# Exascale simulations via the submatrix matrix method

R. Schade, T. Kenter, M. Lass, C. Plessl & T. D. Kühne

Department of Chemistry
Dynamics of Condensed Matter
Paderborn Center for Parallel Computing

UNIVERSITÄT PADERBORN
*Die Universität der Informationsgesellschaft*

# Why Quantum Chemistry?

**AB INITIO QUANTUM CHEMISTRY:**
**A SOURCE OF IDEAS FOR LATTICE GAUGE THEORISTS**

**Kenneth G. WILSON**

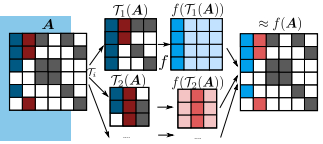
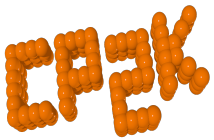
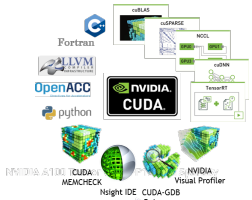
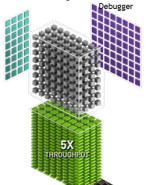
**The Ohio State University, Department of Physics, 174 W. 18th Avenue, Columbus, OH 43210 USA**

**Ab initio quantum chemistry is an emerging computational area that is fifty years ahead of lattice gauge theory, a principal competitor for supercomputer time, and a rich source of new ideas and new approaches to the computation of many fermion systems. An overview of the history, current prospects and future frontiers of quantum chemistry is given, with special emphasis on lessons for lattice gauge theory. Particular reference is given to the role of Gaussian basis functions (in place of grids) and analytic (as opposed to Monte Carlo) methods. The main recommendation to lattice gauge theorists is for greater emphasis on infinite momentum frame studies, using Gaussian basis functions.**

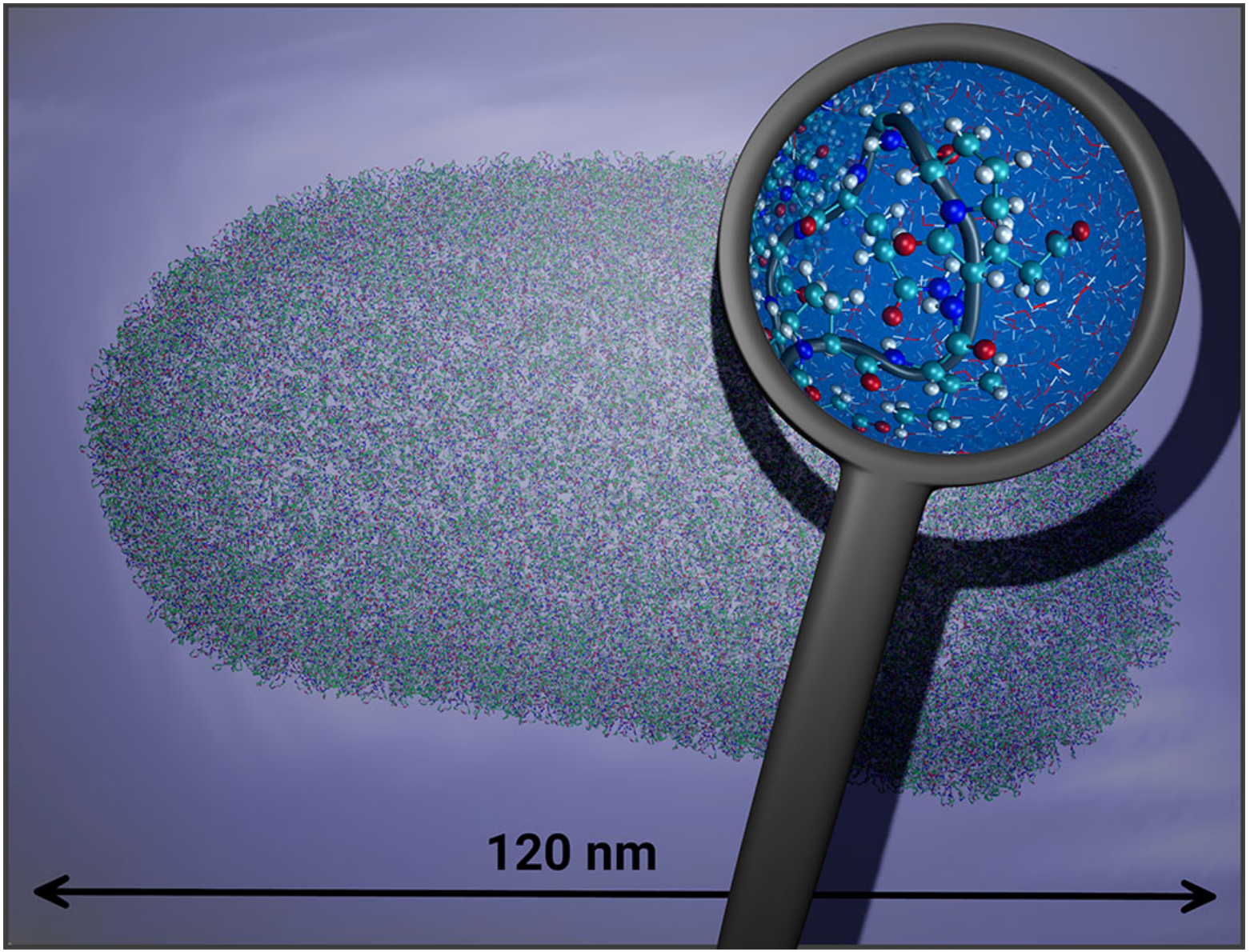
K. G. Wilson, Nucl. Phys. B **17**, 82 (1990)

# Mother of HPC Problems

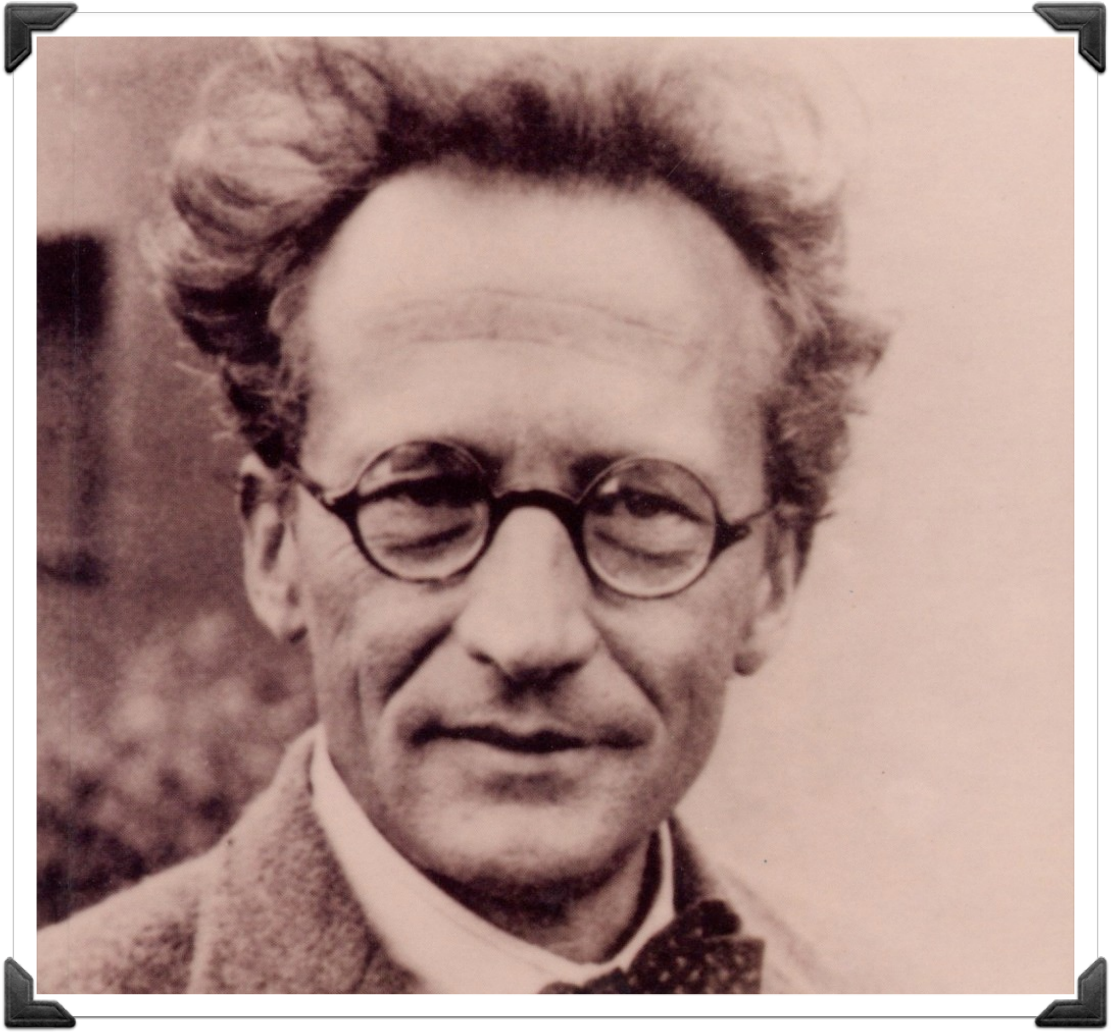
Combine all Levels of the Computer Technology Stack:

| | | |
|---|---|---|
| Algorithm |  | Non-orthogonal local submatrix (NOLSM) method<br>compensation of numerical noise |
| Application |  | CP2K: quantum chemistry and solid-state theory program |
| Libraries |  | NVIDIA cuBLAS with streams and CUDA graphs |
| Computer Architecture |  | Third generation NVIDIA tensor cores<br>with mixed FP16/FP32 support |
| Semiconductor Technology |  | NVIDIA A100 Tensor-core GPU (TSMC 7 nm FinFET) |

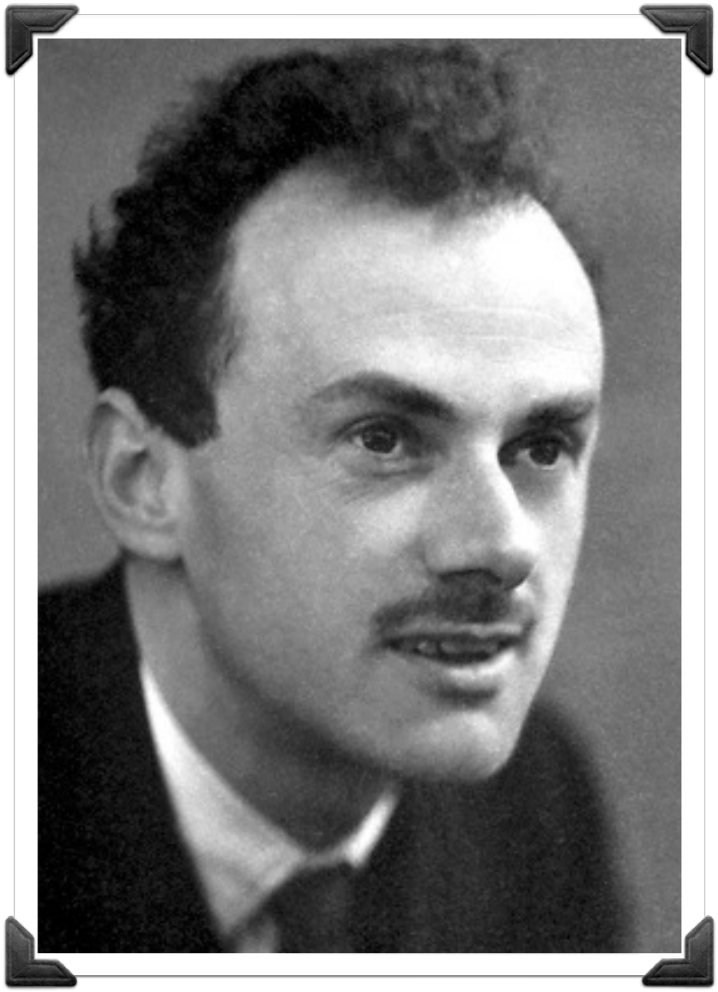
# Computational Microscope



# Schrödinger Equation



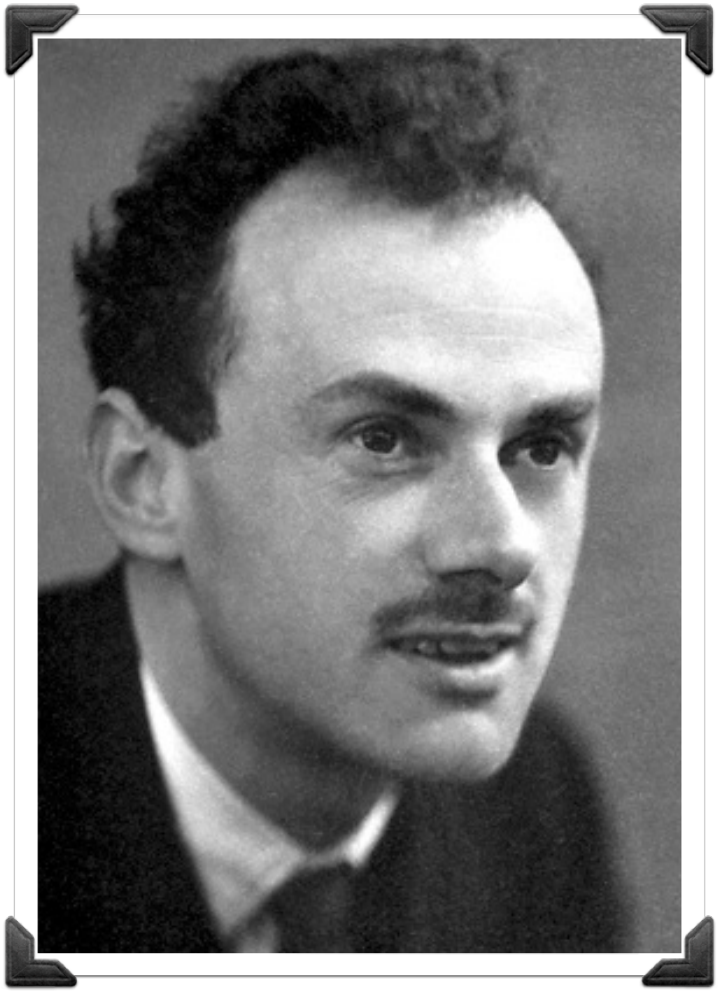
# Schrödinger Equation



„The fundamental laws necessary for the mathematical treatment of large parts of physics and **the whole of chemistry are thus fully known** ...“

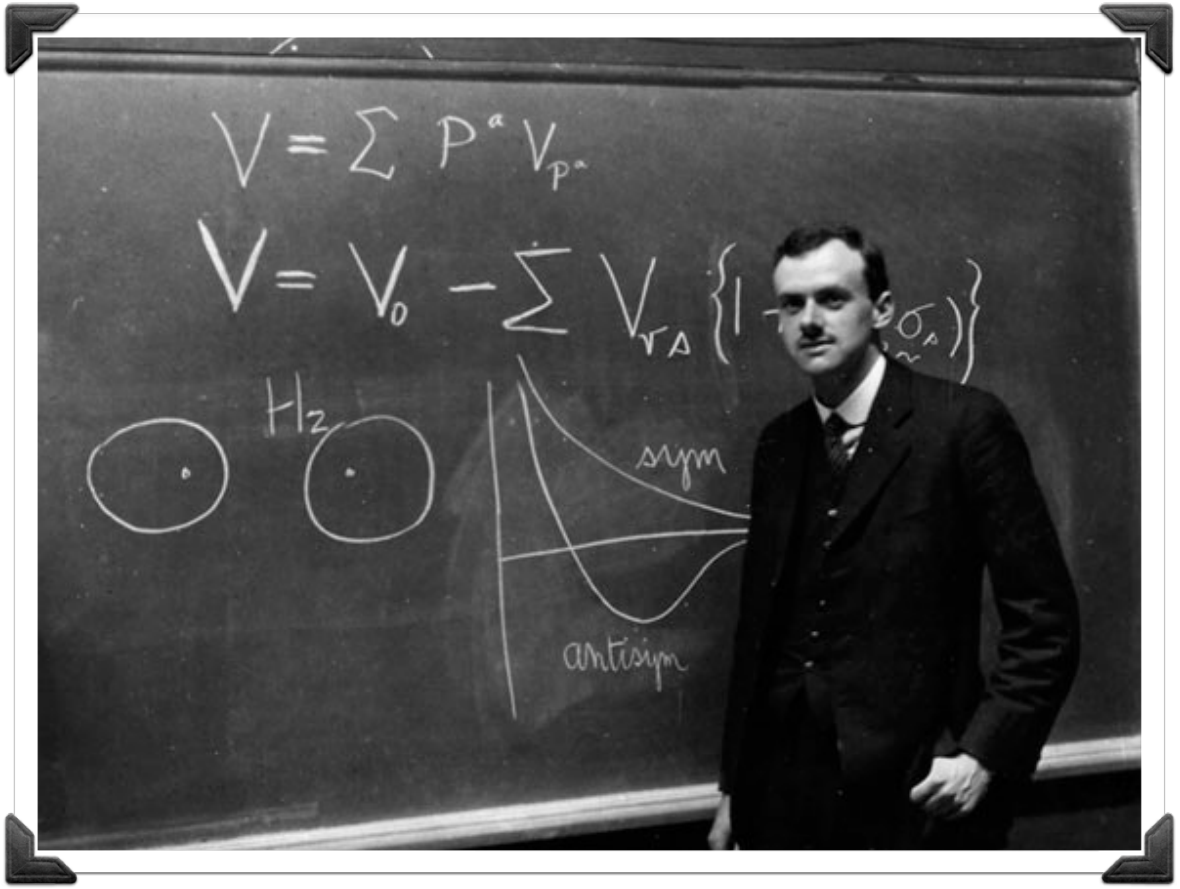
$$\mathcal{H}(\mathbf{r},\mathbf{R})\Psi(\mathbf{r},\mathbf{R}) = E\Psi(\mathbf{r},\mathbf{R})$$

# Schrödinger Equation



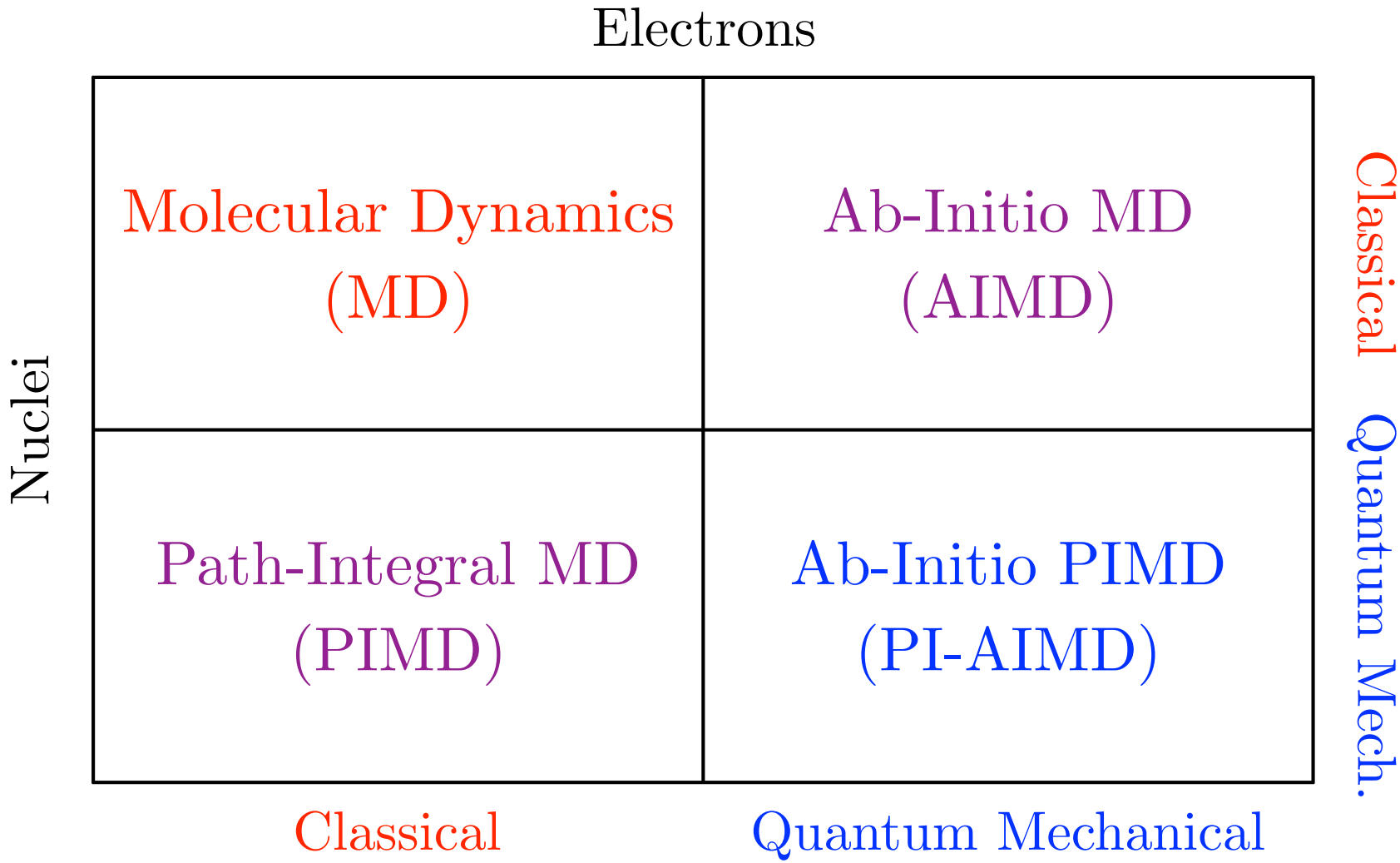
„... the difficulty lies only in the fact that application of these laws leads to equations that are too complex to be solved.“

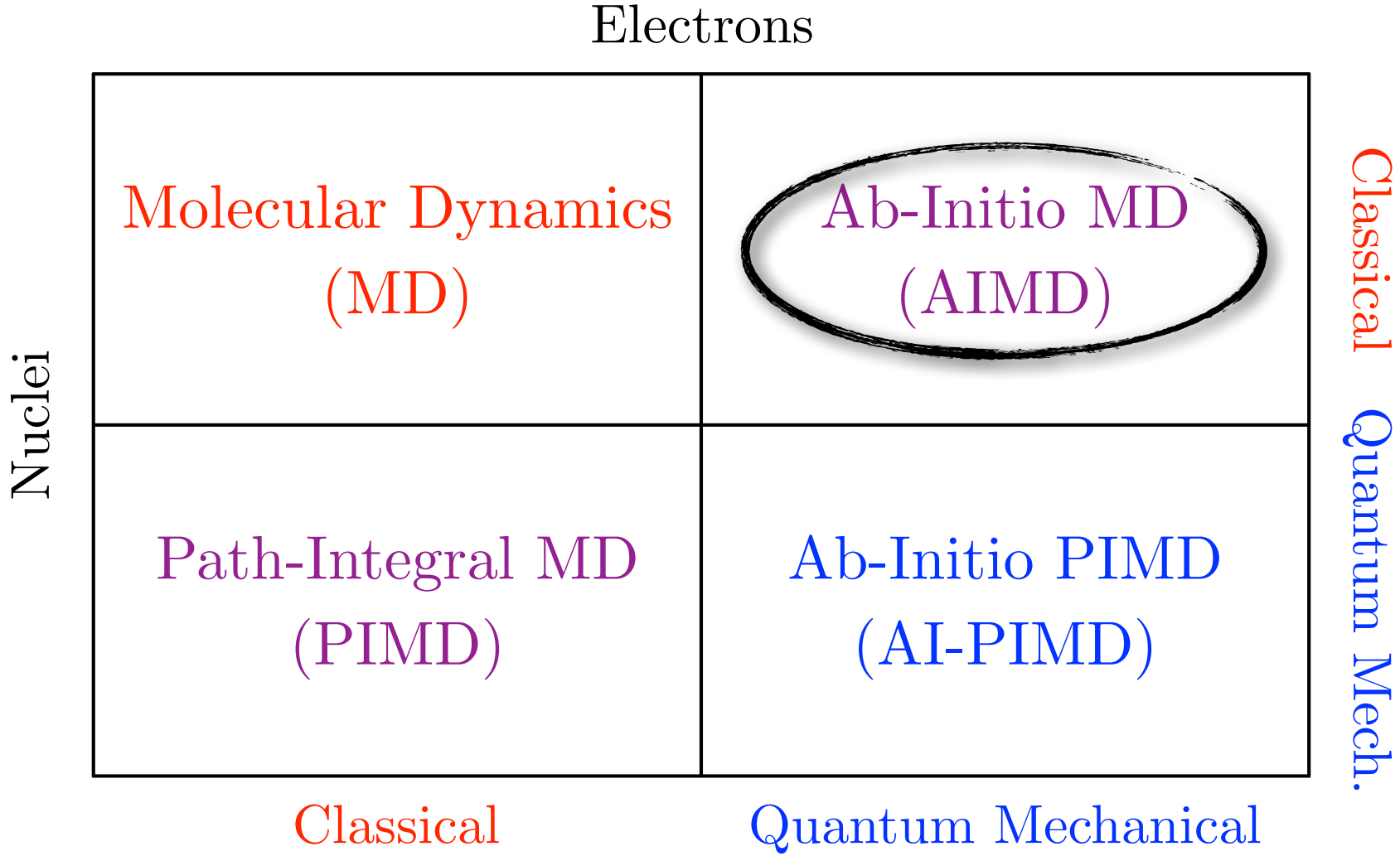
# Schrödinger Equation




„... hence it would be desirable to develop practical approximation schemes for the application of quantum mechanics“

# Born-Oppenheimer






$$\mathcal{H}_e(\boldsymbol{r};\boldsymbol{R})\psi(\boldsymbol{r};\boldsymbol{R}) = \varepsilon(\boldsymbol{R})\psi(\boldsymbol{r};\boldsymbol{R})$$

$$M_I\ddot{\boldsymbol{R}}_I = -\nabla_{\boldsymbol{R}_I}\left[\varepsilon(\boldsymbol{R}) + V_{KK}(\boldsymbol{R})\right]$$

# CP2K: Overview

T. D. Kühne et al., JCP **152**, 194103 (2020)

- Static Calculations
  - Energy & Structure Optimization
  - Transition Paths (String, NEB)
  - Properties: NMR, EPR & XAS
- Sampling Techniques
  - MC, MD & Path-Integral Methods
  - RT-TDDFT/Ehrenfest Dynamics
  - Accelerated FES: Metadynamics
- Energy & Force Methods
  - All-Electron Calculations (GAPW)
  - *Quickstep*: PP Calculations (GPW)
  - Post-HF Methods (RPA, MP2, GW)
  - DFT/HF Methods (HFX, CDFT)
  - Behler-type NN Potentials
  - Semiempirical QC & TB Methods
  - Classical Molecular Mechanics
  - Embedding Methods (IS, QM/MM)

# CP2K: Overview


http://www.cp2k.org

- Static Calculations
  - Energy & Structure Optimization
  - Transition Paths (String, NEB)
  - Properties: NMR, EPR & XAS
- Sampling Techniques
  - MC, MD & Path-Integral Methods
  - RT-TDDFT/Ehrenfest Dynamics
  - Accelerated FES: Metadynamics
- Energy & Force Methods
  - All-Electron Calculations (GAPW)
  - *Quickstep*: PP Calculations (GPW)
  - Post-HF Methods (RPA, MP2, GW)
  - DFT/HF Methods (HFX, CDFT)
  - Behler-type NN Potentials
  - Semiempirical QC & TB Methods
  - Classical Molecular Mechanics
  - Embedding Methods (IS, QM/MM)

# CP2K: Recent Development


http://www.cp2k.org

- Large Scale Computational Kernels
  - PW-based DFT (SIRIUS)
  - Hybrid DFT, MP2 & RPA
  - **Linear Scaling Algorithms**
- Approximate Molecular Dynamics
  - 2nd Generation Car-Parrinello MD
  - **Approximate Computing**
- Massive Parallelism
  - **Mixed MPI / OpenMP**
  - **GPU / FPGA support**
- **Sparse Matrix Algebra**
  - **Distributed Block CSR**
  - **Cannon's Algorithm & SMM**
- Open Source
  - 1.5 mio. lines of code
  - $>$ 6500 regression tests

# Schrödinger Equation


$$\mathbf{H}[\mathbf{C}]\mathbf{C} = \varepsilon\mathbf{S}\mathbf{C}$$

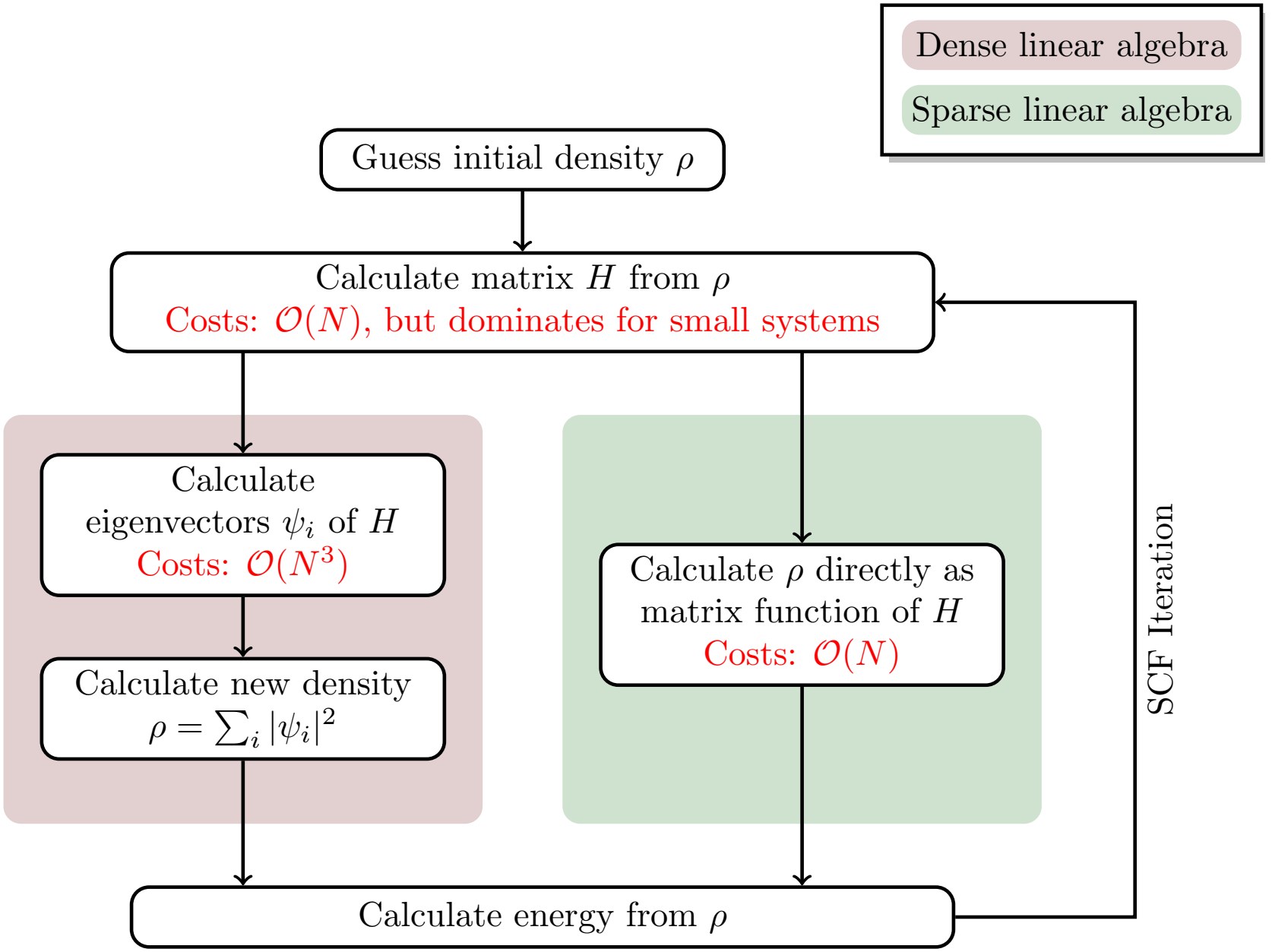
$$\Big\downarrow \mathcal{O}(N^3)$$

$$E = \sum_{i=1}^{N_e} \varepsilon_i - V_{dc} = Tr\left[\mathbf{C}^T\mathbf{H}\mathbf{C}\right] - V_{dc}$$

$$= \min_{\mathbf{P}} Tr\left[\mathbf{P}\mathbf{H}\right]\Big|_{\mathbf{PSP}=\mathbf{P}} - V_{dc}$$

D. Richters and T. D. Kühne, J. Chem. Phys. **140**, 134109 (2014)

# Self-Consistent Field



# The Sign Method

$$\mathbf{P} = \frac{1}{2}\left(\mathbf{I} - \operatorname{sign}\left(\mathbf{S}^{-1}\mathbf{H} - \mu\mathbf{I}\right)\right)\mathbf{S}^{-1}$$

- matrix sign function and inversion can be evaluated iteratively

$$\operatorname{sign}(x) = \frac{x}{|x|} = \frac{x}{\sqrt{x^2}}$$

$$\operatorname{sign}(\boldsymbol{A}) = \boldsymbol{A} \cdot (\boldsymbol{A}^2)^{-1/2}$$

$$\boldsymbol{X}_0 = \boldsymbol{A}$$

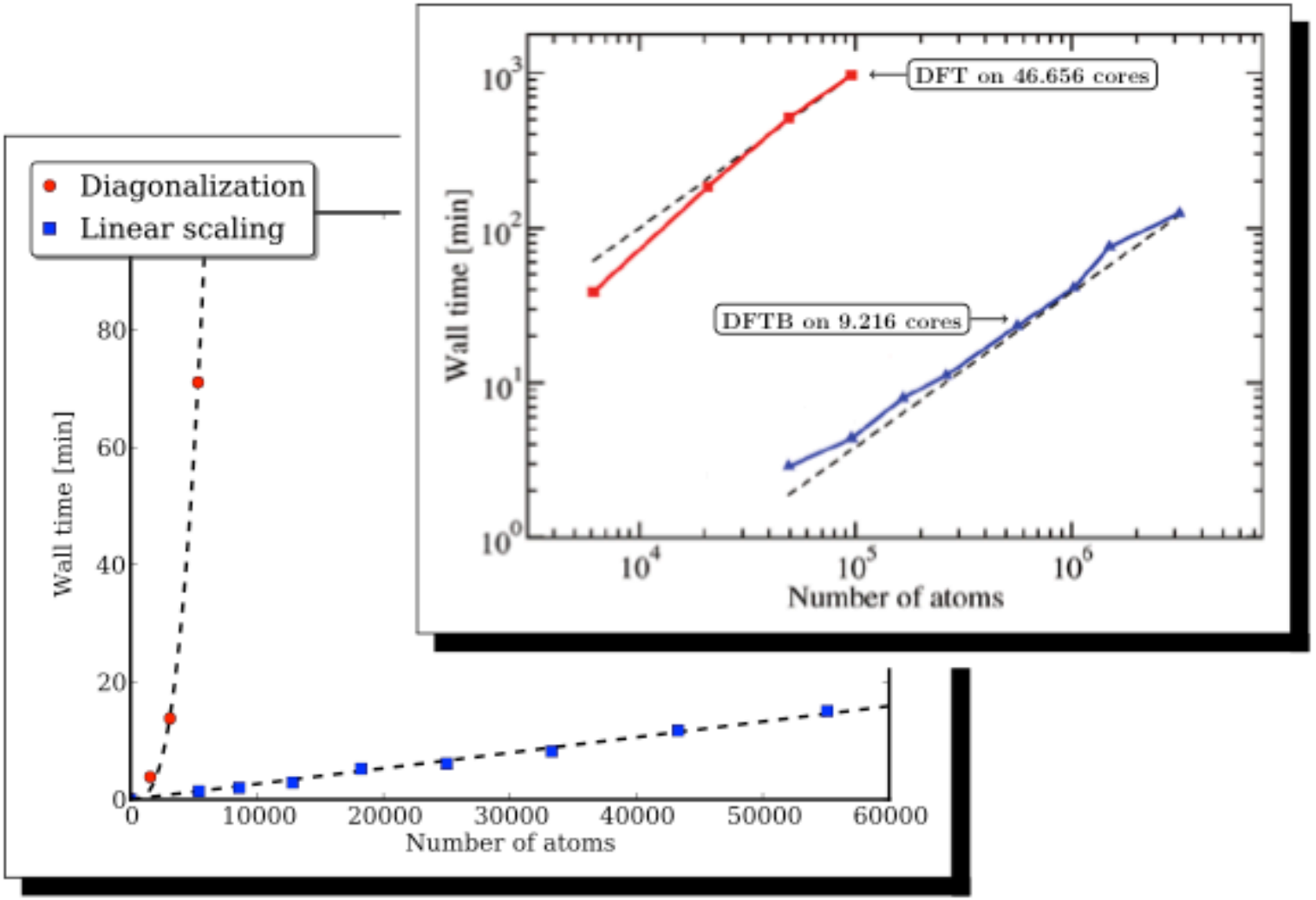
$$\boldsymbol{X}_{i+1} = \frac{1}{2}\boldsymbol{X}_i \cdot (3\boldsymbol{I} - \boldsymbol{X}_i^2)$$

$$\operatorname{sign}(\boldsymbol{A}) = \lim_{i\to\infty} \boldsymbol{X}_i$$

- linear-scaling approach $\mathcal{O}(N)$
- only **multiplications** of distributed sparse matrices are required
- but usually bound by inter-node communication!

M. Lass, S. Mohr, H. Wiebeler, TDK & C. Plessl, ACM Proc. of PASC **7**, 1 (2018)

# The Sign Method




$\mathcal{O}(N)$ methods
are inevitable
for large systems

M. Lass, S. Mohr, H. Wiebeler, TDK & C. Plessl, ACM Proc. of PASC **7**, 1 (2018)

# The DBCSR Library

- Distributed

  MPI parallelization based on Cannon, 2.5D, Carma, or Cosma algorithm

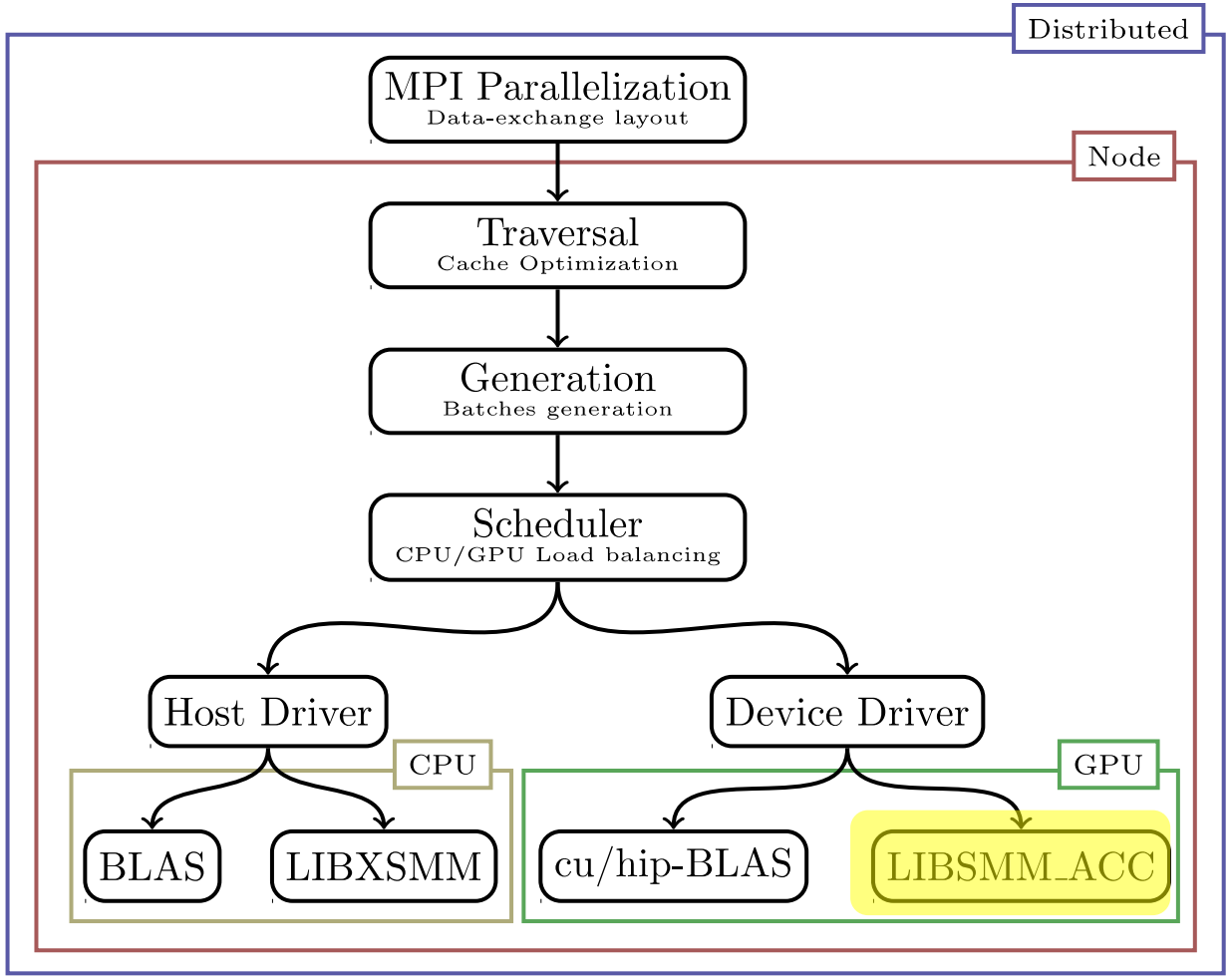
  On node parallelization via OpenMP
- Block Compressed Sparse Row

  Block-sparse, where block corresponds to atoms
- Small matrix-matrix multiplication library on multicore CPUs & GPUs

  libxsmm, libcusmm, libsmm_acc


T. D. Kühne et al., J. Chem. Phys. **152**, 194103 (2021)

# The DBCSR Library

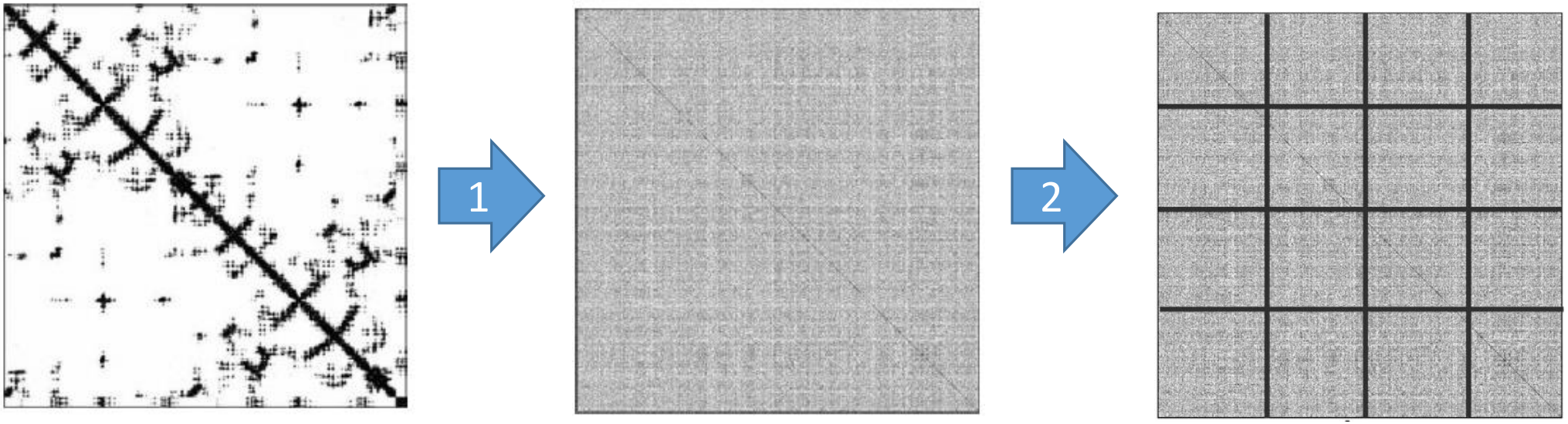



http://dbcsr.cp2k.org

T. D. Kühne et al., J. Chem. Phys. **152**, 194103 (2021)

# The DBCSR Library

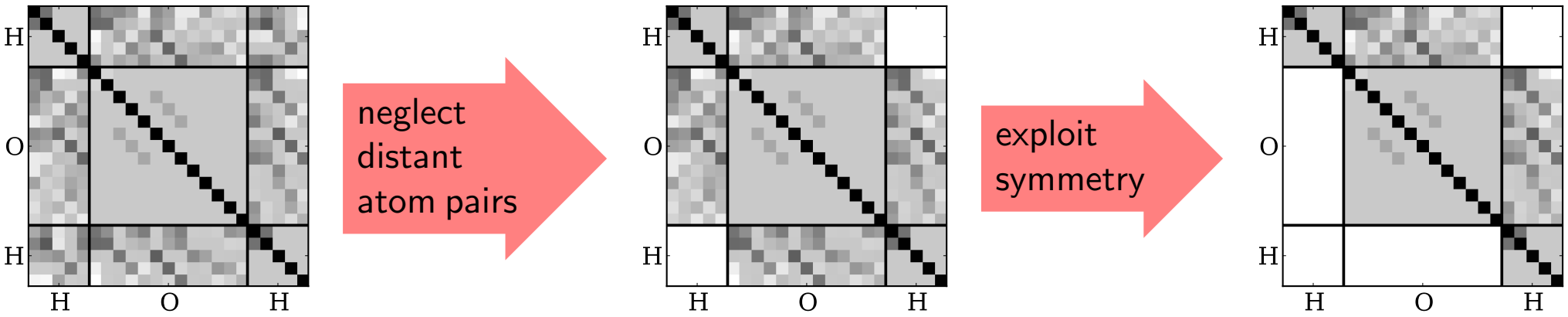
- 1. Random permutation of row and column block indices to balance load
  Each processor is approximately holding the same amount of data, with roughly the same amount of Flops
- 2. 2D grid decomposition over P processors
  Block-sparse, where block corresponds to atoms




T. D. Kühne et al., J. Chem. Phys. **152**, 194103 (2021)

# The DBCSR Library

- DBCSR is based on blocked structure
  - Non-zero elements are small dense blocks, typically 5x5, 13x13, 23x23, ...
  - Take full advantage of the block structured sparse nature of the matrices
  - Each block corresponds to the interaction between two atoms
- Dense limit is as important as the sparse limit
- Provide good scalability for a large number of processors



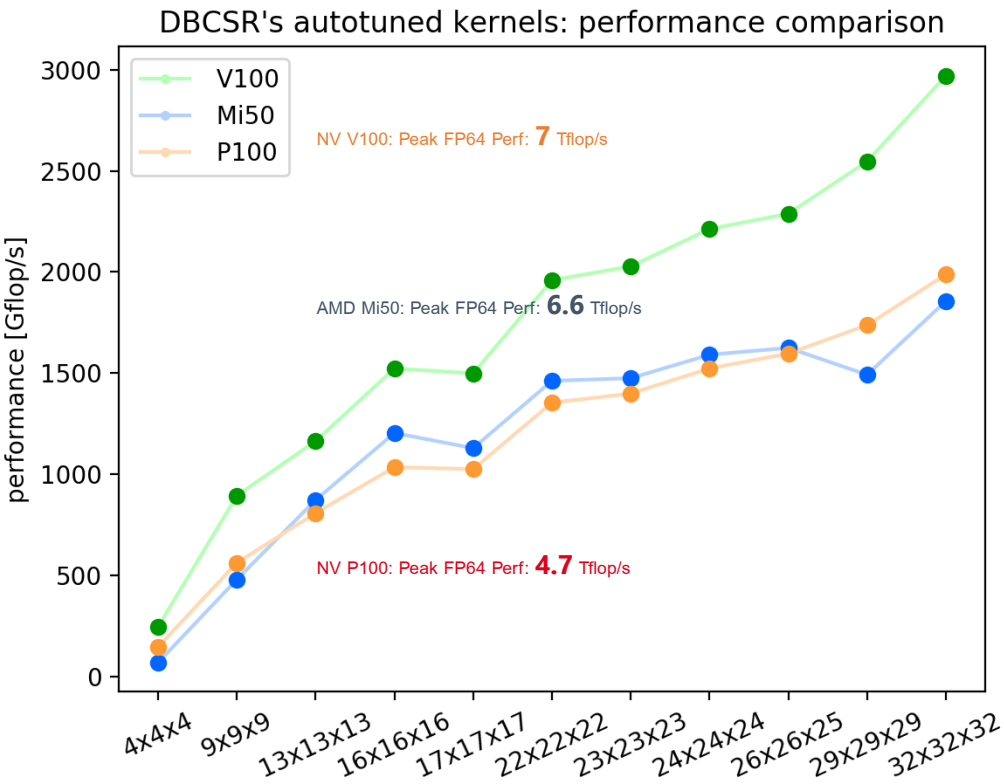

T. D. Kühne et al., J. Chem. Phys. **152**, 194103 (2021)

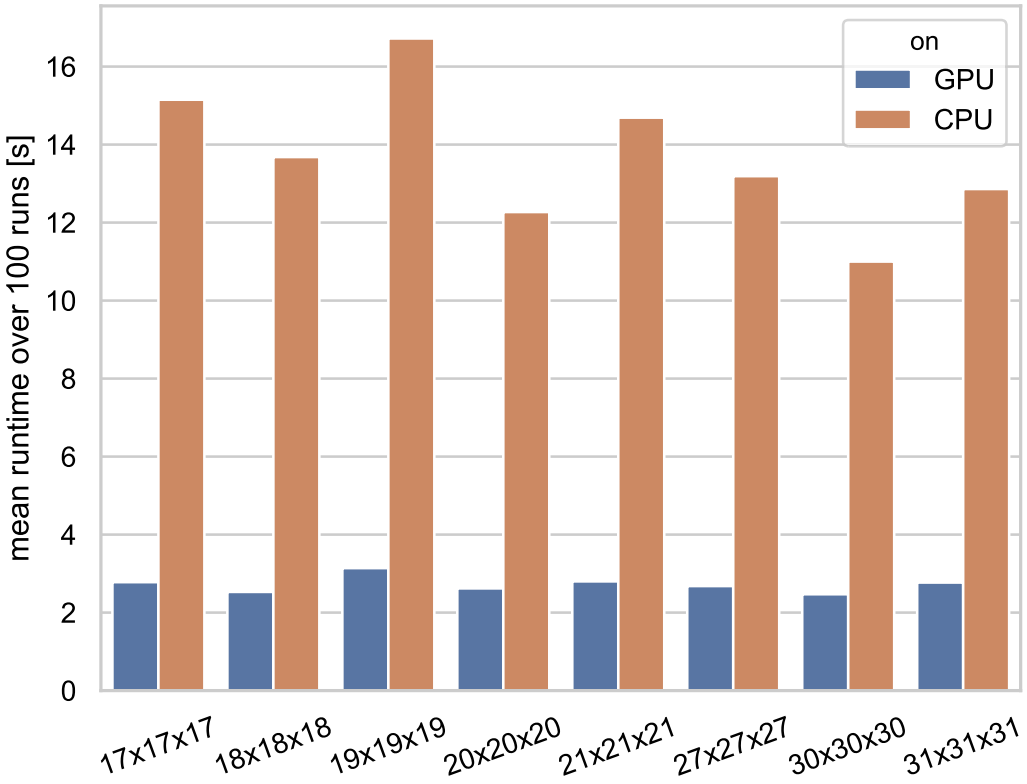
# SMM Libraries

- Optimized libraries were developed that outperform vendor BLAS for SMM
  - LIBXSMM for Intel-based CPU/KNL systems
  - LIBCUSMM for Nvidia GPUs using CUDA
  - LIBSMM_ACC for Nvidia/AMD GPUs using CUDA and HIP
- LIBXSMM generates executable code just-in-time (JIT) by assembling the instructions in-memory
  - All flavors of AVX extensions are supported
  - Avg. speed-up of approx. 3 for LIBXSMM over MKL-DGEMM on KNL
- LIBCUSMM employs a double-buffering technique, based on CUDA streams, to maximize occupancy of the GPU and hide data transfer latency
- LIBSMM_ACC GPU kernels are JIT compiled at runtime
- LICUSMM & LIBSMM_ACC includes an auto-tuning framework to find the best parameters for every (m,n,k)-kernel out of $>$ 100k combinations

T. D. Kühne et al., J. Chem. Phys. **152**, 194103 (2021)

# SMM Libraries




Additional factor 5 speed-up, when using LIBSMM_ACC instead of LIBXSMM on CRAY XC50 node with 12 core Intel Haswell CPU and Nvidia V100 GPU

T. D. Kühne et al., J. Chem. Phys. **152**, 194103 (2021)

# FPGA-based Noctua@PC2

UNIVERSITÄT PADERBORN
Die Universität der Informationsgesellschaft

NEUBAU
Rechenzentrum
NOCTUA

Paderborn Center for Parallel Computing

BLB NRW

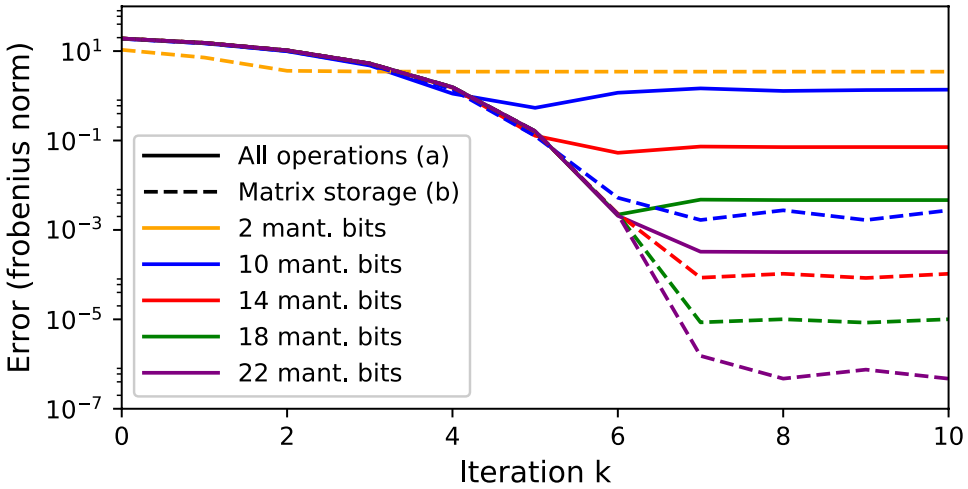
dc_ce
rz-beratung

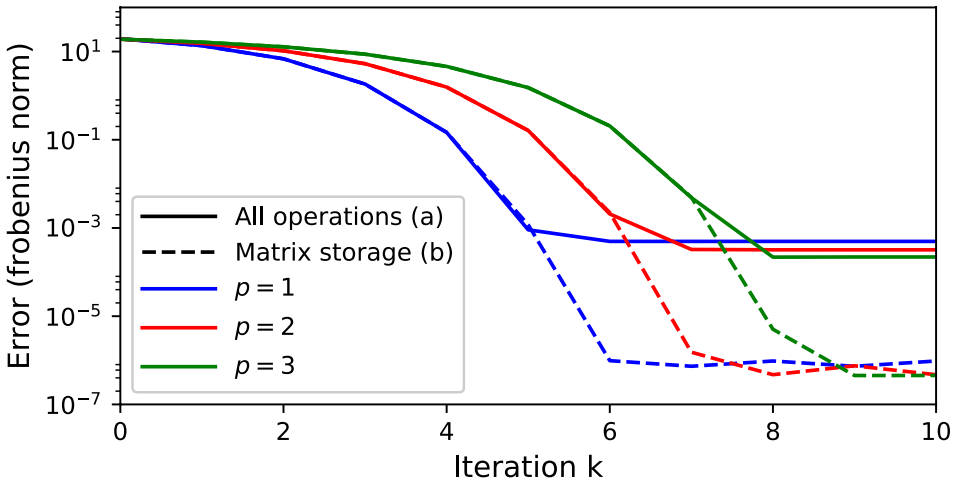
# FPGA-based Inversion

$$B_{k+1} = \frac{1}{p}\left[(p-1)B_k - \left((I - B_k^p A)^q - I\right) B_k^{1-p} A^{-1}\right]$$

$$\overset{p=1}{=} \left[I - (I - B_k A)^q\right] A^{-1}.$$

D. Richters, M. Lass, A. Walter, C. Plessl & T. D. Kühne, Comm. Comp. Phys. **25**, 564 (2019)



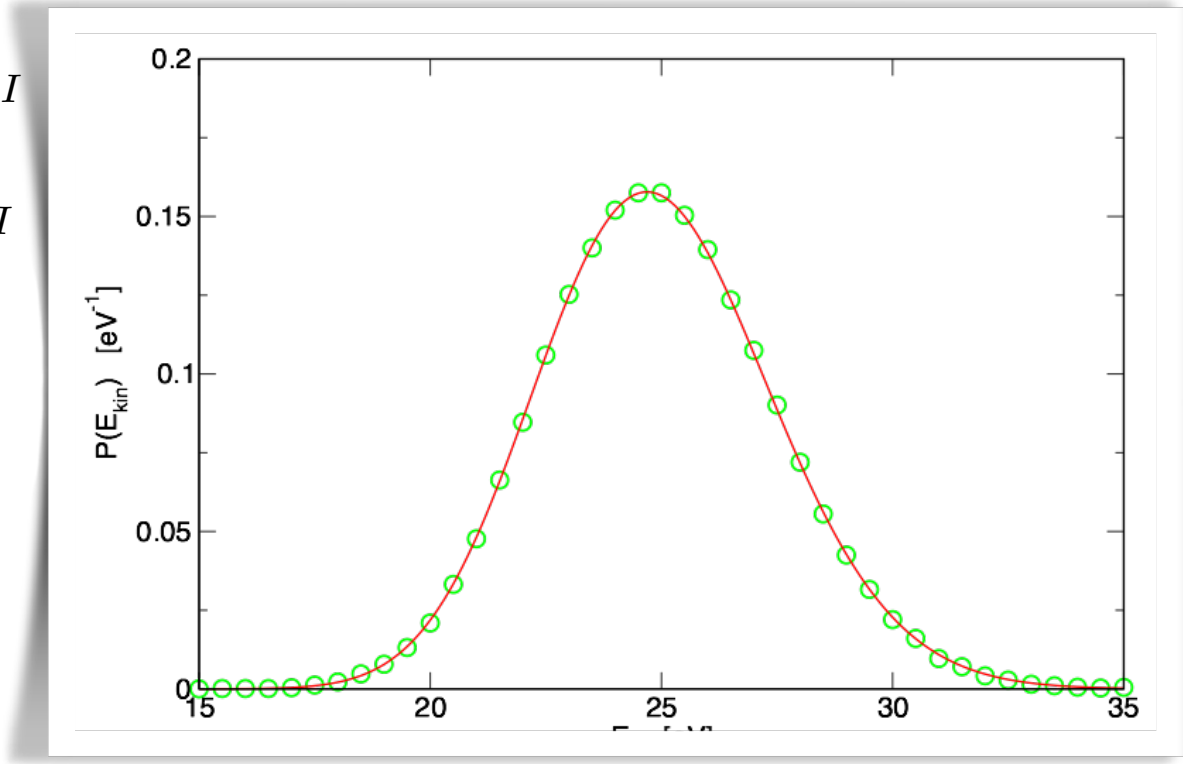

M. Lass, T. D. Kühne and C. Plessl, IEEE Embedded Systems Letters **PP**, 1 (2017)

# Approximate Computing

$$M_I \ddot{\mathbf{R}}_I = \underbrace{\mathbf{F}_I^{\mathrm{BO}} + \boldsymbol{\Xi}_I^N}_{\mathbf{F}_I^{\mathrm{FPGA}}} - \gamma_N M_I \dot{\mathbf{R}}_I$$

$$\left\langle \boldsymbol{\Xi}_I^D(0)\, \boldsymbol{\Xi}_I^D(t) \right\rangle = 2\gamma_D M_I k_B T \delta(t)$$

$$\left\langle \tfrac{1}{2} M_I \dot{\boldsymbol{R}}_I^2 \right\rangle = \tfrac{3}{2} k_B T$$

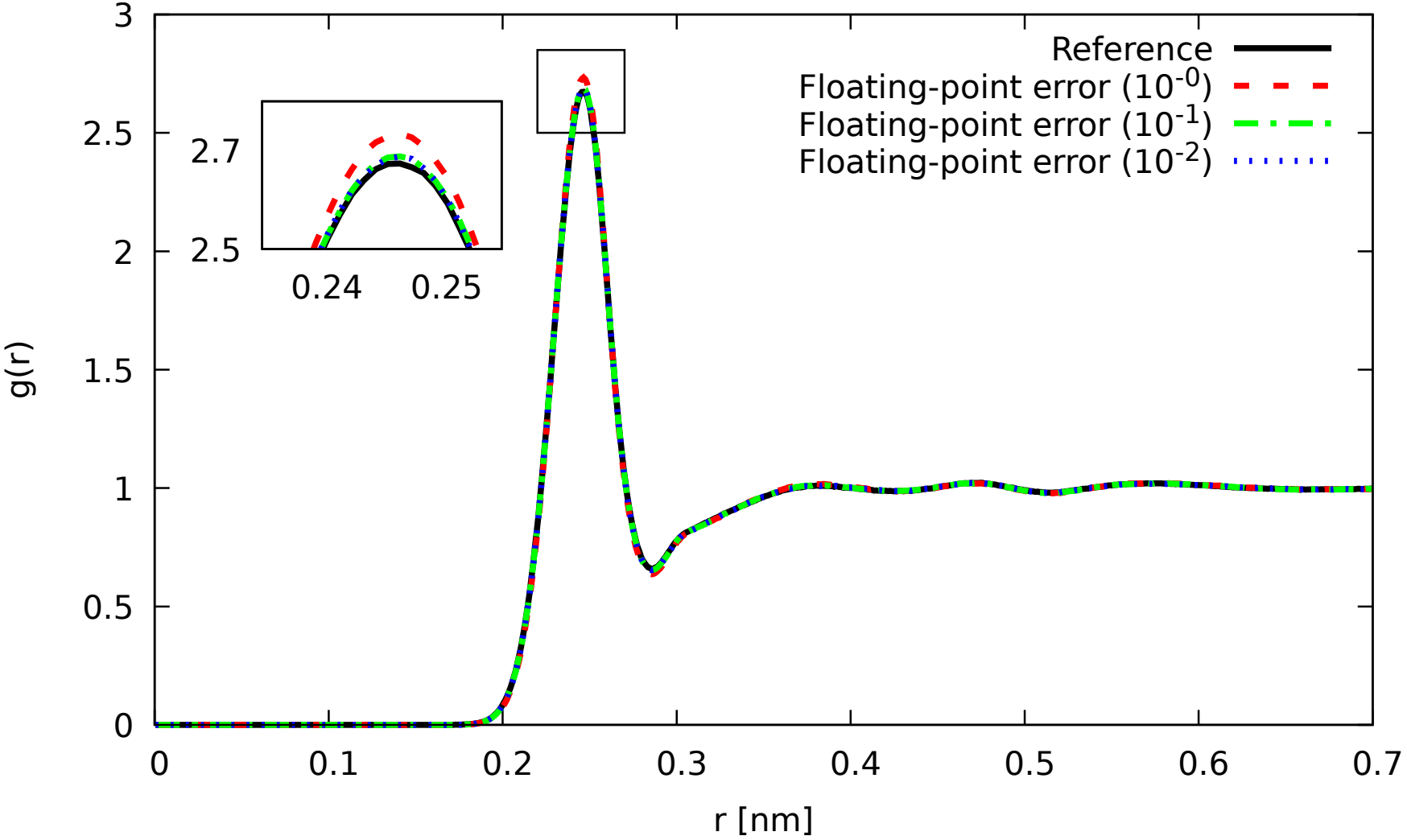




T. D. Kühne, F. R. Mohamed, M. Krack & M. Parrinello, Phys. Rev. Lett. **98**, 066401 (2007)

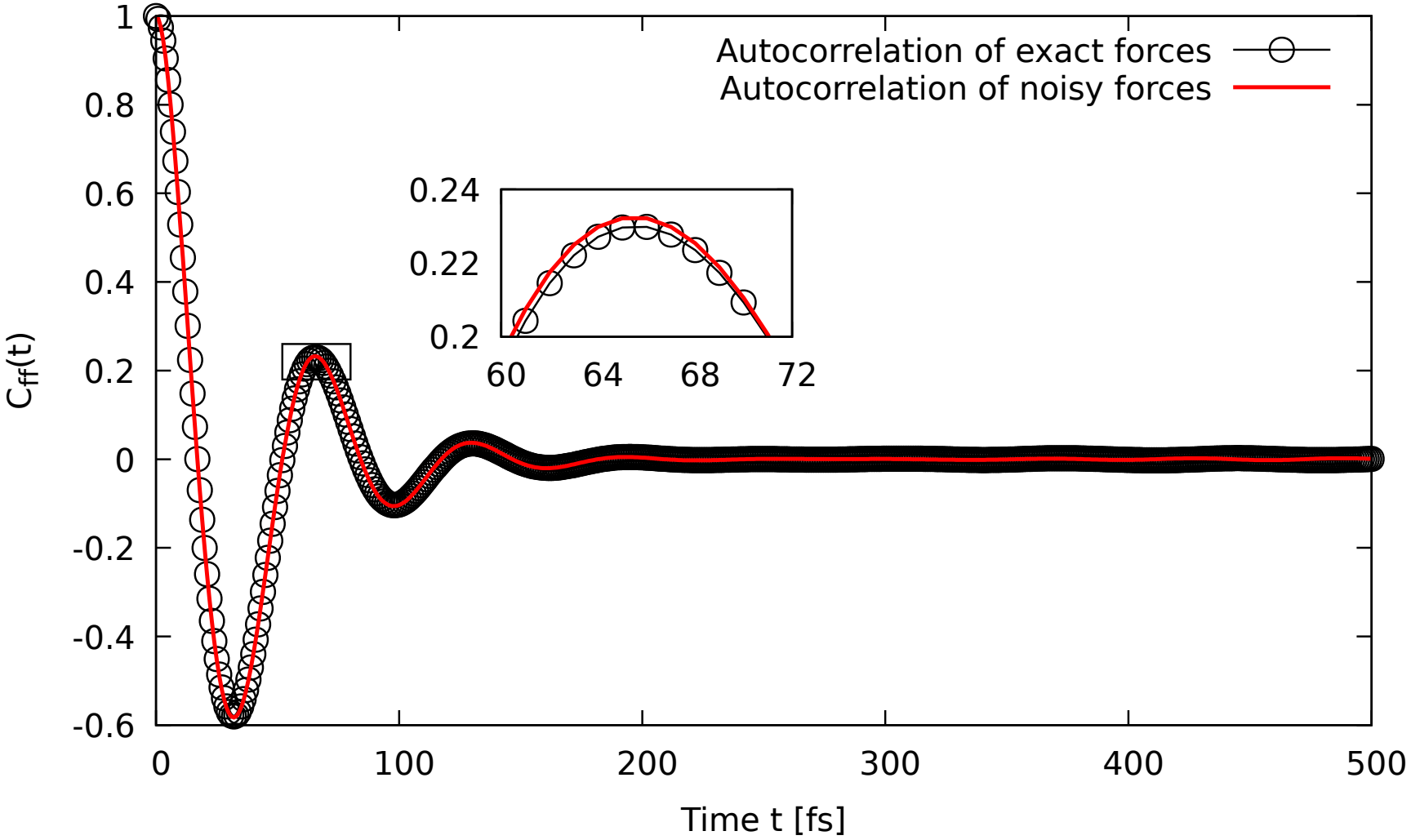
V. Rengaraj M. Lass, C. Plessl and T. D. Kühne, Computation **8**, 39 (2020)

# Approximate Computing




V. Rengaraj M. Lass, C. Plessl and T. D. Kühne, Computation **8**, 39 (2020)

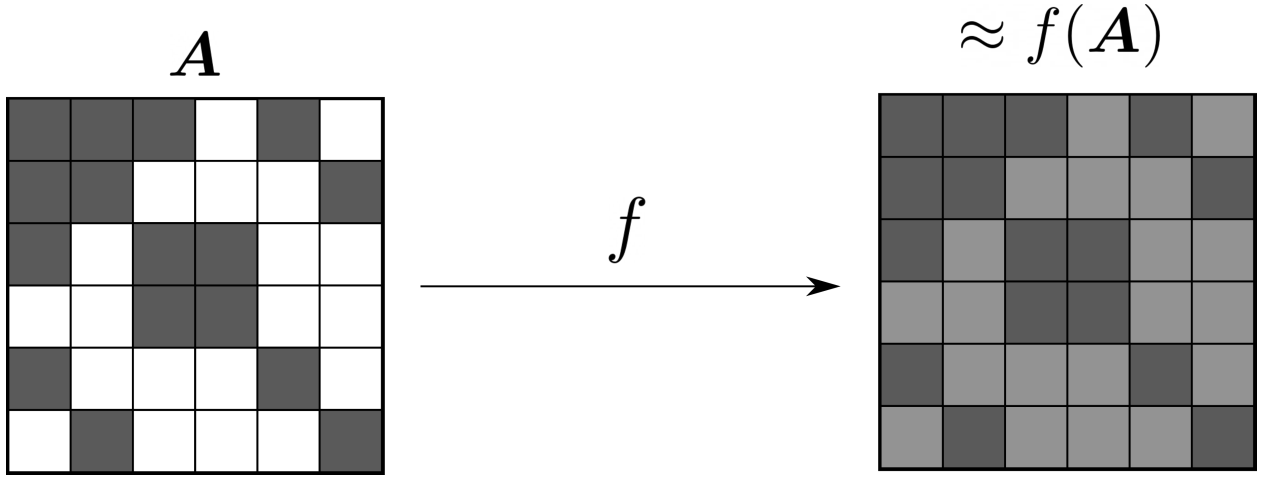
# Approximate Computing




V. Rengaraj M. Lass, C. Plessl and T. D. Kühne, Computation **8**, 39 (2020)

# Submatrix Method

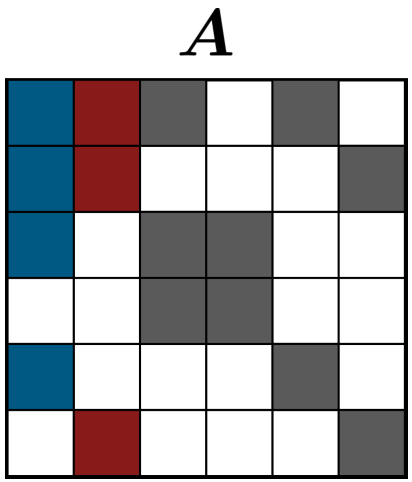
Purpose: Estimate matrix function (e.g. sign or inversion) of a large sparse matrix




M. Lass, S. Mohr, H. Wiebeler, TDK & C. Plessl, ACM Proc. of PASC **7**, 1 (2018)
M. Lass, R. Schade, T. D. Kühne & C. Plessl, IEEE Proc. of SC20 **1**, 80 (2020)

# Submatrix Method

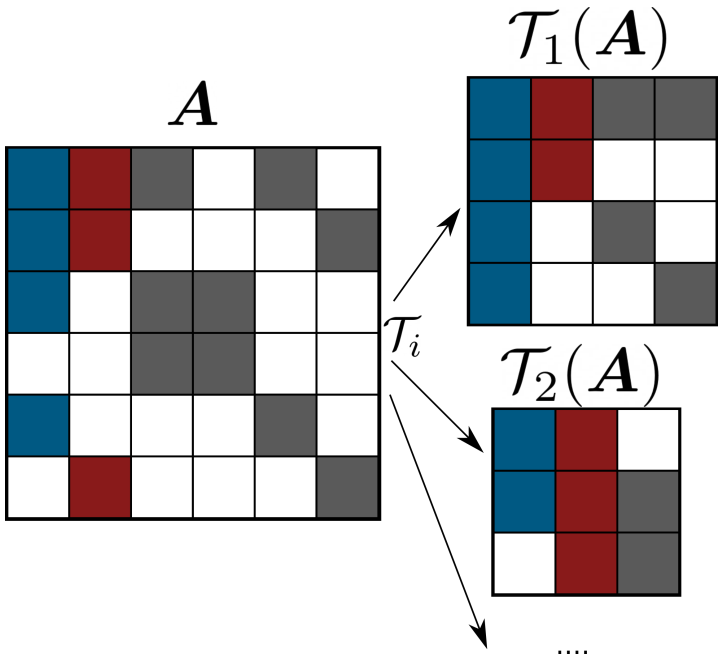
Step 1: Identify nonzero values in every column




M. Lass, S. Mohr, H. Wiebeler, TDK & C. Plessl, ACM Proc. of PASC **7**, 1 (2018)
M. Lass, R. Schade, T. D. Kühne & C. Plessl, IEEE Proc. of SC20 **1**, 80 (2020)

# Submatrix Method

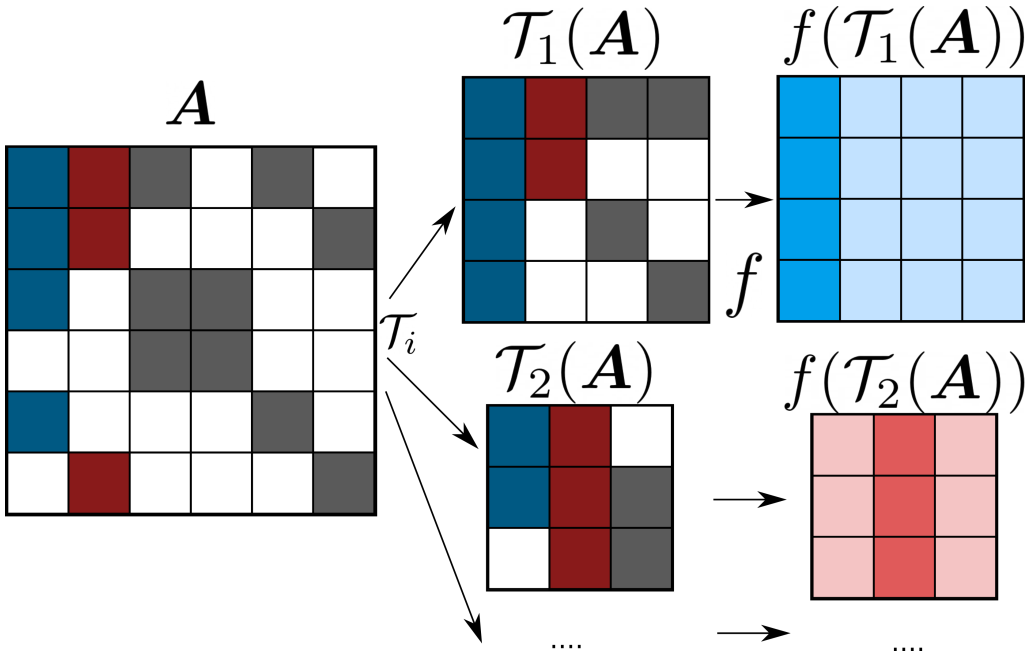
Step 2: Build submatrix $\mathcal{T}_i(\boldsymbol{A})$ for every column *i* with only the rows that have non-zero elements




M. Lass, S. Mohr, H. Wiebeler, TDK & C. Plessl, ACM Proc. of PASC **7**, 1 (2018)
M. Lass, R. Schade, T. D. Kühne & C. Plessl, IEEE Proc. of SC20 **1**, 80 (2020)

# Submatrix Method

Step 3: Apply matrix function *f* to submatrices $\mathcal{T}_i(\boldsymbol{A})$



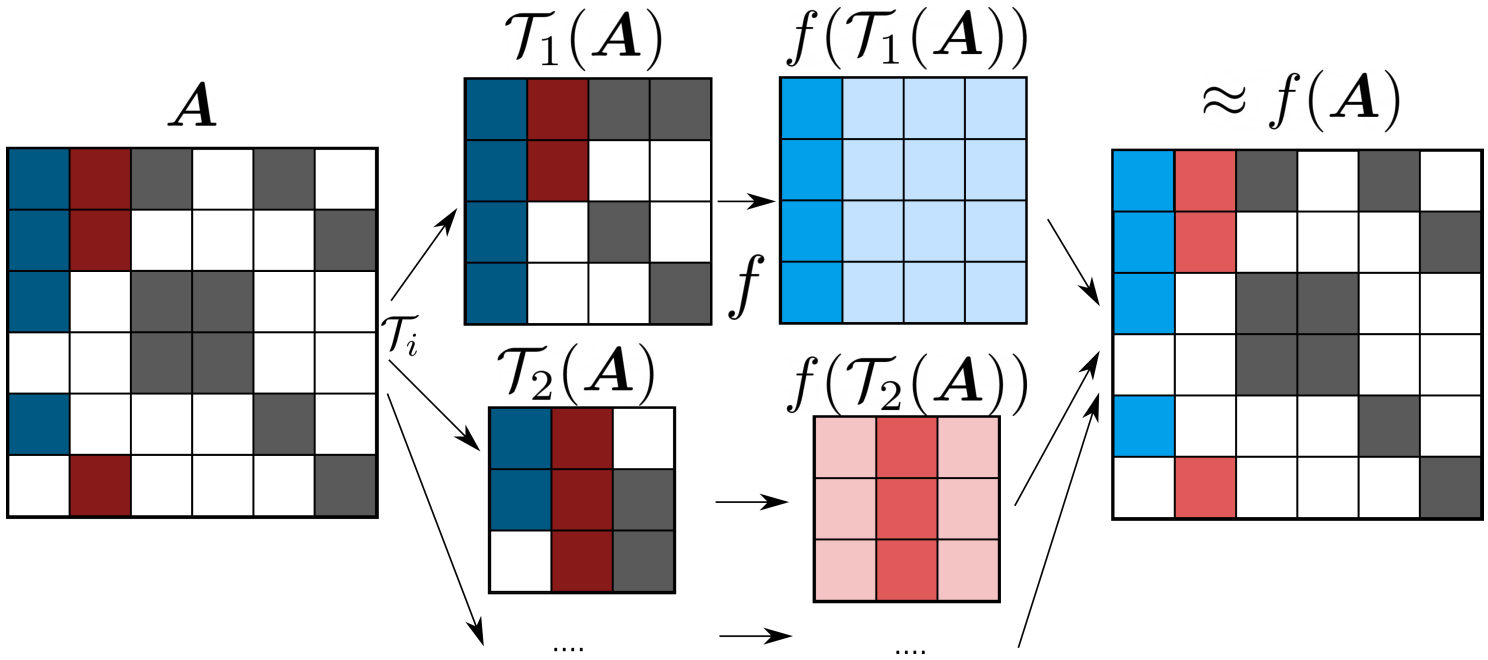


M. Lass, S. Mohr, H. Wiebeler, TDK & C. Plessl, ACM Proc. of PASC **7**, 1 (2018)

M. Lass, R. Schade, T. D. Kühne & C. Plessl, IEEE Proc. of SC20 **1**, 80 (2020)

# Submatrix Method

Step 4: copy resulting columns to result matrix

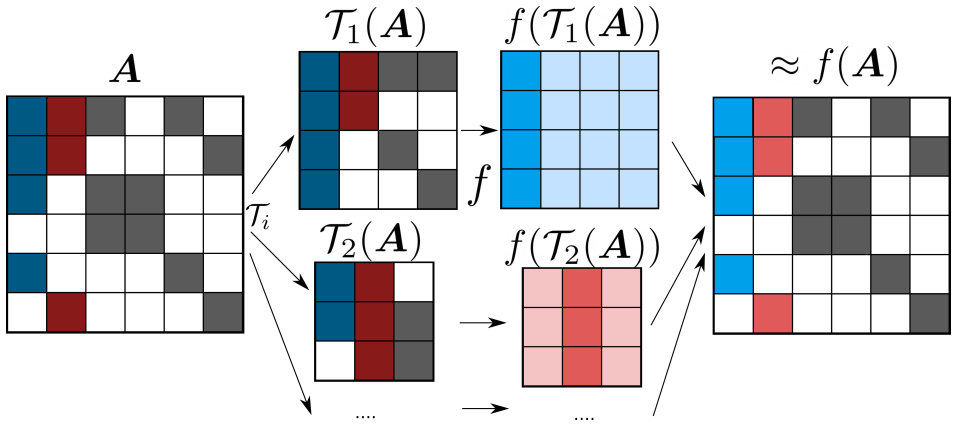



M. Lass, S. Mohr, H. Wiebeler, TDK & C. Plessl, ACM Proc. of PASC **7**, 1 (2018)
M. Lass, R. Schade, T. D. Kühne & C. Plessl, IEEE Proc. of SC20 **1**, 80 (2020)

# Submatrix Method

Properties of the Submatrix method:

- large distributed sparse matrix $\Rightarrow$ many small dense matrices
- suitable for dense linear algebra
- massively parallel
- linear-scaling approach



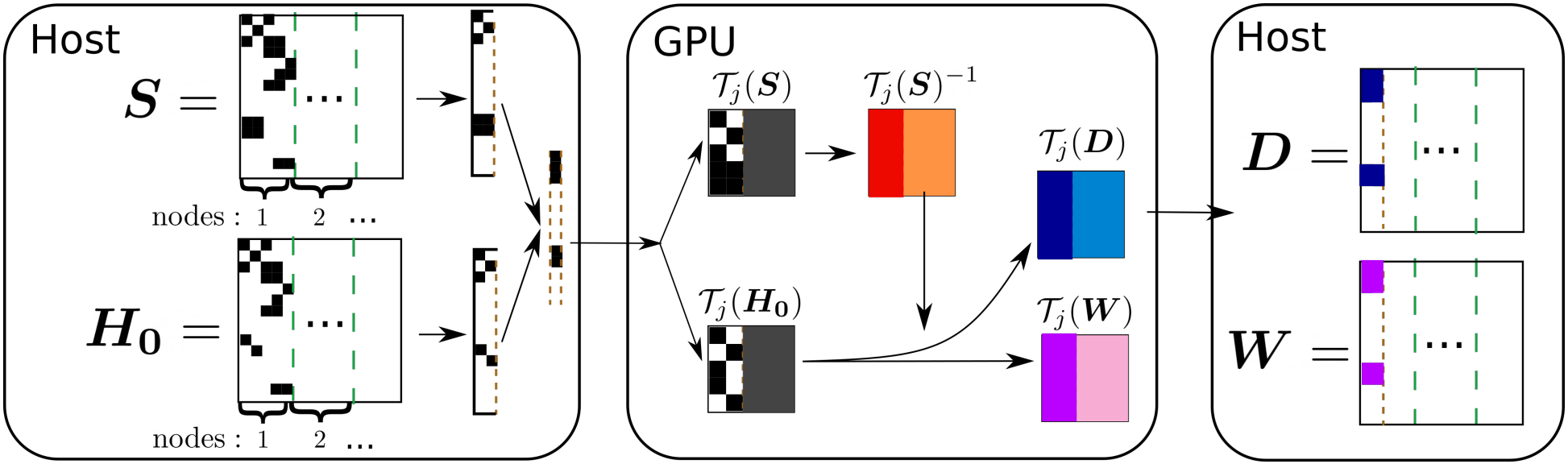


M. Lass, S. Mohr, H. Wiebeler, TDK & C. Plessl, ACM Proc. of PASC **7**, 1 (2018)

M. Lass, R. Schade, T. D. Kühne & C. Plessl, IEEE Proc. of SC20 **1**, 80 (2020)

# Non-Orthogonal Local SM

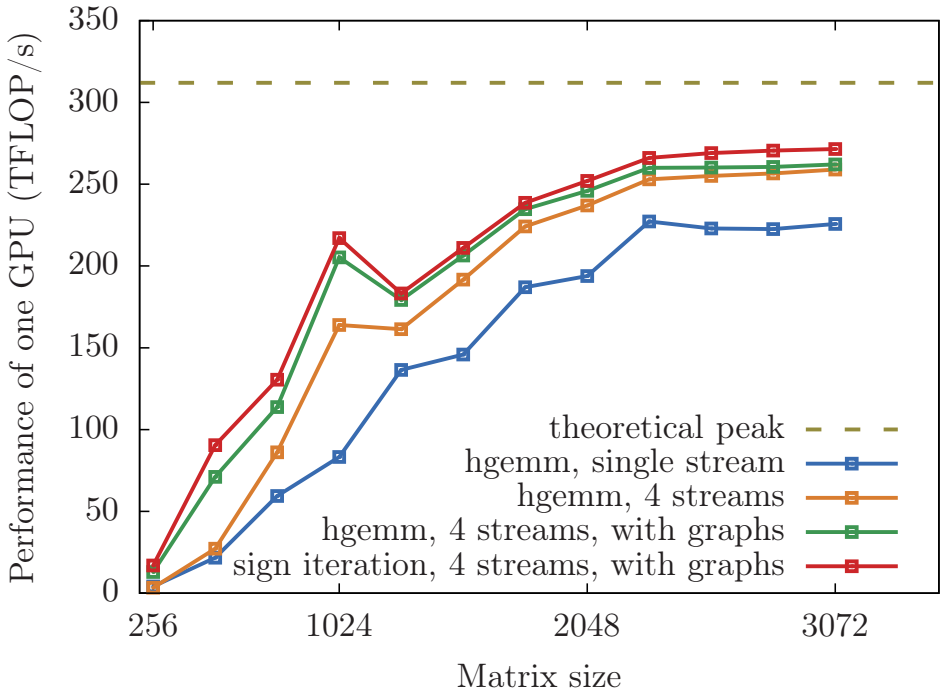
Implementation with GPUs:




$\Rightarrow$ minimal communication between nodes and CPU-to-GPU

R. Schade, T. Kenter, H. Elgabarty, M. Lass, ..., TDK & C. Plessl, arXiv:2104.08245 (2021)

# Non-Orthogonal Local SM




**Customized ”batching” for matrix iterations:**

**Kernel from cuBLAS**

**+CUDA streams:**
to load the full GPU

**+CUDA graphs:**
reduce kernel launch latency

$\Rightarrow$ for intermediate sized matrices ($\geq 1000$) about 60%-85% of peak can be reached

R. Schade, T. Kenter, H. Elgabarty, M. Lass, ..., TDK & C. Plessl, arXiv:2104.08245 (2021)

# Pre-Exascale Simulation

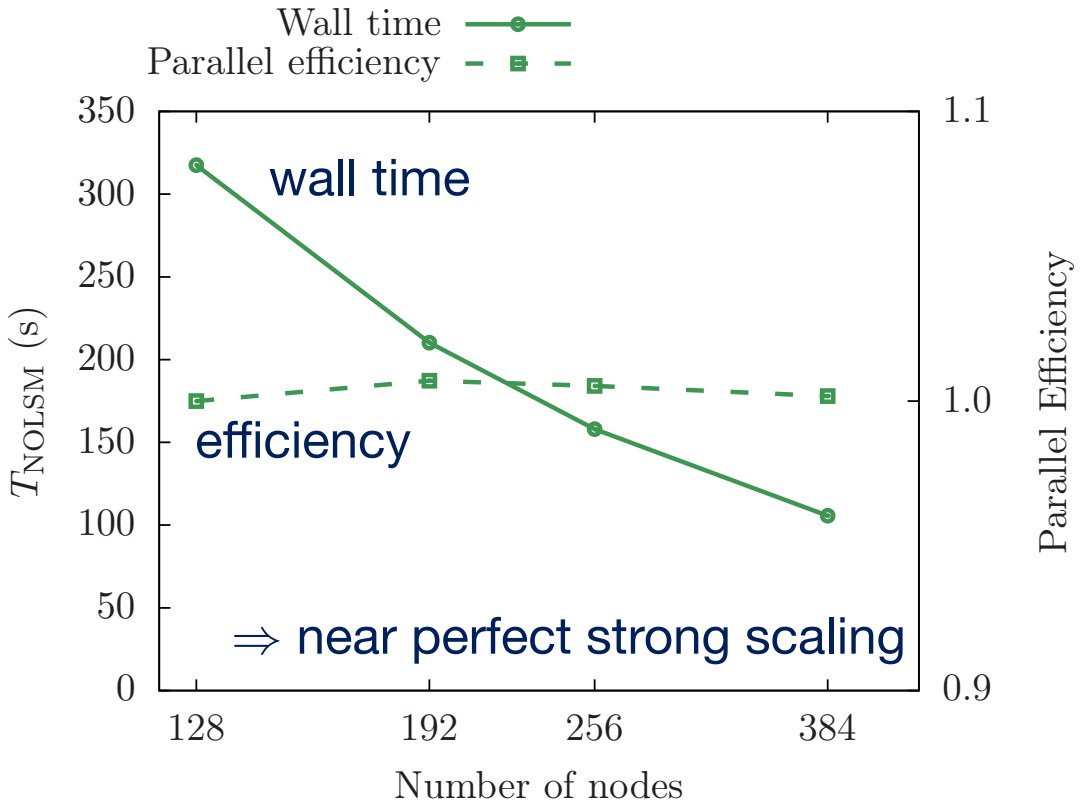
- 936 GPU-nodes with each:
  - CPU: 2xAMD EPYC 7402
  - Memory: 512 GB DDR4-3200 RAM
  - GPU: 4 × NVIDIA A100, 40 GB, NVLink3
  - Network: 4 × Mellanox HDR200 InfiniBand ConnectX 6 (200 Gbit/s each)
- Peak TC Performance:
  - FP64: 73 PFLOP/s
  - FP16/FP32: 1170 PFLOP/s


| Code, Year | Method | Basis | System | # Atoms | # Cores | Machine | Peak Performance | Efficiency |
|---|---|---|---|---|---|---|---|---|
| CPMD [22] 2005 | DFT | PW | bulk SiC | 1k | 1.2k CPU | IBM p690 | 1.087 TFlop/s | ≈ 20% |
| Qbox [23] 2006 | DFT | PW | bulk Mo | 8*1k | 128k CPU | IBM BlueGene/L | 207.3 TFlop/s | 56.5% |
| LS3DF [24] 2009 | DFT | PW | bulk ZnTeO | 36k | 147k CPU | Cray Jaguar | 442 TFlop/s | ≈ 33% |
| CP2K [25] 2012 | LS-DFT | GPW | bulk $H_2$ | 1m | 47k CPU | Cray XT5 | | |
| ONETEP [26] 2014 | LS-DFT | NGWF | amyloid fibril trimer | 42k | 115k CPU | IBM BlueGene/Q | | |
| RSDFT [27] 2014 | DFT | RS-FD | Si nanowire | 107k | 664k CPU | K-Computer | 5.48 PFlop/s | 51.67% |
| LDC-DFT [28] 2014 | SS-DFT | RMG-PW | bulk SiC | 6.3m | 786k CPU | IBM Blue Gene/Q | 5.08 PFlop/s | 50.5% |
| OpenAtom [29] 2016 | DFT | PW | periodic MOF | 32*424 | 262k CPU | IBM BlueGene/Q | | ≈ 52% |
| MGmol [30] 2016 | LS-DFT | FD | bulk $H_2O$ | 1.2m | 1.6m CPU | IBM BlueGene/Q | | ≈ 39% |
| DFT-FE [31] 2019 | DFT | FEM | Mg cluster | 10.5k | 159k CPU + 22.8k GPU | IBM Summit | 46 PFlop/s | 27.8% |
| CONQUEST [32] 2020 | LS-DFT | PAO | bulk Si | 1m | 200k CPU | K-Computer | | |

R. Schade, T. Kenter, H. Elgabarty, M. Lass, ..., TDK & C. Plessl, arXiv:2104.08245 (2021)

# Pre-Exascale Simulation

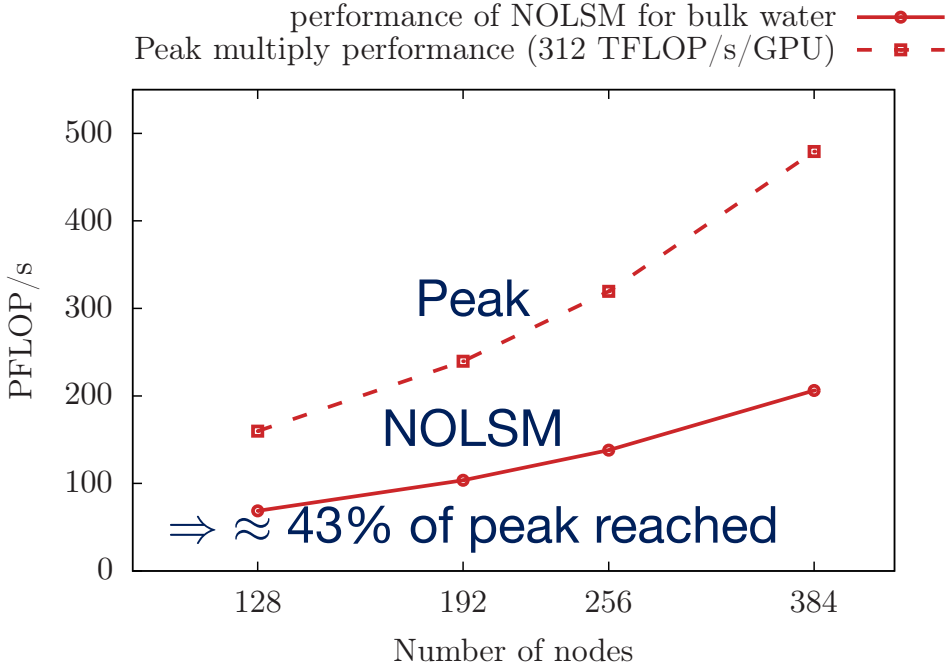
**Strong scaling of bulk water with 102 million atoms:**



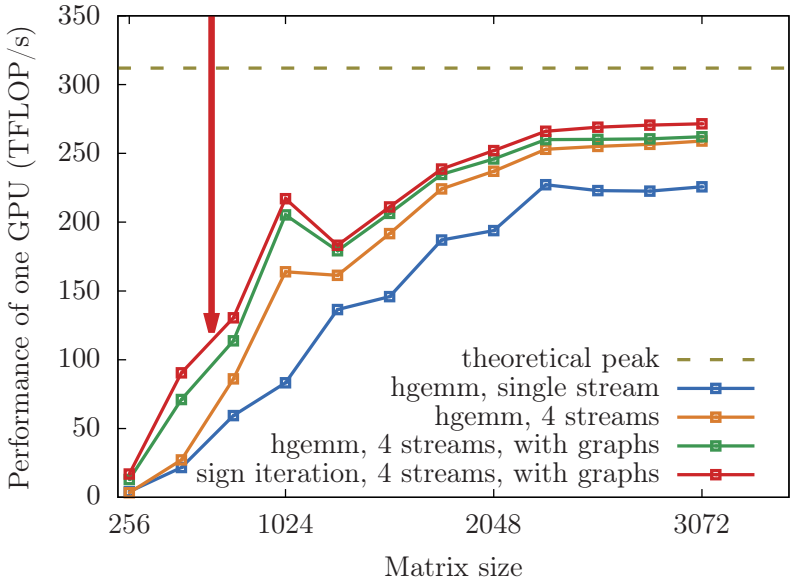

R. Schade, T. Kenter, H. Elgabarty, M. Lass, ..., TDK & C. Plessl, arXiv:2104.08245 (2021)

# Pre-Exascale Simulation

**Strong scaling of bulk water with 102 million atoms:**




average submatrix dimension ≈ 660

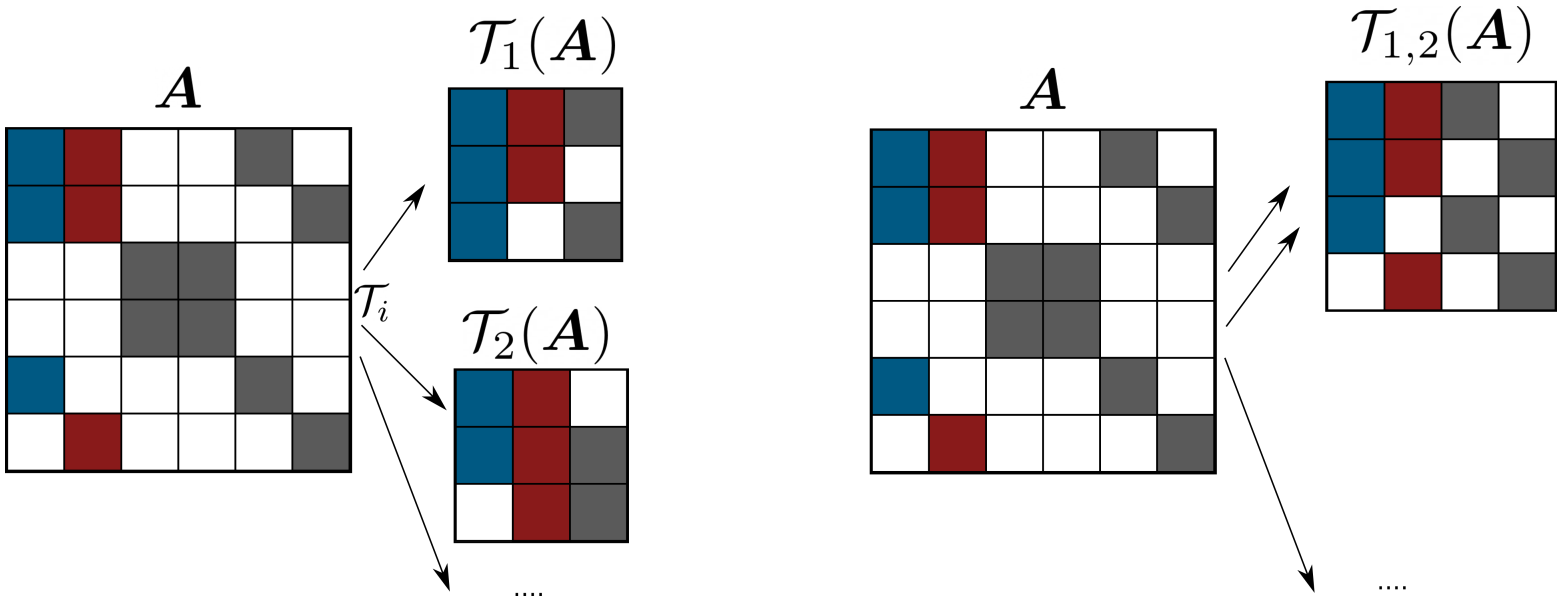



⇒ small matrix sizes limit achievable performance

R. Schade, T. Kenter, H. Elgabarty, M. Lass, ..., TDK & C. Plessl, arXiv:2104.08245 (2021)

# Combination of Submatrices

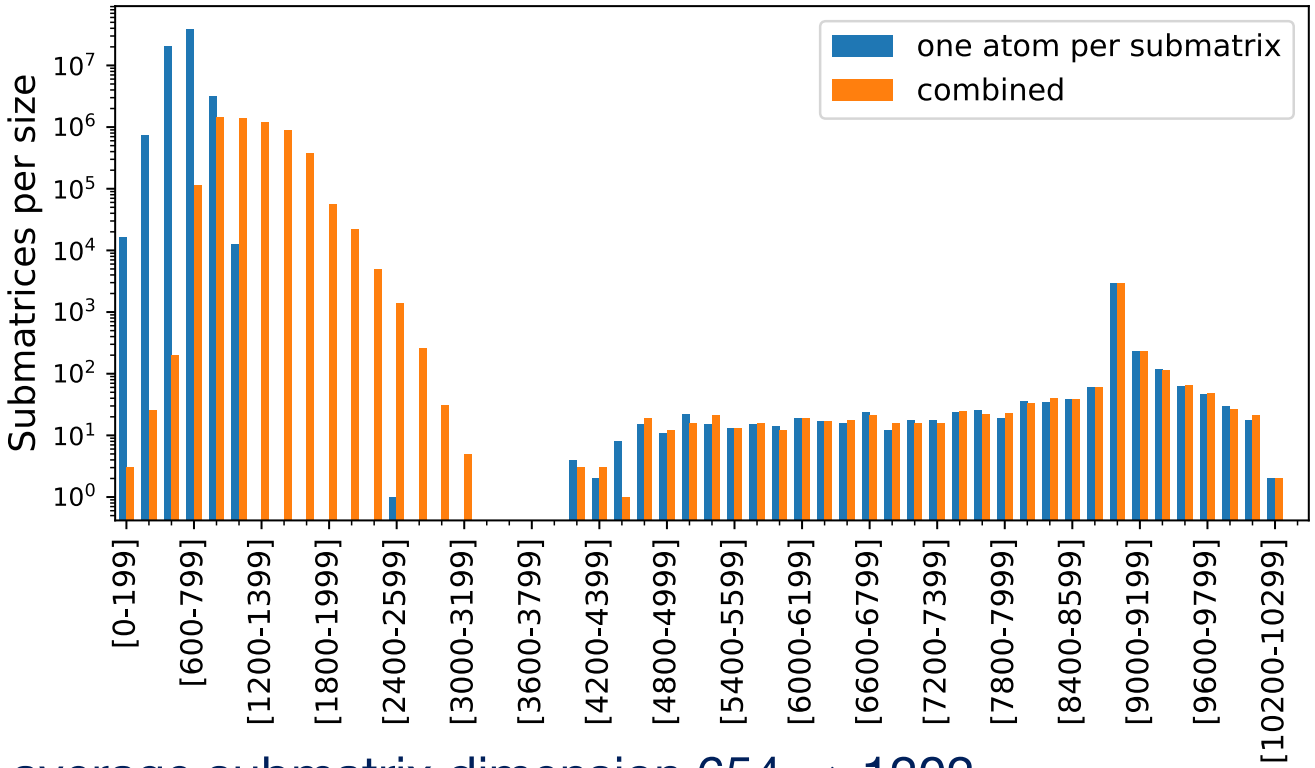
Idea: use flexibility of submatrix method and combine submatrices with similar columns




$\Rightarrow$ fewer but larger submatrices!

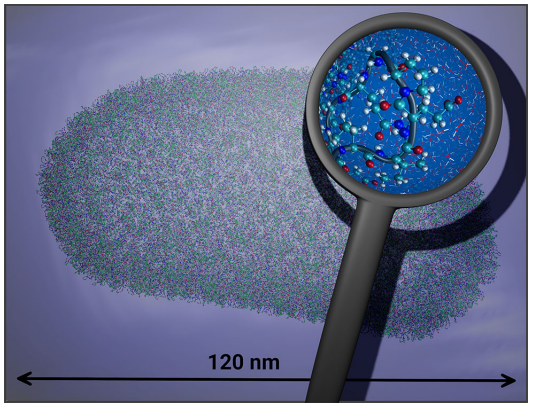
R. Schade, T. Kenter, H. Elgabarty, M. Lass, ..., TDK & C. Plessl, arXiv:2104.08245 (2021)

# Combination of Submatrices




average submatrix dimension 654 $\rightarrow$ 1202

**and** reduction of floating-point operations by factor 1.65!

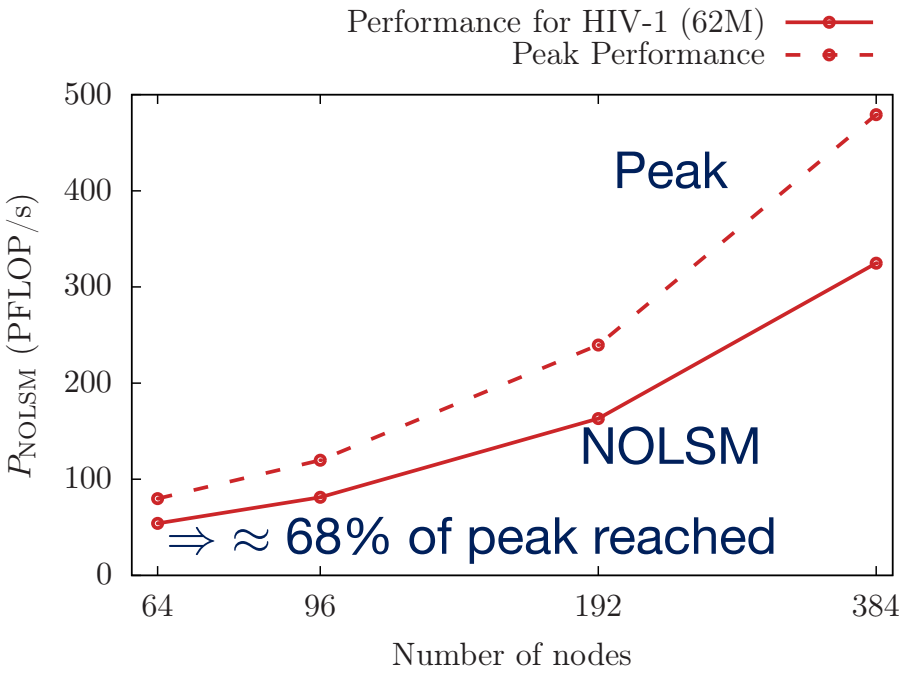



HIV-1 capsid in aqueous solution with 62 million atoms

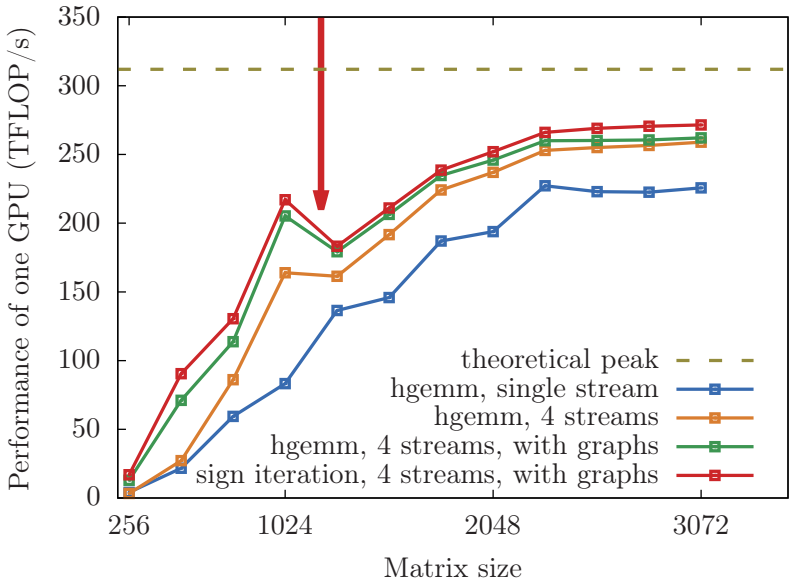
R. Schade, T. Kenter, H. Elgabarty, M. Lass, ..., TDK & C. Plessl, arXiv:2104.08245 (2021)

# Combination of Submatrices

**Strong scaling of HIV-1 with 62.5 million atoms:**




average submatrix dimension ≈ 1200

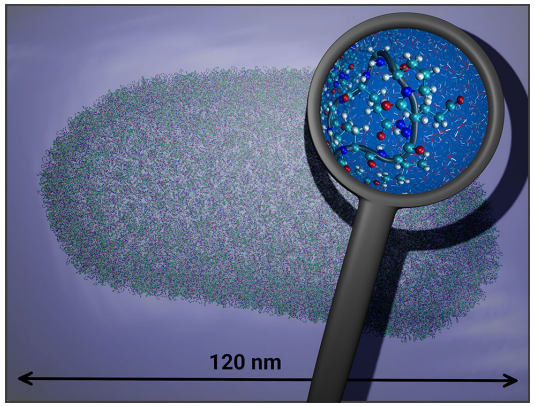



⇒ 324 PFlops FP16/FP32 for 384 nodes (68% of TC Peak)

R. Schade, T. Kenter, H. Elgabarty, M. Lass, ..., TDK & C. Plessl, arXiv:2104.08245 (2021)

# Summary & Outlook

- extended electronic-structure based molecular dynamics simulations to more than 100 million atoms
  - 102 million atoms for bulk water
  - 62 million atoms for HIV-1 capsid
- Non-orthogonal local submatrix (NOLSM) method:
  - for matrix functions of **large sparse matrices**
  - massively parallel **communication-avoiding** method
  - GPU-accelerated for NVIDIA GPUs
    - minimal transfer between host and GPUs
    - matrix construction on GPUs
    - mixed-precision on **NVIDIA tensor cores**
  - compensation schema for numerical noise
  - candidate for one of the first scientific FP16/FP32-exaflop simulations




HIV-1 capsid in aqueous solution with 62 million atoms

R. Schade, T. Kenter, H. Elgabarty, M. Lass, ..., TDK & C. Plessl, arXiv:2104.08245 (2021)

# Acknowledgements


- Robert Schade, PC2: NOLSM
- Michael Lass, PC2: Submatrix Method
- Tobias Kenter, PC2: Combination of Submatrices
- Christian Plessl, PC2: AC, FPGA
- Ole Schütt, Google: LIBCUSMM
- Valery Weber, IBM ZRL: DBCSR
- Alfio Lazzaro, Cray EMEA Research Lab: DBCSR
- Hans Pabst, Intel Extreme Computing: LIBXSMM
- Joost VandeVondele, CSCS: COSMA, LIBSMM_ACC
- Stephan Mohr, BSC: Submatrix Method
- Matthias Krack, PSI: CP2K
- Jürg Hutter, UZH: CP2K


Paderborn Center for Parallel Computing

Google

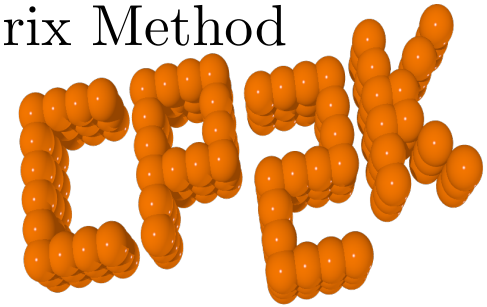
IBM Research | Zurich

CRAY

intel

CSCS
Centro Svizzero di Calcolo Scientifico
Swiss National Supercomputing Centre

BSC Barcelona Supercomputing Center
Centro Nacional de Supercomputación




T. D. Kühne et al., J. Chem. Phys. **152**, 194103 (2021)