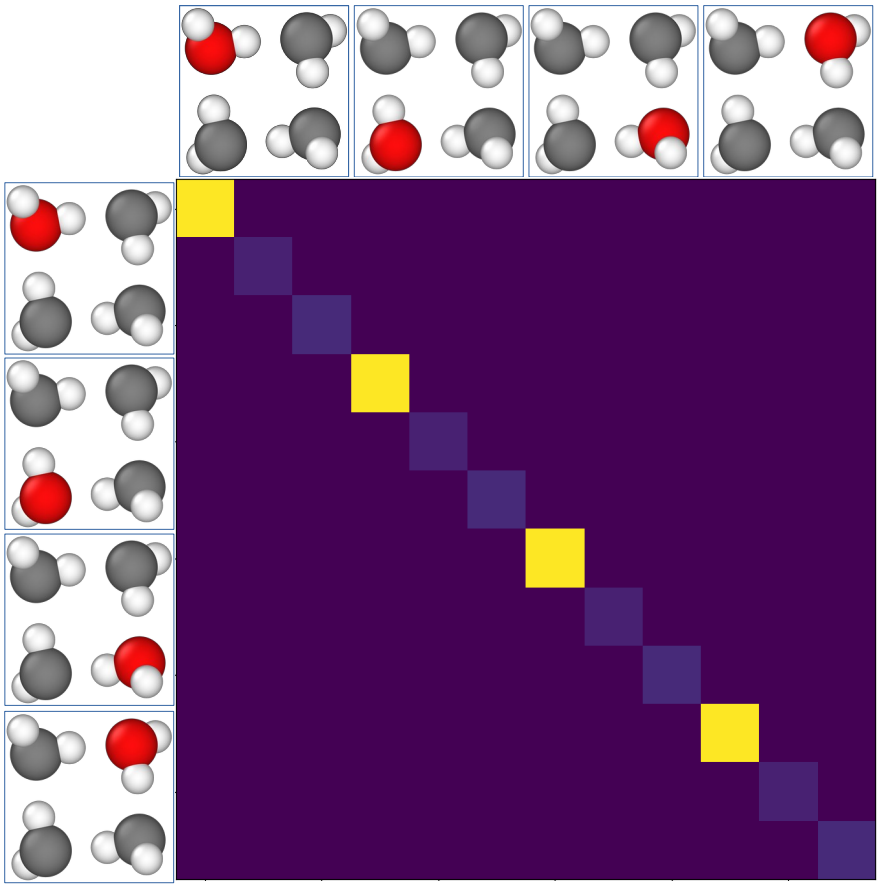


arctan(abs(DM)*100))

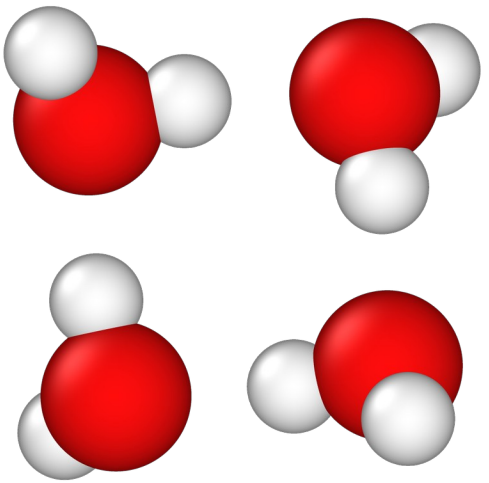
Each row/column corresponds to a basis function localized at some atom



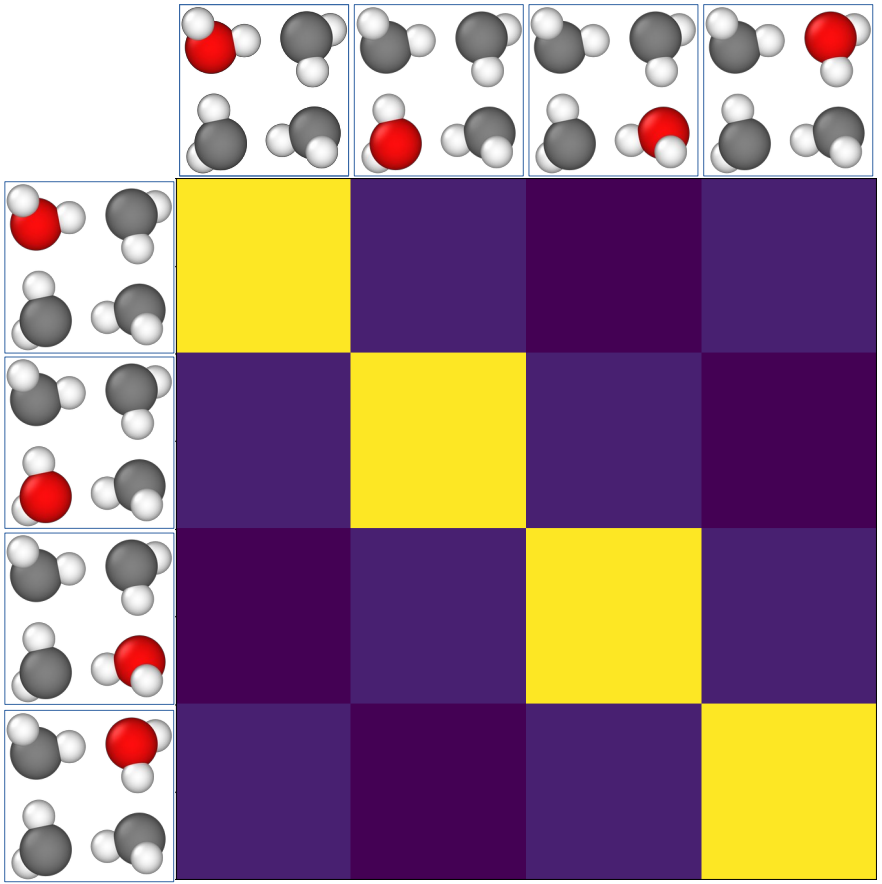
Free atom initialization



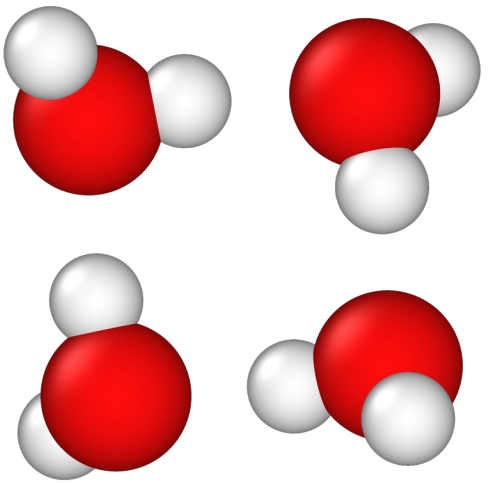
Each row/column corresponds to a basis function localized at some atom



Molecular initialization



Each row/column corresponds to a basis function localized at some atom



```
dm_mols =bsum.bsum(bsum.bsum(ar.data.dm**2, [44]*4, axis=0 ), [44]*4, axis=1)**0.5
plt.imshow(np.arctan(np.abs(dm_mols/44/44)*100)); plt.show()
```

SchNOrb model for water

Unifying machine learning and quantum chemistry with a deep neural network for molecular wavefunctions

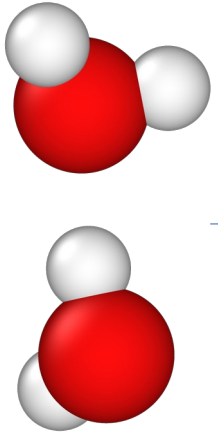
K. T. Schütt, M. Gastegger, A. Tkatchenko, K.-R. Müller & R. J. Maurer

Nature Communications **10**, Article number: 5024 (2019) | Cite this article

Julia Westermayr

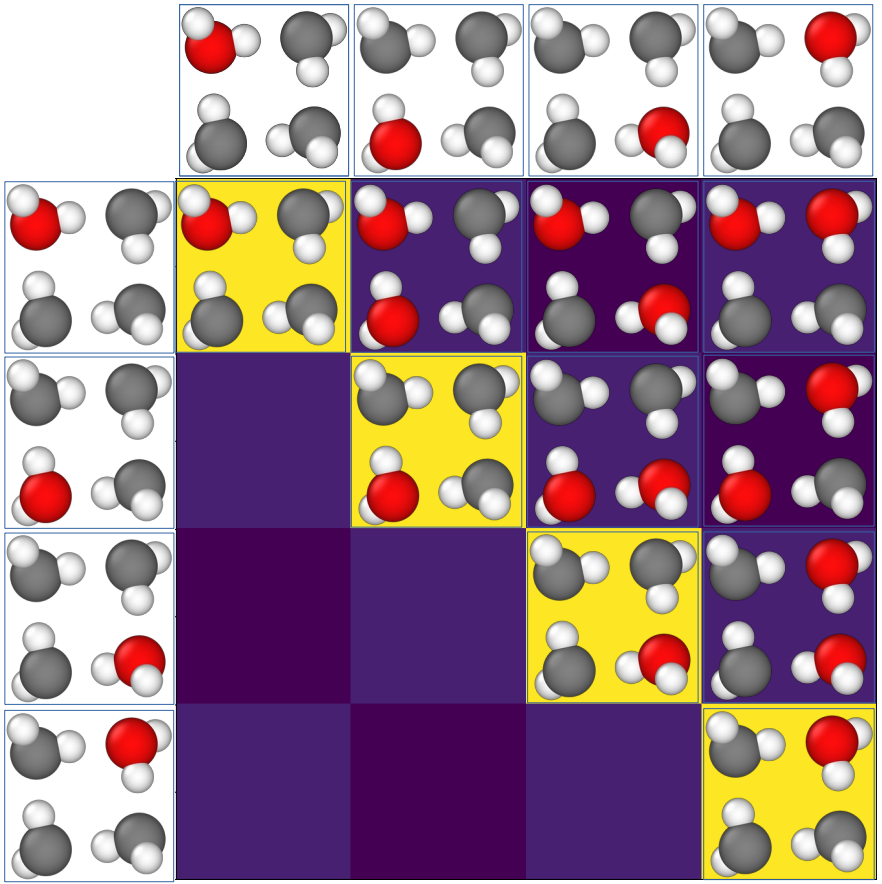
Reinhard Maurer

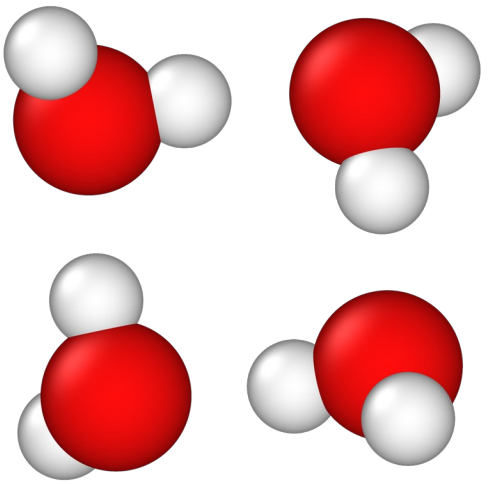
WARWICK
THE UNIVERSITY OF WARWICK



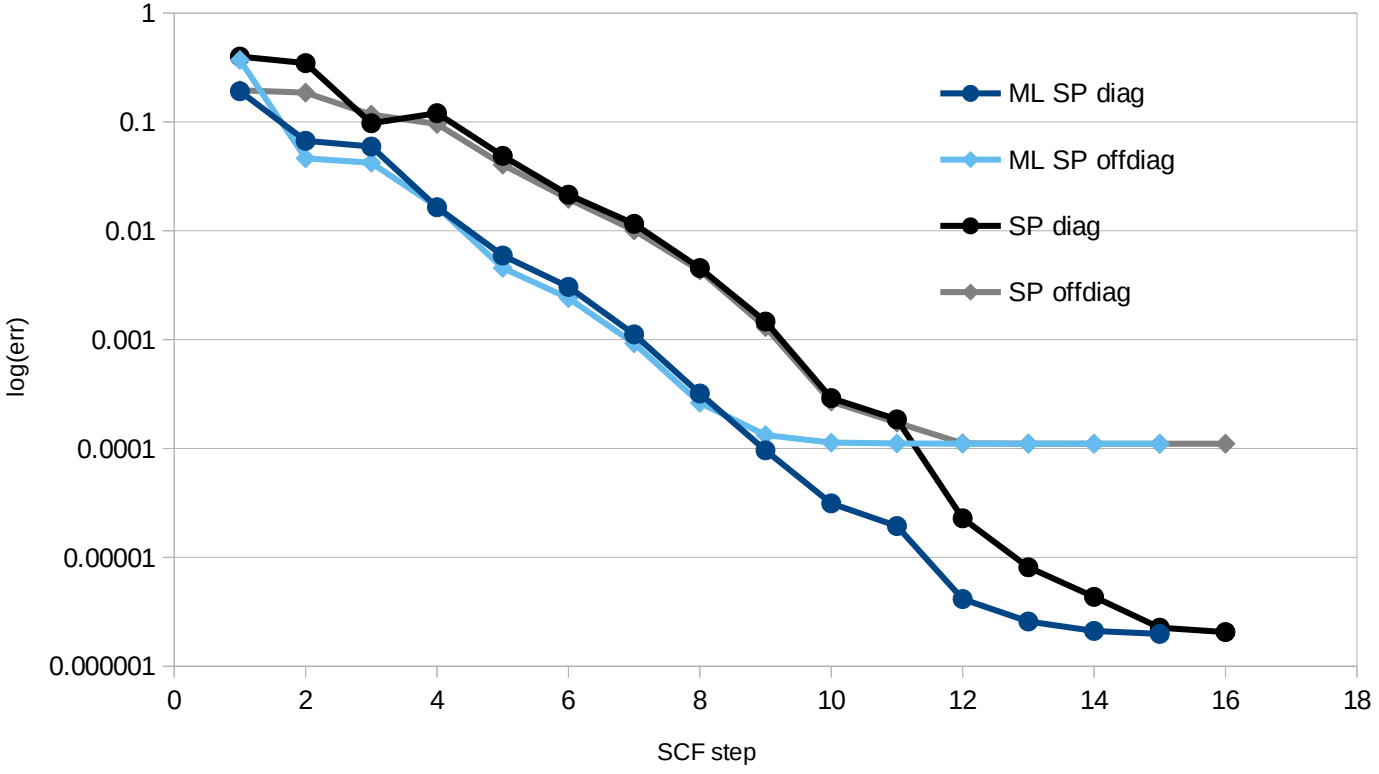
SchNOrb → ρ

DM stitching algorithm

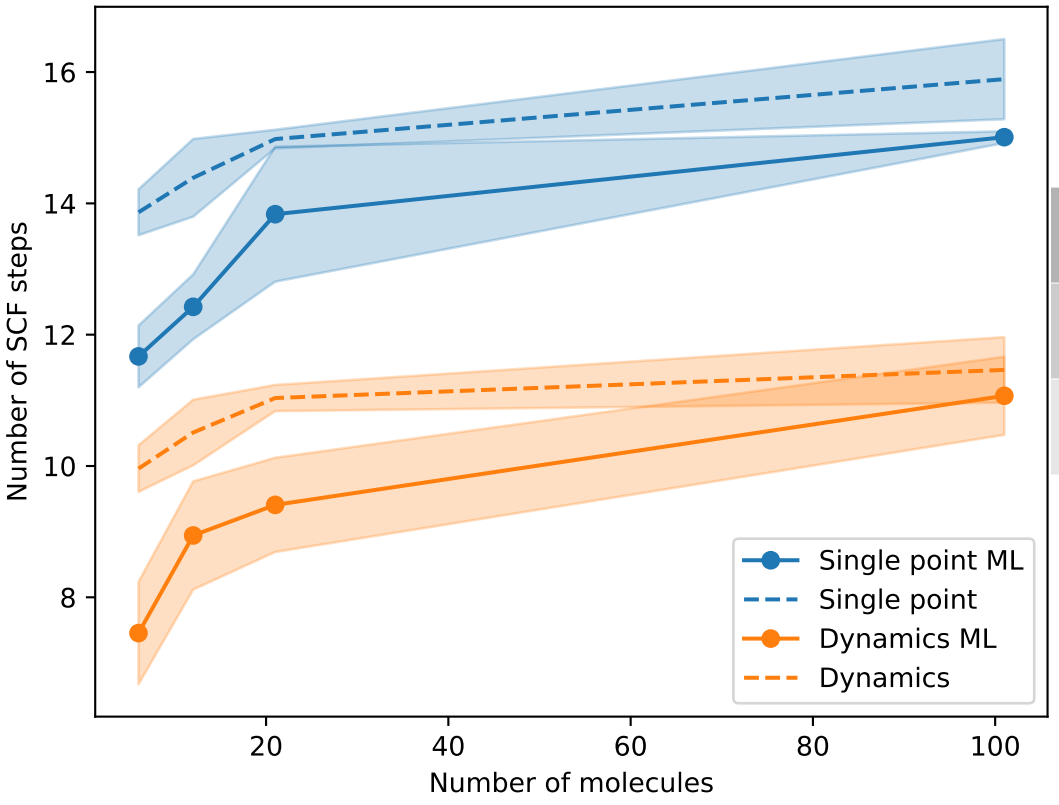




Errors over SCF loop



Achieved SCF acceleration



	6	12	21	101
Single point	15.83%	13.66%	7.64%	5.56%
Dynamics	25.15%	14.92%	14.74%	3.45%

ASI availability

- FHI-aims `master` branch
- JOSS paper: 10.21105/joss.05186
- ASI sources & tests: https://gitlab.com/pvst/asi
- Documentation: https://pvst.gitlab.io/asi/
- `asi4py` in `pip`: https://pypi.org/project/asi4py/

Acknowledgements

eCSE

FLF

SUPERCOMPUTING WALES

UWCHGYFRIFIADURA CYMRU

- Yi Yao
- Balint Aradi
- Tom Keal
- Mariana Rossi
- Julia Westermayr