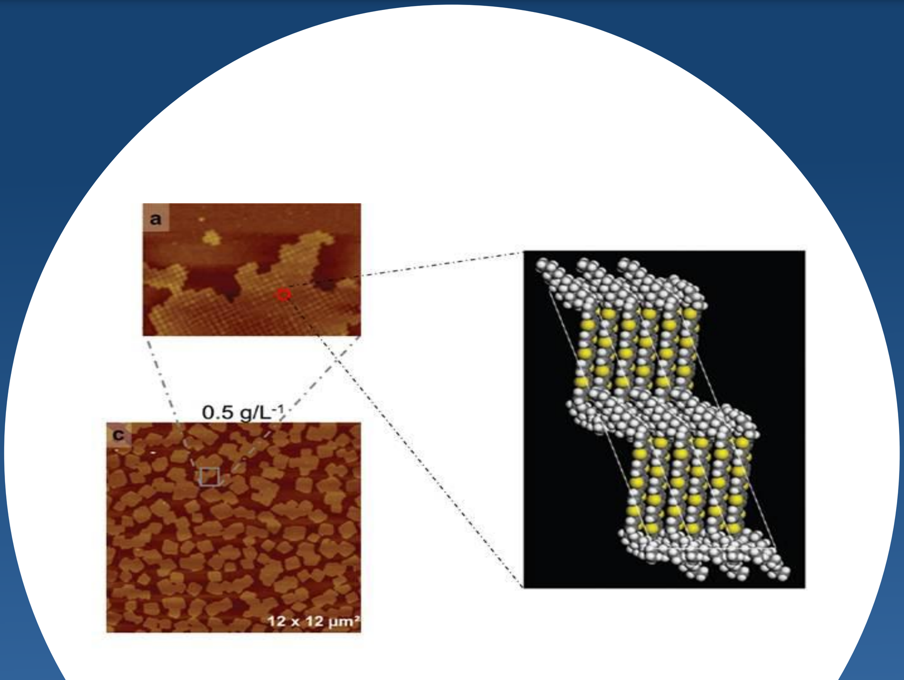
GPU-accelerated Bloch wave calculation for quantitative structure factor determination

Feng Wang Robert S. Pennington Christoph T. Koch
feng.wang@uni-ulm.de robert.pennington@uni-ulm.de christoph.koch@uni-ulm.de
Institute for Experimental Physics, Ulm University, Albert-Einstein-Allee 11, 89081 Ulm, Germany

Motivation



M. Defaux et al., Advanced Materials 24 (2012) 973–978

analysis of the atomic structure of very small volumes

- thin (nano crystals are not 150nm thick)
- light elements (organic)
- large unit cell
  - organic thin films for OLEDs, OFETs
  - 2D crystals 10-30nm thick and big unit cell

Dynamical electron diffraction simulation using the Bloch waves

the propagation of electrons in matrix notation

$$\psi(t) = \mathbf{S}(t)\psi(0)$$

S is the scattering matrix

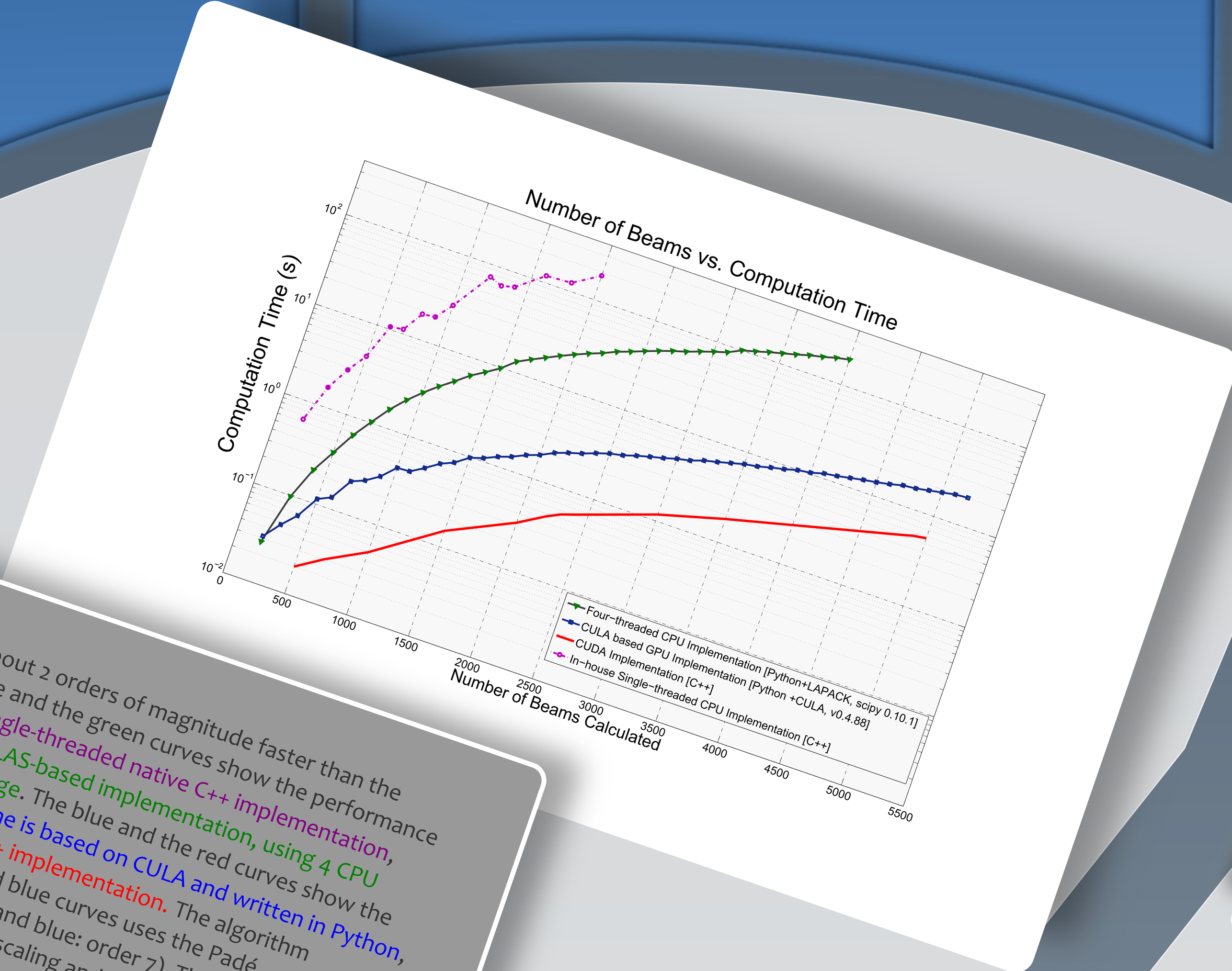
$$\mathbf{S} = e^{i\pi\lambda t\mathbf{A}}$$

A is the structure factor matrix

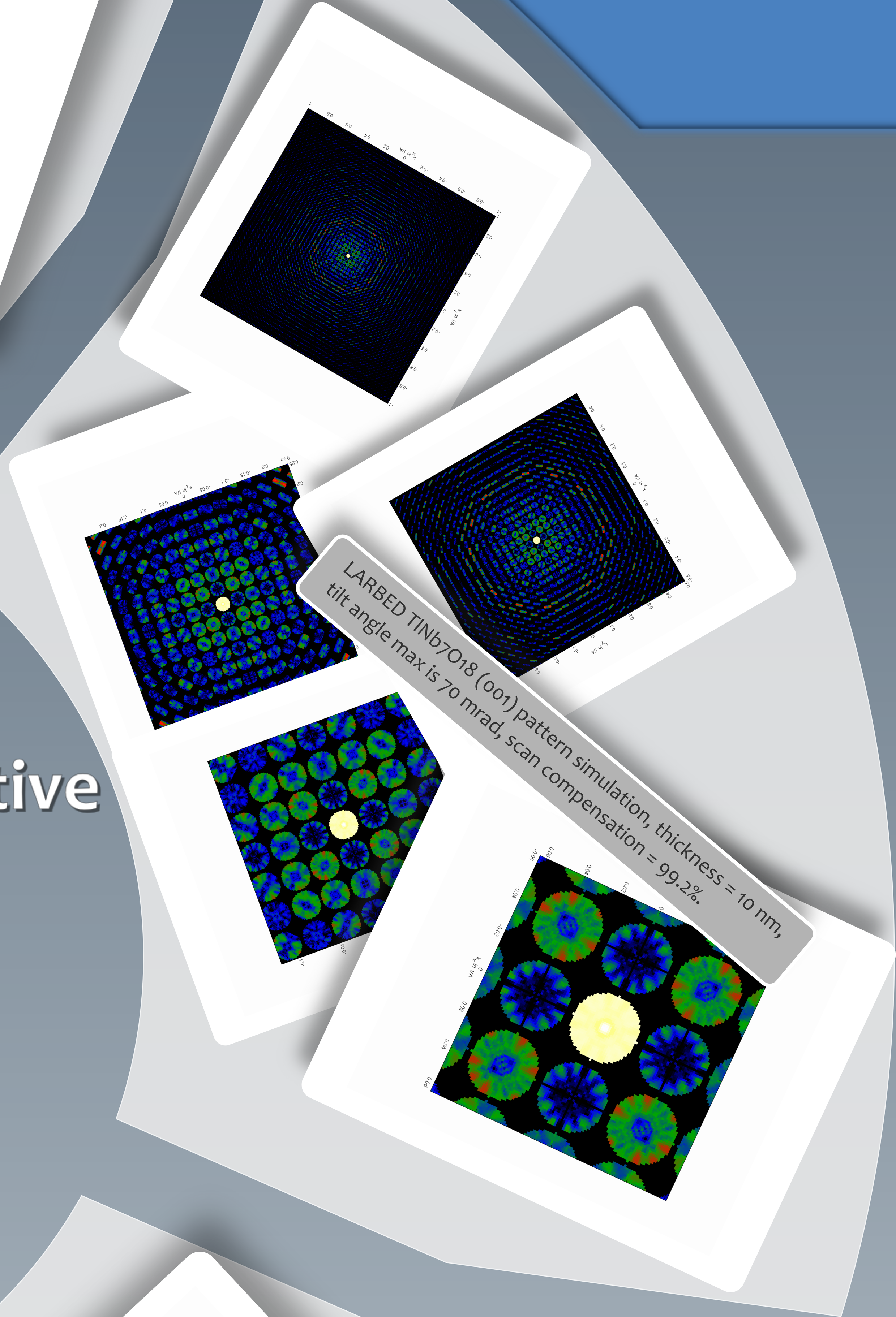
$$A = \begin{pmatrix} \ddots & \vdots & \vdots & \vdots & \vdots & \vdots & \iddots \\ \cdots & U_0-(k_t+h)^2 & U_{h-g} & U_h & U_{h+g} & U_{h+h} & \cdots \\ \cdots & U_{g-h} & U_0-(k_t+g)^2 & U_g & U_{g+g} & U_{g+h} & \cdots \\ \cdots & U_{-h} & U_{-g} & U_0-k_t^2 & U_g & U_h & \cdots \\ \cdots & U_{-g-h} & U_{-g-g} & U_{-g} & U_0-(k_t-g)^2 & U_{-g+h} & \cdots \\ \cdots & U_{-h-h} & U_{-h-g} & U_{-h} & U_{-h+g} & U_0-(k_t-h)^2 & \cdots \\ \iddots & \vdots & \vdots & \vdots & \vdots & \vdots & \ddots \end{pmatrix}$$

the intensity of the spots in the diffraction are squared moduli of the central column of S

$$I_{m,n} = |S_{m,n}|^2 = [e^{i\pi\lambda t\mathbf{A}}]_{m,n}[e^{i\pi\lambda t\mathbf{A}}]^*_{m,n}$$

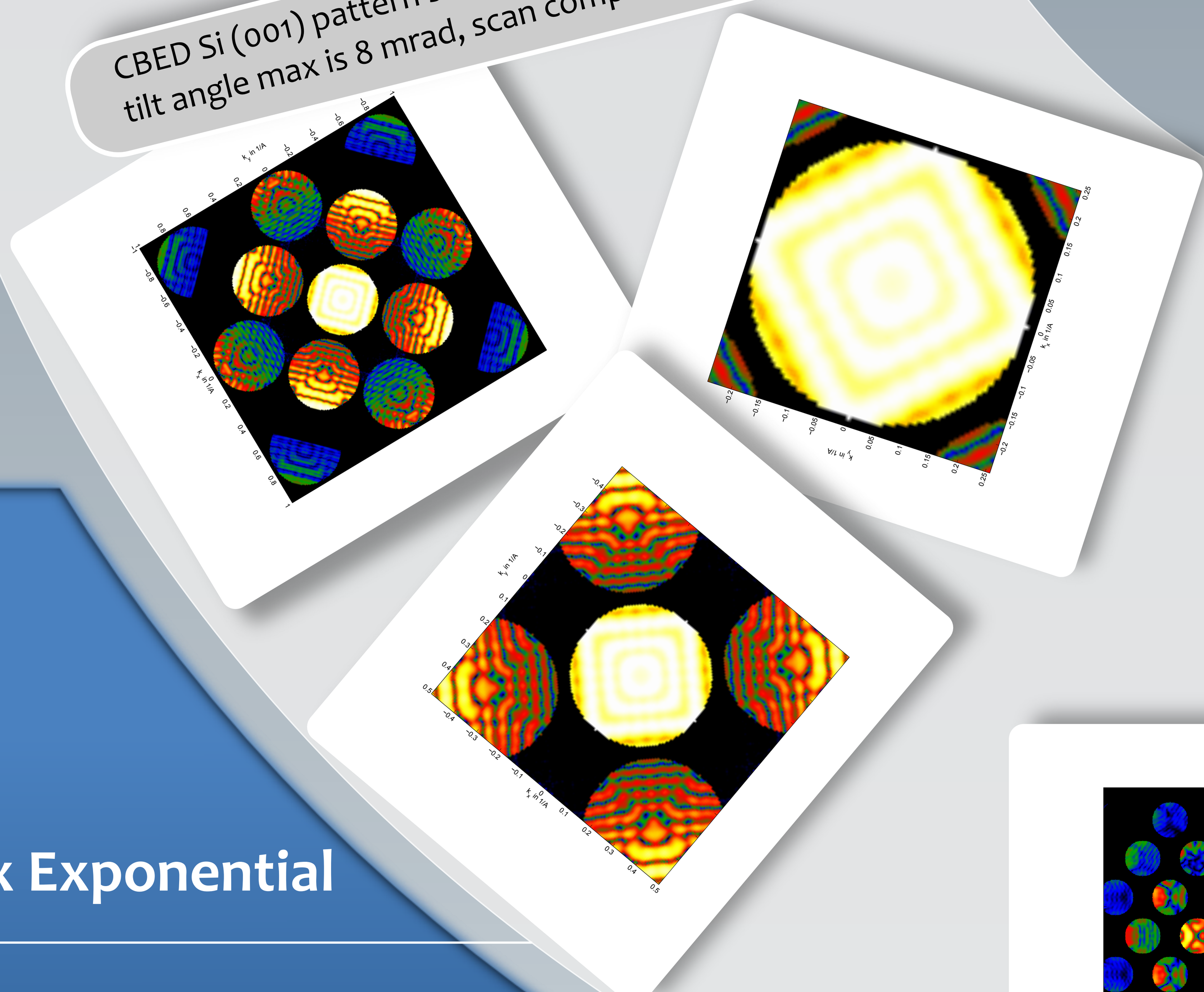


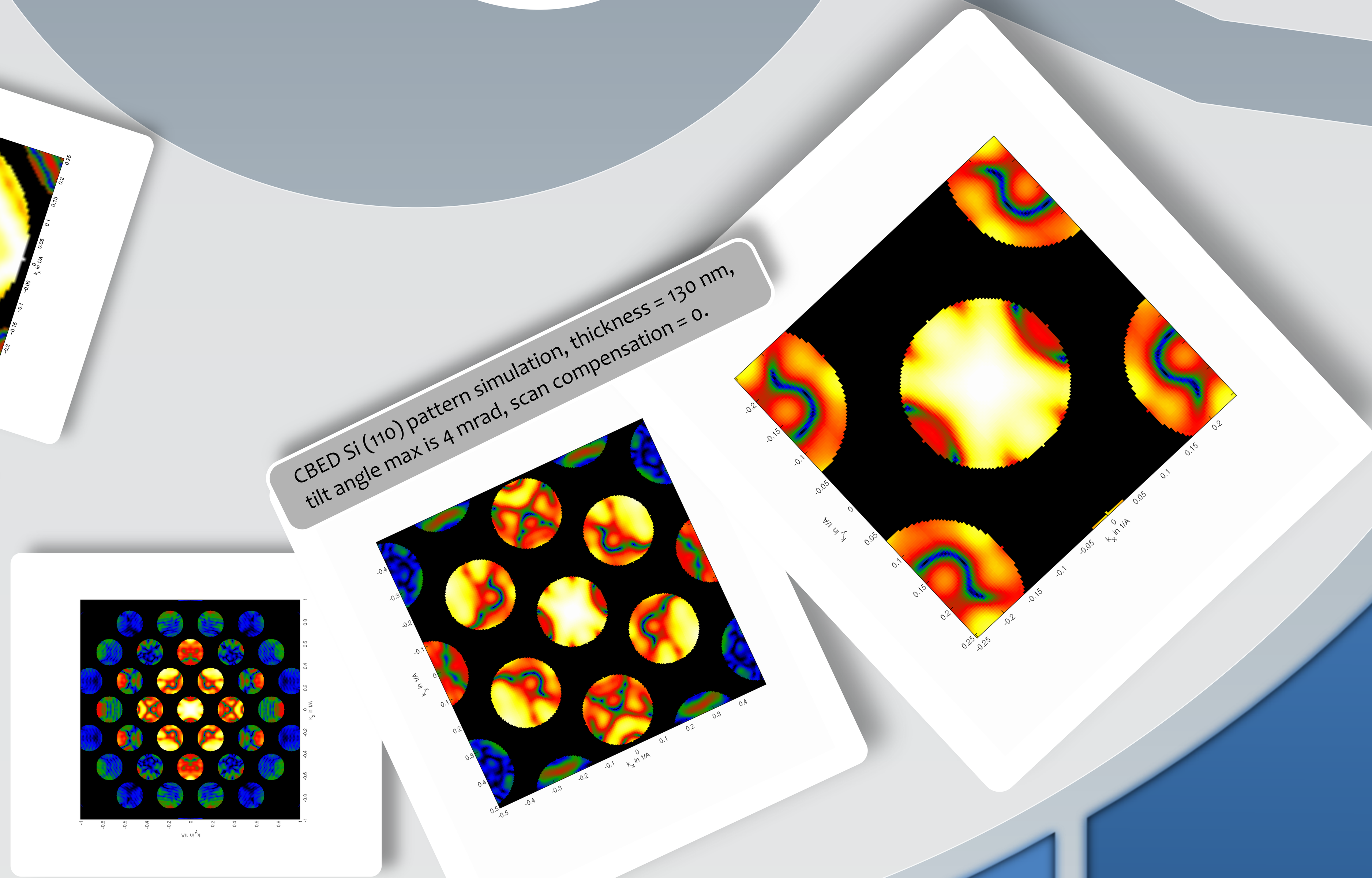
The GPU-based CUDA code is about 2 orders of magnitude faster than the parallelized CPU code. The purple and the green curves show the performance on the CPU: the purple one is a single-threaded native C++ implementation, while the green one is a LAPACK/BLAS-based implementation, using 4 CPU threads and the SciPy Python package. The blue and the red curves show the performance on the GPU: the blue one is based on CULA and written in Python, while the red one is a native CUDA C++ implementation. The algorithm corresponding to the purple, green and blue curves uses the Padé approximation (purple: order 14, green and blue: order 7). The algorithm corresponding to the red curve uses the scaling and squaring method.



LARBED TiNb7O18 (001) pattern simulation, thickness = 10 nm, tilt angle max is 70 mrad, scan compensation = 99.2%.

CBED Si (001) pattern simulation, thickness = 130 nm, tilt angle max is 8 mrad, scan compensation = 0.





CBED Si (110) pattern simulation, thickness = 130 nm, tilt angle max is 4 mrad, scan compensation = 0.

Matrix Exponential

Scaling & Squaring with Power Series

- $B \leftarrow \dfrac{A}{2^s}$ so $||B||_\infty \simeq 1$
- $e^B = I + B + \dfrac{B^2}{2!} + \dfrac{B^3}{3!} + \dfrac{B^4}{4!} + \dfrac{B^5}{5!} + \ldots$
- $e^A \simeq \{e^B\}^{\{2^s\}}$

Polynomial Truncation

$$e^B \simeq I + B + \frac{B^2}{2!} + \frac{B^3}{3!} + \frac{B^4}{4!} + \frac{B^5}{5!} + \frac{B^6}{6!} + \frac{B^7}{7!} + \frac{B^8}{8!} + \frac{B^9}{9!}$$

8 matrix-matrix multiplications

$$e^B \simeq (a_0I+a_1B+a_2B^2+a_3B^3)\,(b_0I+b_1B+b_2B^2+b_3B^3)\,(c_0I+c_1B+c_2B^2+c_3B^3)$$

4 matrix-matrix multiplications

or
a = { 1.04769359793033, 0.136216585231228, 0.175032858463863, 0.00019558817717526 };

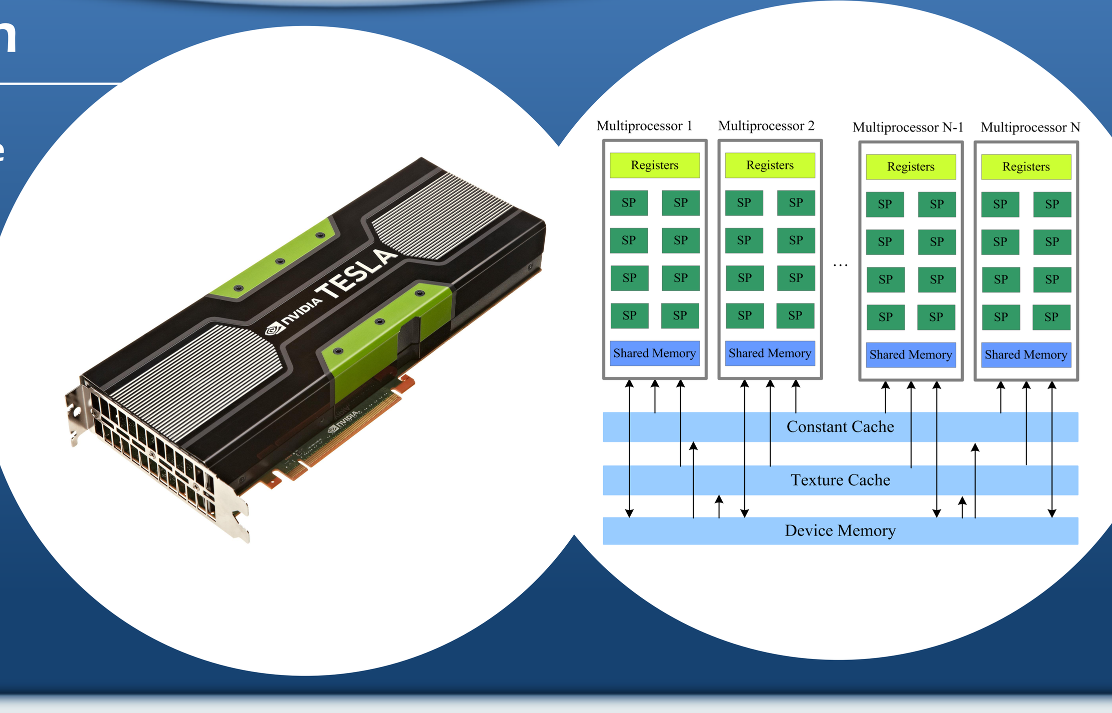
CUDA Acceleration

Nvidia Tesla K20C Performance

- Single Precision Performance 2.04 Tflops/s
- Double Precision Performance 1.16 Tflops/s

Intel(R) Xeon(R) CPU W3550

- Single Precision Performance 0.02 Tflop/s



References

1. C. Deininger, G. Necker and J. Mayer, Ultramicroscopy 54 (1994) pp. 15-30.
2. J.-M. Zuo, M. Kim, M. O'Keefe, and J.C.H. Spence, Nature 401 (1999) p. 49.
3. C.T. Koch, Ultramicroscopy 111 (2011) pp. 828-840.
4. C.T. Koch and J.C.H. Spence, J. Phys. A: Math. Gen. 36 (2003) pp. 803-816.
5. C.T. Koch, arXiv:0810.3811v1.
6. J.C.H. Spence and J.-M. Zuo, Electron Microdiffraction, (Plenum, New York) (1992).
7. C. B. Moler and C. F. Van Loan, SIAM Review 20 (1978) pp. 801-836.
8. C. B. Moler and C. F. Van Loan, SIAM Review 45 (2003) pp. 3-49.
9. M.L. Liou, Proc. Inst. Elect. Electron. Engrs. 54 (1966) pp. 20-23.
10. E. Jones, T. Oliphant, P. Peterson et al. SciPy: Open source scientific tools for Python.
11. J. R. Humphrey, D. K. Price, K. E. Spagnoli, A. L. Paolini, and E. J. Kelmelis, SPIE Defense and Security Symposium (DSS), 2010.
12. The authors acknowledge the Carl Zeiss Foundation as well as the German Research Foundation (DFG, Grant No. KO 2911/7-1).