



Hochleistungsrechnen

Mathematische Bibliotheken

Skriptversion 14.10.2021

Prof. Dr. Philipp Neumann

Helmut-Schmidt-Universität/Universität der Bundeswehr Hamburg

Hochleistungsrechnen

Mathematische Bibliotheken



That's me 😊

Skriptversion 01.11.2020

Prof. Dr. Philipp Neumann

Helmut-Schmidt-Universität/Universität der Bundeswehr Hamburg

Interested in Hiwi? Job Positions Open @HSU

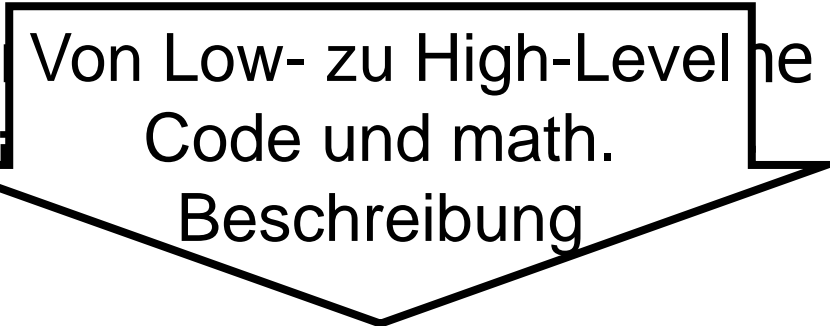
- Topics
 - Code optimization and performance analysis
 - Coupling particle- and grid-based simulations
 - Machine learning & numerical simulation
 - Hardware-aware implementations of simulations
- Requirements
 - Programming skills, preferably C++/Python
 - Interest in numerical simulation, high performance computing and Code-Gefrickel 😊
- Formal stuff
 - No regular presence times, flexible working environment
 - Start: anytime
 - Duration: at least 6 months, long-term perspectives
- If you're interested, please contact me:
Philipp Neumann, philipp.neumann@hsu-hh.de



Überblick

1. Mathematische Bibliotheken: Wieso? Weshalb? Warum?
2. Numerische Lineare Algebra und andere Grundbausteine
3. Löserpakete für Wissenschaftliche Anwendungen/PDEs
4. Software-Frameworks
5. Zusammenfassung und Wiederholung

Überblick

1. Mathematische Bibliotheken: Wieso? Weshalb? Warum?
2. Numerische Lineare Algebra und  Von Low- zu High-Level
Code und math.
Beschreibung
3. Löserpakete für Wissenschaftler
4. Software-Frameworks
5. Zusammenfassung und Wiederholung

1. Mathematische Bibliotheken: Wieso? Weshalb? Warum? (1)

Mathematische Grundfunktionalitäten als Bausteine in Vielzahl von Problemstellungen des wissenschaftlichen Rechnens und darüber hinaus

- Numerische Simulation
 - Lösen von (nicht-)linearen Gleichungssystemen, bspw. Poisson-Gleichung zur Bestimmung des physikalischen Drucks in inkompressiblen Strömungen
 - Vektor- und Matrixoperationen zur Lösung von ODEs, bspw. chemische Reaktionsprozesse
 - Prä-Konditionierung schlecht konditionierter Probleme, bspw. in der Struktur-/Fluidmechanik
 - ...

1. Mathematische Bibliotheken: Wieso? Weshalb? Warum? (2)

Mathematische Grundfunktionalitäten als Bausteine in Vielzahl von Problemstellungen des wissenschaftlichen Rechnens und darüber hinaus

- Datenanalyse

- Glätten von Daten durch Matrix-Vektor-Operationen
- Datenfilter, bspw. via Fourieranalysen und Hauptkomponentenanalysen
- ...

1. Mathematische Bibliotheken: Wieso? Weshalb? Warum? (3)

Mathematische Grundfunktionalitäten als Bausteine in Vielzahl von Problemstellungen des wissenschaftlichen Rechnens und darüber hinaus

- Maschinelles Lernen
 - Algorithmik innerhalb Deep Learning/Neuronaler Netze basierend auf Vektor/Matrix/Tensor-Operationen
 - Spezifische Gleichungssysteme in Regressionsansätzen, bspw. Least-Squares-Systeme (symmetrisch positiv definite Matrizen)
 - ...
 - Fun Fact: Schließung des Kreises
„Lineare Algebra \rightarrow Maschinelles Lernen \rightarrow Lineare Algebra“,
siehe bspw. Götz und Anzt (2018) für Prä-Konditionierung

1. Mathematische Bibliotheken: Wieso? Weshalb? Warum? (4)

Vielzahl an sich schnell weiterentwickelnder Hardware

- Mathematische Bibliotheken als Baustein zwischen hardware-optimierter Grundfunktionalität und anwendungsspezifischer Problemstellung
- Hardware-spezifische Bibliotheken, bspw. für Intel-Architekturen, Nvidia GPUs

1. Mathematische Bibliotheken: Warum nicht?

**Gruppen- bzw.
Einzelarbeit 😊
3 Min.**



1. Mathematische Bibliotheken: Warum nicht?

- Software-Abhängigkeiten
- Bitweise Reproduzierbarkeit
- Granularität: Low- vs. High-Level-Implementierungen
- Bedienbarkeit/Nutzbarkeit
- Nachhaltigkeit: Long-term Support für Bibliotheken?

2. Numerische Lineare Algebra und andere Grundbausteine

- **BLAS, ATLAS**
- **LAPACK, Scalapack**
- **NumPy**
- Hypre: Scalable Linear Solvers and Multigrid Methods
 - Weitere Informationen: <https://computation.llnl.gov/projects/hypr-scalable-linear-solvers-multigrid-methods>
- ParMETIS
 - MPI-parallele Bibliothek zur Partitionierung unstrukturierter Graphen
 - Weitere Informationen: <http://glaros.dtc.umn.edu/gkhome/metis/parmetis/overview>
- FFTW (Fastest Fourier Transform in the West)
 - Weitere Informationen: <http://www.fftw.org/benchfft/>

BLAS

- BLAS=Basic Linear Algebra Subprograms, eingeteilt in 3 Levels
- Implementiert grundlegende Bausteine aus der linearen Algebra
 - Skalierung eines Vektors mit Konstante
 - Matrix-Vektor-Multiplikation
 - Matrix-Matrix-Multiplikation (GEMM=general matrix-matrix multiply)
 - Spezialisierungen für symmetrische,hermitsche/Dreiecks-/Bandmatrizen
 - ...
- Verschiedenste Implementierungen (inklusive Hardware-Hersteller-Implementierungen)
 - OpenBLAS
 - Intel Math Kernel library (MKL), Nvidia's cuBLAS
 - Automatically tuned linear algebra software (ATLAS)
- Weitere Informationen: www.netlib.org/blas/

BLAS

- BLAS=Basic Linear Algebra Subprograms, eingeteilt in 3 Levels
- Implementiert grundlegende Bausteine aus der linearen Algebra
 - Skalierung eines Vektors mit Konstante
 - Matrix-Vektor-Multiplikation
 - Matrix-Matrix-Multiplikation (GEMM=general matrix-matrix multiply)
 - Spezialisierungen für symmetrische,hermitsche/Dreiecks-/Bandmatrizen
 - ...
- Verschiedenste Hersteller-Implementierungen
 - OpenBLAS
 - Intel Math Kernel Library (MKL), NVIDIA'S cuBLAS
 - Automatically tuned linear algebra software (ATLAS)
- Weitere Informationen: www.netlib.org/blas/


Was nützt Selbstadaption/
Auto-Tuning für BLAS und
HPC?

- Ziel: Automatische Optimierung für spezifische Hardware
- Beispiel: Matrix-Matrix-Multiplikation
 - Optimales Cache-Blocking
- Weitere Informationen:
 - <http://www.netlib.org/lapack/lawnspdf/lawn131.pdf>
 - R.C. Whaley, J.J. Dongarra. Automatically tuned linear algebra software. SC '98 proceedings, pp. 1-27, 1998

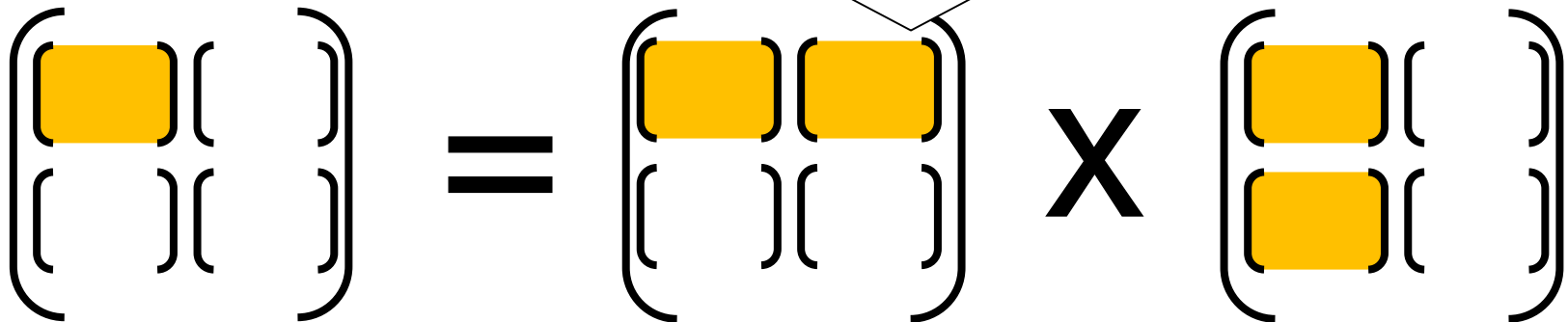
$$\left(\quad \right) = \left(\quad \right) \times \left(\quad \right)$$

ATLAS

- Ziel: Automatische Optimierung für spezifische Hardware
- Beispiel: Matrix-Matrix-Multiplikation
 - Optimales Cache-Blocking
- Weitere Informationen
 - <http://www.netlib.org>
 - R.C. Whaley, J.J. Dongarra: ScaLAPACK: Linear algebra software. SC98

Tuning der Blockgröße  zur Verhinderung von Cache Misses

[pdf](#)
linear
1998

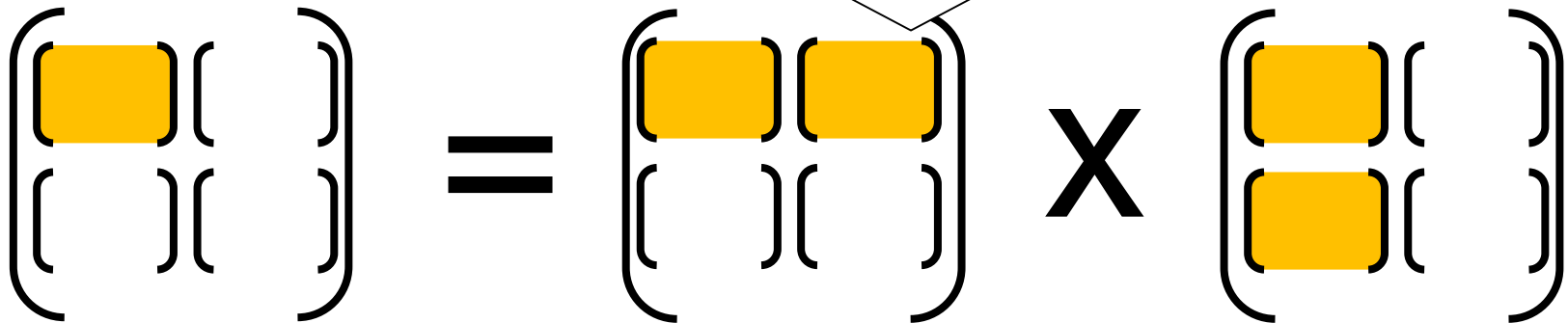


ATLAS

- Ziel: Automatische Optimierung
- Beispiel: Matrix-Matrix-Multiplikation
→ Optimales Cache-Blocking
- Weitere Informationen
 - <http://www.netlib.org>
 - R.C. Whaley, J.J. Dongarra. ScaLAPACK: Linear algebra software. SC98

Auto-Tuning ist
Gegenstand
aktueller HPC-
Forschung

Tuning der Block-
größe zur
Verhinderung von
Cache Misses



LAPACK

- LAPACK=Linear Algebra Package
- Implementierungen für
 - Lösung linearer Gleichungssysteme
 - Lineare Ausgleichsprobleme
 - Eigenwertprobleme
 - QR-Zerlegung, Householder-Transformation, Singulärwertzerlegung, ...
- Verwendung von BLAS
- Verschiedene Implementierungen (MKL, ...)
- Weitere Informationen: <http://www.netlib.org/lapack/>

BLAS und LAPACK: Parallelisierung

- Parallelisierung und Threadsicherheit abhängig von Implementierung
- OpenBLAS¹: „If your application is already multi-threaded, it will conflict with OpenBLAS multi-threading. Thus, you must set OpenBLAS to use single thread as following.
 - export OPENBLAS_NUM_THREADS=1 in the environment variables. Or
 - Call `openblas_set_num_threads(1)` in the application on runtime. Or
 - Build OpenBLAS single thread version, e.g. make `USE_THREAD=0 USE_LOCKING=1` (see comment below)

If the application is parallelized by OpenMP, please build OpenBLAS with `USE_OPENMP=1`”

- Intel oneAPI MKL²: „Is a seamless upgrade ... of the Intel Math Kernel Library”
 - „Data Parallel C++ (DPC++) APIs maximize performance and cross-architecture portability
 - Introduces C and Fortran OpenMP offload for Intel GPU acceleration”

[1https://github.com/xianyi/OpenBLAS/wiki/faq#multi-threaded](https://github.com/xianyi/OpenBLAS/wiki/faq#multi-threaded)

[2https://software.intel.com/content/www/us/en/develop/tools/oneapi/components/onemkl.html?wapkw=\(Intel\)](https://software.intel.com/content/www/us/en/develop/tools/oneapi/components/onemkl.html?wapkw=(Intel))

ScaLAPACK

- ScaLAPACK=Scalable LAPACK, entworfen für MIMD-Systeme
- Weitere Informationen:
 - http://www.netlib.org/utk/people/JackDongarra/PAPERS/077_1996_scalapack-a-portable-linear-algebra-library-for-distributed-memory.pdf
 - J. Choi et al. ScaLAPACK: a portable linear algebra library for distributed memory computers — design issues and performance. Computer Physics Communications 97(1-2): 1-15, 1996
- Ähnliche Entwicklungen:
 - PLASMA=Parallel Linear Algebra Software for Multicore Architectures (<http://icl.cs.utk.edu/plasma/software/>)
 - MAGMA=Matrix Algebra on GPU and Multicore Architectures (<http://icl.cs.utk.edu/magma/>)

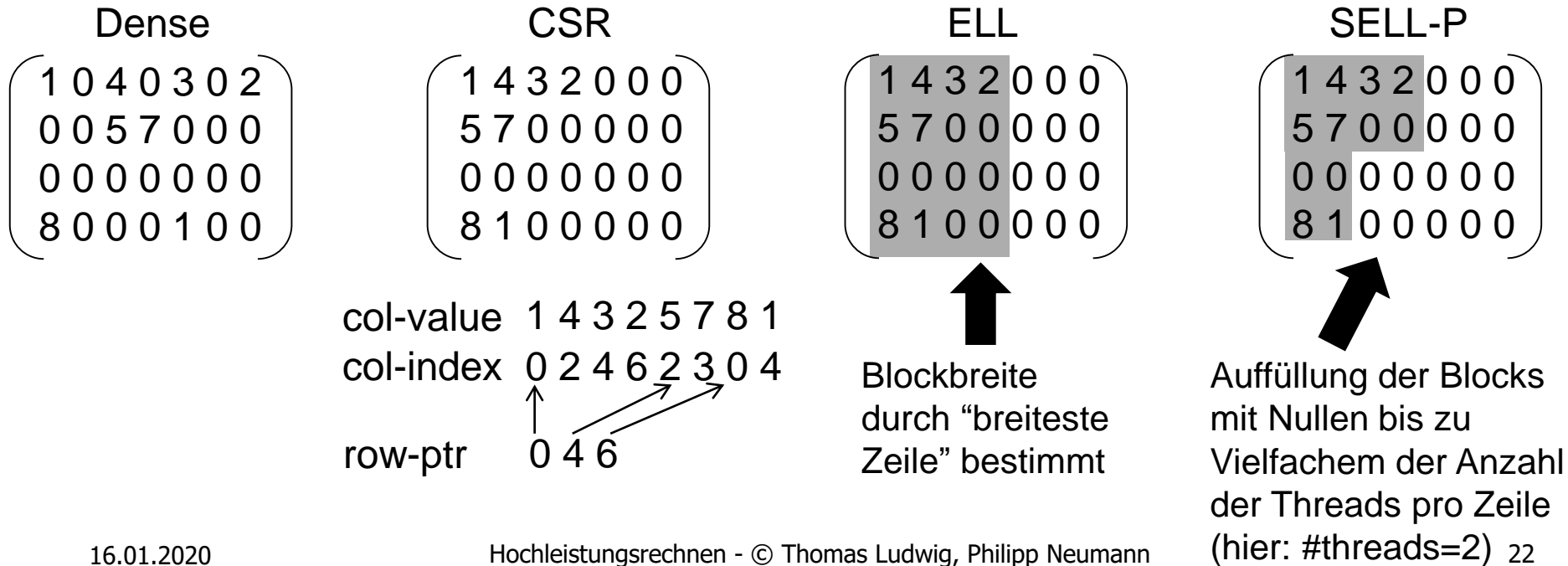
ScaLAPACK

- ScaLAPACK=Scalable LAPACK, entworfen für MIMD-Systeme
- Weitere Informationen:
 - http://www.netlib.org/utk/people/JackDongarra/PAPERS/077_1996_scalapack-a-portable-linear-algebra-library-for-distributed-memory.pdf
 - J. Choi et al. ScaLAPACK: a portable linear algebra library for distributed memory computers — design and implementation. *Computer Physics Communications* 97(1):23-32, 1995.
- Ähnliche Entwicklungen:
 - PLASMA=Parallel Linear Algebra Software for Heterogeneous Architectures (<http://icl.cs.utk.edu/plasma/software/>)
 - MAGMA=Matrix Algebra on GPU and Multicore Architectures (<http://icl.cs.utk.edu/magma/>)

Entwickelt für
heterogene
Architekturen

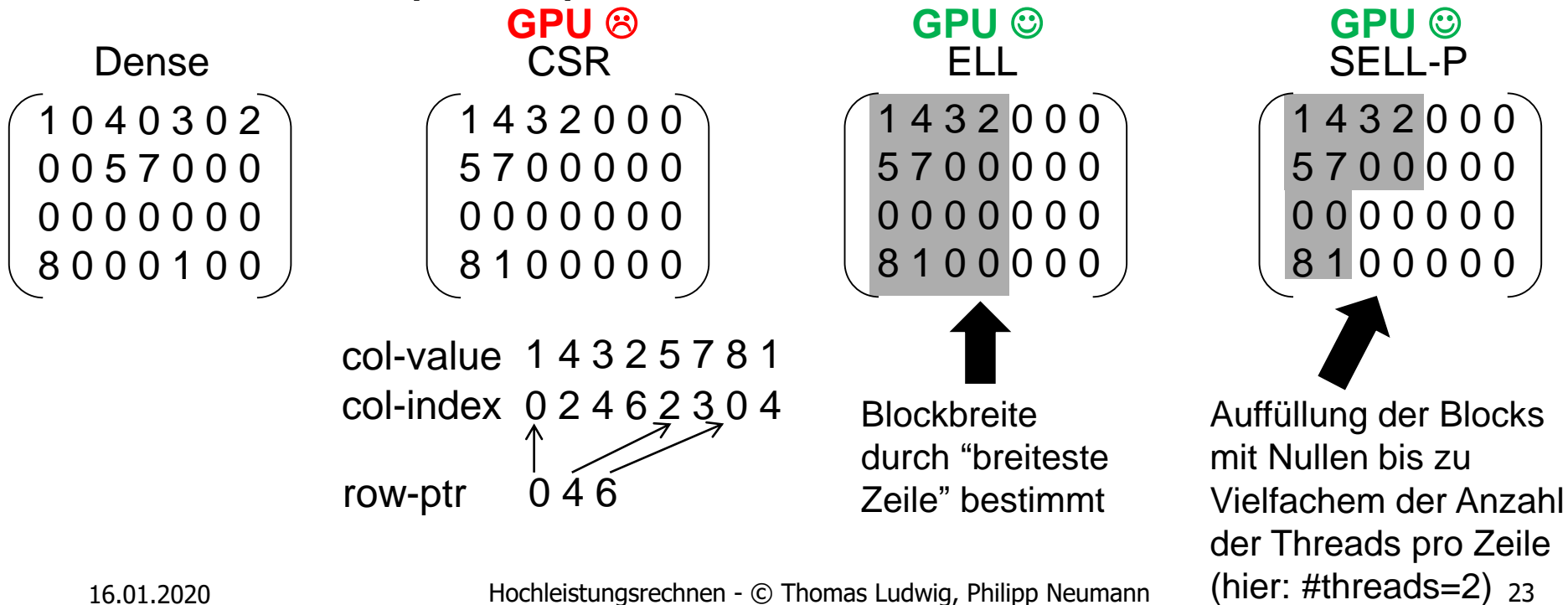
Exkurs: Dünnbesetzte Matrizen/SPMV (1)

- SPMV=Sparse Matrix Vector Multiplication
- Datenstrukturen (bspw. MAGMA):
Compressed Storage Row (CSR), ELLPACK (ELL), padded sliced
ELLPACK (SELL-P)



Exkurs: Dünnbesetzte Matrizen/SPMV (1)

- SPMV=Sparse Matrix Vector Multiplication
- Datenstrukturen (bspw. MAGMA):
Compressed Storage Row (CSR), ELLPACK (ELL), padded sliced ELLPACK (SELL-P)



Exkurs: Dünnbesetzte Matrizen/SPMV (2)

- Shared-Memory Parallelisierung
 - Zeilenweise Verteilung und Thread-Scheduling
 - Pinning von #Zeilen pro Thread
 - Anfällig für Last Imbalanz
 - Dynamische Verteilung der Zeilen an Threads
 - Segmentverteilung (basierend auf Nicht-Null-Einträgen)
 - CSR: Pinning der drei Arrays auf einen Memory-Controller ☹
 - Partitioniertes Speichern der Arrays
 - CSR: Konträr zu SIMD-Nutzung
 - Alternative: Submatrizen/-zeilen verteilen
 - ELL/SELL-P: Effizientes Handling von Matrix-Blöcken
 - erhöhte Last durch Nulleinträge

Exkurs: Dünnbesetzte Matrizen/SPMV (3)

- Distributed-Memory Parallelisierung
 - (Gleichmäßige) Verteilung der Input/Output-Vektoren auf Prozesse
 - Verteilung von Zeilenblöcken auf Prozesse
 - Aufteilung der Zeilenblöcke: On-Process vs. Off-Process Block/Vektor
 - On-Process Block x On-Prozess Vektor \rightarrow On-Process Vektor
 - Off-Process Block x Off-Process Vektor \rightarrow On-Process Vektor
 - Kommunikation von Off-Process Vektoreinträgen zwischen Prozessen
 - \rightarrow Datenlokalität? Kommunikationsreduktion?





NumPY

- Implementierung basierend auf C und Python für Verwendung in Python
- Features:
 - N-Dimensionale Arrays
 - Tools zur Integration von C++- und Fortran-Code
 - Funktionen für lineare Algebra, FFT, Zufallszahlen
 - Universal Functions
 - Funktionen für Broadcasting
- Weitere Informationen: <http://www.numpy.org/>

NumPY: Universal Functions

```
import numpy as np
import timeit
import sys
```

```
def recip(v):
    w= np.empty(len(v))
    for i in range(len(v)):
        w[i] = 1.0/v[i]
    return w
```

```
if __name__=="__main__":
    np.random.seed(42)
    v = np.random.randint(1,1000, \
        size=1000000)
    if (sys.argv[1]=="recip"):
        recip(v)
    elif (sys.argv[1]=="norecip"):
        1.0/v
```

```
time python ./ufunc.py recip
real 0m2.387s
```

...

```
time python ./ufunc.py norecip
real 0m0.367s
```

...

- Python-Programm langsam (dynamisches Type-Checking, Interpreter etc.) ☹
- Ufuncs: „Vektorisierte“ Operationen auf ndarray-Elementen

NumPY: Broadcasting

```
import numpy as np
```

```
np.random.seed(42)
```

```
v=np.random.rand(10)
```

```
avg=v.mean()
```

```
print(v-avg)
```

- Spezifische Behandlung von Arrays unterschiedlicher Größe
- Einfach und effizient für schnelles Skripting
- Implementierung je nach Kontext fehleranfällig
- Beispiel-Code: Verschiebung von Zufallszahlen-Verteilung auf Null-Mittelwert

3. Löserpakete für Wissenschaftliche Anwendungen/PDEs

- **PETSc**
- **Trilinos**
- **SciPy**

PETSc/Tao (1)

- PETSC=Portable, Extensible Toolkit for Scientific Computation
- Unterstützt MPI, GPUs durch CUDA/OpenCL, MPI-GPU hybrid
- Funktionalitäten
 - Vektor- und Matrix-Operationen
 - Daten- und Gittermanagement (Strukturierte und unstrukturierte Gitter, Graphen und Netzwerke, ...)
 - Lineare Löser (Vorkonditionierer, Krylov-Methoden, Geometrisches Mehrgitterverfahren)
 - Nichtlineare Löser
 - Zeitschrittverfahren für ODEs
 - Optimierungsverfahren (Tao optimization library)
- Weitere Informationen:
<https://www.mcs.anl.gov/petsc/index.html>

PETSc/Tao (2)

- Software, die von PETSc (optional) genutzt wird:
 - BLAS, LAPACK
 - FFTW
 - ParMeTiS
 - ...
- Software, die PETSc nutzt:
 - MOOSE (Multiphysics Object-Oriented Simulation Environment)
 - SLEPc (Scalable Library for Eigenvalue Problems)
 - FEnICS (Finite-Elemente-Software zur Lösung von PDEs)
 - Firedrake (Finite-Elemente-Software zur Lösung von PDEs)
 - deal.II (Finite-Elemente-Software zur Lösung von PDEs)
 - PyClaw (Python-basierte Löser für hyperbolische PDEs)
 - ...

PETSc/Tao: Parallelisierung (1)

- Exzerpt von (~~11. Dez 2018~~ ~~25. Nov 2019~~ ~~09. Jan 2020~~ 14.10.2021):
<https://petsc.org/release/miscellaneous/threads/>

“The core PETSc team has come to the consensus that **pure MPI using neighborhood collectives and the judicious using of MPI shared memory** (for data structures that you may not wish to have duplicated on each MPI process due to memory constraints) **will provide the best performance for HPC simulation needs on current generation systems, next generation systems and exascale systems.** It is also a much simpler programming model then MPI + threads (leading to simpler code).”

→ Shared-Memory Parallelisierung möglich in Vec/Mat-Klassen oder Management einzelner PETSc-Objekte pro Thread, etc.

PETSc/Tao: Parallelisierung (2)

- Local-to-Global Mapping: Differenzierung zwischen lokaler und globaler Nummerierung von Vektoreinträgen, Zellen, Knoten, ...
 - Berücksichtigung von Ghost Points
- Data Management for Distributed Arrays (DMDA): Parallelisierungskonzept für konzeptionell reguläre Rechtecksgitter
 - Beispiel: siehe nächste Folien
- Unterstützung von Gather- und Scatter-Operationen, bspw. via PETSc-spezifische VecScatter*-Befehle



PETSc: Beispiel Poisson-Löser (1)

- Exzerpt aus NS-EOF (Inkompressible Strömungssimulation für Lehrzwecke); Nutzung von PETSc, V. 3.3
 - Details: P. Neumann, C. Kowitz, F. Schraner, D. Azarnykh. J. Parallel Distrib. Comput. 105:83-91, 2017
- Algorithmus pro Zeitschritt
 - Berechne partielle Ableitungen aus Impulsgleichungen und Prognose für neue Strömungsgeschwindigkeiten u, v, w exklusive Druckanteil
 - **Berechne Druck p zum nächsten Zeitschritt aus Poisson-Gleichung (7-Punkt-Stencil im Diskreten):**
$$\partial^2 p / \partial x^2 + \partial^2 p / \partial y^2 + \partial^2 p / \partial z^2 = r(u, v, w)$$
rechte Seite der Gleichung enthält Geschwindigkeitsprognosen
 - Korrigiere Geschwindigkeitsfeld mit Hilfe neuer Druckwerte

PETSc: Beispiel Poisson-Löser (1)

- Exzerpt aus NS-EOF (Inkompressible Strömungssimulation für Lehrzwecke); Nutzung von PETSc, V. 3.3
 - Details: P. Neumann, C. Kowitz, F. Schraner, D. Azarnykh. J. Parallel Distrib. Comput. 105:83-91, 2017
- Algorithmus pro Zeitschritt
 - Berechne partielle Ableitungen aus Impulsgleichungen und Prognose für neue Strömungsgeschwindigkeiten u, v, w exklusive Druckanteil
 - **Berechne Druck p zum nächsten Zeitschritt aus Poisson-Gleichung (7-Punkt-Stencil im Diskreten):**
$$p_{i-1jk} + p_{ij-1k} + p_{ijk-1} - 6p_{ijk} + p_{i+1jk} + p_{ij+1k} + p_{ijk+1} = r_{ijk}$$
rechte Seite der Gleichung enthält Geschwindigkeitsprognosen
 - Korrigiere Geschwindigkeitsfeld mit Hilfe neuer Druckwerte

PETSc: Beispiel Initialisierung (1)

```
KSPCreate(PETSC_COMM_WORLD,&_ksp);  
PCCreate(PETSC_COMM_WORLD,&_pc);
```

...

```
PetscErrorCode (*computeMatrix)(KSP, Mat, Mat, void*) = NULL;  
computeMatrix = computeMatrix3D;  
DMDACreate3d(  
    PETSC_COMM_WORLD, bx, by, bz, DMDA_STENCIL_STAR,  
    parameters.geometry.sizeX+2, parameters.geometry.sizeY+2,  
    parameters.geometry.sizeZ+2,  
    parameters.parallel.numProcessors[0],  
    parameters.parallel.numProcessors[1],  
    parameters.parallel.numProcessors[2],  
    1, 2,  
    parameters.parallel.sizes[0], parameters.parallel.sizes[1],  
    parameters.parallel.sizes[2],  
    &_da);
```

PETSc: Beispiel Initialisierung (1)

```
KSPCreate(PETSC_COMM_WORLD,&_ksp);  
PCCreate(PETSC_COMM_WORLD,&_pc);
```

...

```
PetscErrorCode (*computeMatrix)(KSP, Mat, void*) = NULL;  
computeMatrix = computeMatrix3D;
```

```
DMDACreate3d(  
    PETSC_COMM_WORLD, bx, by, bz,  
    parameters.geometry.sizeX+2, parameters.geometry.sizeY+2,  
    parameters.geometry.sizeZ+2,  
    parameters.parallel.numProcessors[0],  
    parameters.parallel.numProcessors[1],  
    parameters.parallel.numProcessors[2],  
    1, 2,  
    parameters.parallel.sizes[0], parameters.parallel.sizes[1],  
    parameters.parallel.sizes[2],  
    &_da);
```

Initialisierung von Krylov
Subspace Solver Context
und Vorkonditionierer

PETSc: Beispiel Initialisierung (1)

```
KSPCreate(PETSC_COMM_WORLD,&_ksp);  
PCCreate(PETSC_COMM_WORLD,&_pc);
```

...

```
PetscErrorCode (*computeMatrix)(KSP, Mat, Mat, void*) = NULL;  
computeMatrix = computeMatrix3D;
```

```
DMDACreate3d(  
    PETSC_COMM_WORLD, bx, by, bz, DMDA_STENCIL_STAR,  
    parameters.geometry.sizeX+2, parameters.geometry.sizeY+2,  
    parameters.geometry.sizeZ+2,  
    parameters.parallel.numProcessors[0],  
    parameters.parallel.numProcessors[1],  
    parameters.parallel.numProcessors[2],  
    1, 2,  
    parameters.parallel.sizes[0], parameters.parallel.sizes[1],  
    parameters.parallel.sizes[2],  
    &_da);
```

Setze Funktionspointer,
um PETSc's
Matrixinitialisierung später
durchführen zu können

PETSc: Beispiel Initialisierung (1)

```
KSPCreate(PETSC_COMM_WORLD,  
PCCreate(PETSC_COMM_WORLD,  
...  
PetscErrorCode (*computeMatrix)(&  
computeMatrix = computeMatrix,&  
DMDACreate3d(  
    PETSC_COMM_WORLD, bx, by, bz, DMDA_STENCIL_STAR,  
    parameters.geometry.sizeX+2, parameters.geometry.sizeY+2,  
    parameters.geometry.sizeZ+2,  
    parameters.parallel.numProcessors[0],  
    parameters.parallel.numProcessors[1],  
    parameters.parallel.numProcessors[2],  
    1, 2,  
    parameters.parallel.sizes[0], parameters.parallel.sizes[1],  
    parameters.parallel.sizes[2],  
    &_da);
```

Initialisierung der
eigentlichen PETSc-
Datenstruktur

PETSc: Beispiel Initialisierung (1)

```
KSPCreate(PETSC_COMM_WORLD,&_ksp);  
PCCreate(PETSC_COMM_WORLD,&_pc);
```

...

```
PetscErrorCode (*computeMatrix)(KSP  
computeMatrix = computeMatrix3D;
```

Parameter zur
Randbehandlung

```
DMDACreate3d(  
    PETSC_COMM_WORLD, bx, by, bz, DMDA_STENCIL_STAR,  
    parameters.geometry.sizeX+2, parameters.geometry.sizeY+2,  
    parameters.geometry.sizeZ+2,  
    parameters.parallel.numProcessors[0],  
    parameters.parallel.numProcessors[1],  
    parameters.parallel.numProcessors[2],  
    1, 2,  
    parameters.parallel.sizes[0], parameters.parallel.sizes[1],  
    parameters.parallel.sizes[2],  
    &_da);
```


PETSc: Beispiel Initialisierung (1)

```
KSPCreate(PETSC_COMM_WORLD,&_ksp);  
PCCreate(PETSC_COMM_WORLD,&_pc);
```

...

```
PetscErrorCode (*computeMatrix)(KSP, Mat, Mat, void*) = NULL;  
computeMatrix = computeMatrix3D;
```

```
DMDACreate3d(  
    PETSC_COMM_WORLD, bx, by, bz, DMDA_STENCIL_STAR,  
    parameters.geometry.sizeX+2, parameters.geometry.sizeY+2,  
    parameters.geometry.sizeZ+2,  
    parameters.parallel.numProcessors,  
    parameters.parallel.numProcessors,  
    parameters.parallel.numProcessors,  
    1, 2,  
    parameters.parallel.sizes[0], parameters.parallel.sizes[1],  
    parameters.parallel.sizes[2],  
    &_da);
```

Stencilformat für
Matrixzeilen-Beschreibung;
legt Ghost Points und
Nachrichtentransfer fest:
STAR: weniger Nachrichten
BOX: mehr Nachrichten

PETSc: Beispiel Initialisierung (1)

```
KSPCreate(PETSC_COMM_WORLD,&_ksp);  
PCCreate(PETSC_COMM_WORLD,&_pc);
```

...

```
PetscErrorCode (*computeMatrix)(KSP  
computeMatrix = computeMatrix3D;
```

```
DMDACreate3d(  
  PETSC_COMM_WORLD, bx, by, bz,
```

```
  MDA_STENCIL_STAR,  
  parameters.geometry.sizeX+2, parameters.geometry.sizeY+2,  
  parameters.geometry.sizeZ+2,
```

```
  parameters.parallel.numProcessors[0],
```

```
  parameters.parallel.numProcessors[1],
```

```
  parameters.parallel.numProcessors[2],
```

```
  1, 2,
```

```
  parameters.parallel.sizes[0], parameters.parallel.sizes[1],
```

```
  parameters.parallel.sizes[2],
```

```
  &_da);
```

Globale Größe des 3D-Systems

PETSc: Beispiel Initialisierung (1)

```
KSPCreate(PETSC_COMM_WORLD,&_ksp);  
PCCreate(PETSC_COMM_WORLD,&_pc);
```

...

```
PetscErrorCode (*computeMatrix)(KSP  
computeMatrix = computeMatrix3D;
```

```
DMDACreate3d(  
  PETSC_COMM_WORLD, bx, by, bz, P
```

```
  parameters.geometry.sizeX+2, parameters.geometry.sizeY+2,  
  parameters.geometry.sizeZ+2,
```

```
  parameters.parallel.numProcessors[0],
```

```
  parameters.parallel.numProcessors[1],
```

```
  parameters.parallel.numProcessors[2],
```

```
  1, 2,
```

```
  parameters.parallel.sizes[0], parameters.parallel.sizes[1],
```

```
  parameters.parallel.sizes[2],
```

```
  &_da);
```

Anzahl der MPI-Prozesse
pro Dimension

PETSc: Beispiel Initialisierung (1)

```
KSPCreate(PETSC_COMM_WORLD,&_ksp);  
PCCreate(PETSC_COMM_WORLD,&_pc);
```

...

```
PetscErrorCode (*computeMatrix)(KSP  
computeMatrix = computeMatrix3D;
```

```
DMDACreate3d(  
  PETSC_COMM_WORLD, bx, by, bz, DMDA_STENCIL_STAR,  
  parameters.geometry.sizeX+2, parameters.geometry.sizeY+2,  
  parameters.geometry.sizeZ+2,  
  parameters.parallel.numProcessors[0],  
  parameters.parallel.numProcessors[1],  
  parameters.parallel.numProcessors[2],  
  1, 2,  
  parameters.parallel.sizes[0], parameters.parallel.sizes[1],  
  parameters.parallel.sizes[2],  
  &_da);
```

Anzahl der Variablen pro
Zelle und Stencilbreite

PETSc: Beispiel Initialisierung (1)

```
KSPCreate(PETSC_COMM_WORLD,&_ksp);
PCCreate(PETSC_COMM_WORLD,&_pc);
...
PetscErrorCode (*computeMatrix)(KSP, PetscInt L;
computeMatrix = computeMatrix3D;
DMDACreate3d(
    PETSC_COMM_WORLD, bx, by, bz, DMDA_STENCIL_STAR,
    parameters.geometry.sizeX+2, parameters.geometry.sizeY+2,
    parameters.geometry.sizeZ+2,
    parameters.parallel.numProcessors[0],
    parameters.parallel.numProcessors[1],
    parameters.parallel.numProcessors[2],
    1, 2,
    parameters.parallel.sizes[0], parameters.parallel.sizes[1],
    parameters.parallel.sizes[2],
    &_da);
```

Arrays mit Anzahl der Zellen

PETSc: Beispiel Initialisierung (1)

```
KSPCreate(PETSC_COMM_WORLD,&_ksp);  
PCCreate(PETSC_COMM_WORLD,&_pc);
```

...

```
PetscErrorCode (*computeMatrix)(KSP  
computeMatrix = computeMatrix3D;
```

```
DMDACreate3d(  
  PETSC_COMM_WORLD, bx, by, bz,
```

```
  parameters.geometry.sizeX+2, parameters.geometry.sizeY+2,  
  parameters.geometry.sizeZ+2,  
  parameters.parallel.numProcessors[0],  
  parameters.parallel.numProcessors[1],  
  parameters.parallel.numProcessors[2],  
  1, 2,  
  parameters.parallel.sizes[0], parameters.parallel.sizes[1],  
  parameters.parallel.sizes[2],  
  &_da);
```

Distributed-Array Objekt,
welches durch diese
Methode initialisiert wird

PETSc: Beispiel Initialisierung (2)

```
...  
PCSetType(_pc,PCILU);  
PCFactorSetLevels(_pc,1);  
KSPSetPC(_ksp,_pc);
```

Definition des
Vorkonditionierers

```
...  
KSPSetFromOptions(_ksp);  
KSPSetInitialGuessNonzero(_ksp,PETSC_TRUE);  
KSPSetUp(_ksp);
```

Konfiguration weiterer Optionen
über Kommandozeile

Start von Nicht-Null-Lösung für
Druckfeld

```
...
```

Anlegen der Löserinstanz

PETSc: Beispiel computeMatrix3D(...) (1)

```
for (k = limitsZ[0]; k < limitsZ[1]; k++){  
    for (j = limitsY[0]; j < limitsY[1]; j++){  
        for (i = limitsX[0]; i < limitsX[1]; i++){
```

Schleife über alle Zellen
und Initialisierung der
Matrix-Zeile

```
        ...  
        stencilValues[1] = 2.0/(dx_L *(dx_L+dx_R)); // left  
        stencilValues[0] = 2.0/(dx_R *(dx_L+dx_R)); // right  
        stencilValues[2] = 2.0/(dx_T *(dx_T+dx_Bo)); // top  
        stencilValues[3] = 2.0/(dx_Bo*(dx_T+dx_Bo)); // bottom  
        stencilValues[4] = 2.0/(dx_B *(dx_B+dx_F)); // back  
        stencilValues[5] = 2.0/(dx_F *(dx_B+dx_F)); // front  
        stencilValues[6] = -2.0/(dx_R*dx_L)-2.0/(dx_T*dx_Bo)-  
            2.0/(dx_F*dx_B); // center
```


PETSc: Beispiel computeMatrix3D(...) (2)

```
// Definition of positions. Order must correspond to values
column[0].i = i+1; column[0].j = j;  column[0].k = k;
column[1].i = i-1; column[1].j = j;  column[1].k = k;
column[2].i = i;  column[2].j = j+1; column[2].k = k;
column[3].i = i;  column[3].j = j-1; column[3].k = k;
column[4].i = i;  column[4].j = j;  column[4].k = k+1;
column[5].i = i;  column[5].j = j;  column[5].k = k-1;
column[6].i = i;  column[6].j = j;  column[6].k = k;
```

```
MatSetValuesStencil(A, 1, &row, 7, column, stencilValues,
INSERT_VALUES);
```

...

Alle Sonderfälle:
Randbehandlung
etc.

Definition der Indexeinträge in
Zeile und Setzen aller
Informationen in PETSc

PETSc: Beispiel computeMatrix3D(...) (3)

```
MatAssemblyBegin(A,MAT_FINAL_ASSEMBLY);  
MatAssemblyEnd(A,MAT_FINAL_ASSEMBLY);
```

Assemblierung der
globalen Matrix-
Information

```
MatNullSpace nullspace;  
MatNullSpaceCreate(PETSC_COMM_WORLD,PETSC_TRUE,  
0,0,&nullspace);  
MatSetNullSpace(A,nullspace);  
MatNullSpaceDestroy(&nullspace);
```

Definition des Kerns
(=NullSpace) der Matrix:
Konstante Druckfelder

PETSc: Beispiel solve()

```
KSPSetComputeRHS(_ksp, computeRHS3D, &_ctx);  
KSPSetComputeOperators(_ksp, computeMatrix3D, &_ctx);  
KSPSolve(_ksp, PETSC_NULL, _x);
```

Definition von
rechter Seite
und Matrix

Endlich wird gerechnet 😊

```
// Then extract the information
```

```
PetscScalar ***array;  
DMDAVecGetArray(_da, _x, &array);
```

Zurückschreiben der Lösung in
Simulations-Datenstruktur

```
for (int k = _firstZ; k < _firstZ + _lengthZ; k++){  
    for (int j = _firstY; j < _firstY + _lengthY; j++){  
        for (int i = _firstX; i < _firstX + _lengthX; i++){  
            pressure.getScalar(i - _firstX + _offsetX, j - _firstY + _offsetY,  
                k - _firstZ + _offsetZ) = array[k][j][i];  
        }  
    }  
}
```

...

Trilinos

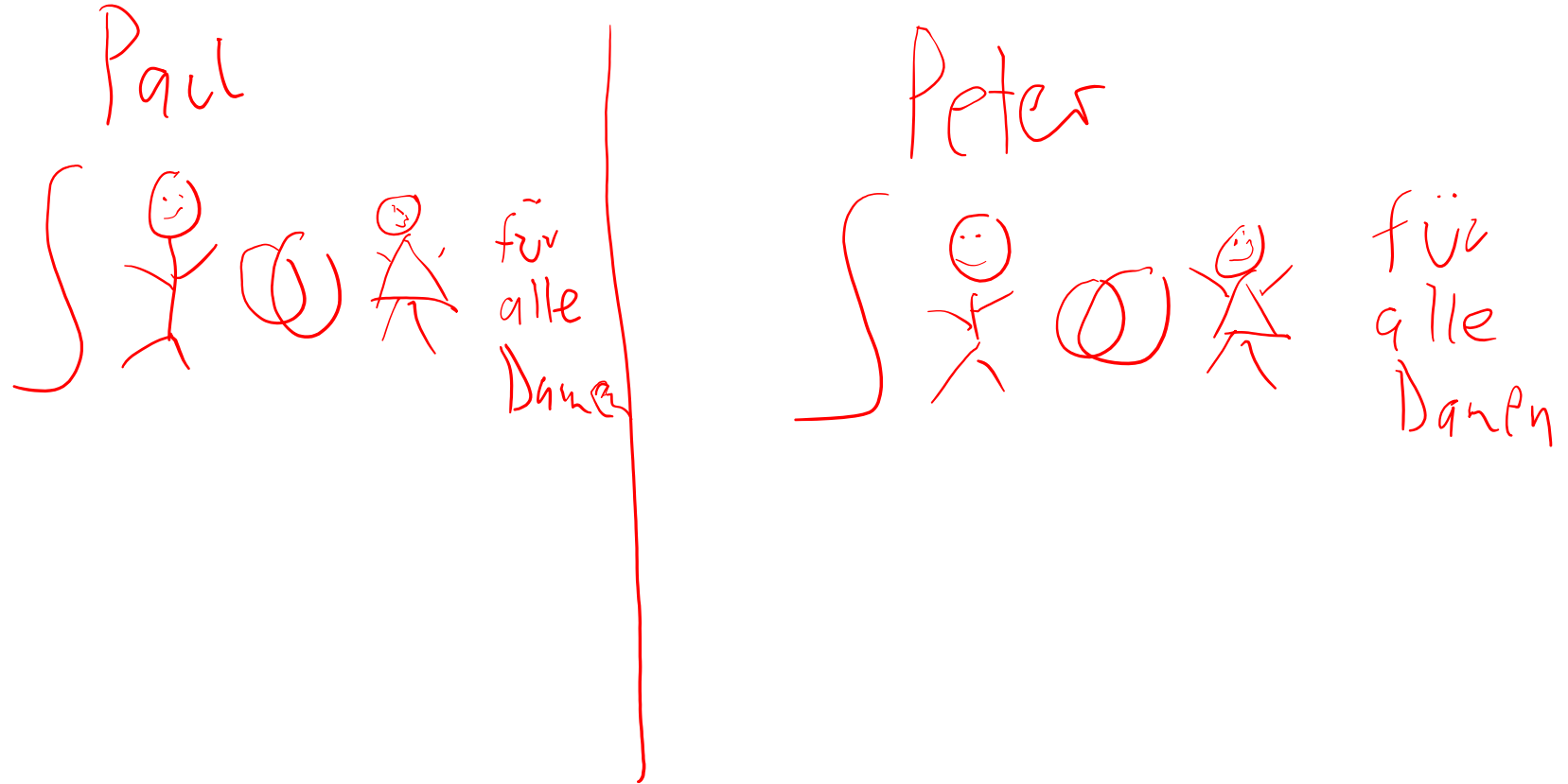
- Pakete-Sammlung zur Lösung von „large-scale, complex multi-physics engineering and scientific problems“
- Weitere Informationen: <https://trilinos.org/>
- Einteilung der Pakete in „Capability Areas“:
 - User Experience
 - Parallel Programming Environments
 - Framework & Tools
 - Software Engineering Technologies and Integration
 - I/O Support
 - Meshes, Geometry & Load Balancing
 - Discretization
 - Scalable Linear Algebra (Pakete Epetra, Tpetra, etc.; Paket Teuchos wrappt BLAS/LAPACK)
 - Linear & Eigen Solvers
 - Embedded Nonlinear Analysis Tools

- Nutzt NumPy
- Funktionalitäten
 - Numerische Integration (Quadratur)
 - Optimierung
 - Interpolation
 - FFT
 - Symbolisches Rechnen
 - Visualisierung und Datenanalyse (Pandas)
 - ...
- Weitere Informationen: <https://www.scipy.org/>

4. Software-Frameworks

- Finite Elemente Frameworks
 - **FEniCS**
 - **deal.II**
- Andere Frameworks
 - DUNE
 - DUNE=Distributed and Unified Numerics Environment
 - Basiert auf generischer Programmierung (Expression Templates)
 - Kernmodule:
 - dune-common (allg. Infrastruktur)
 - dune-geometry
 - dune-grid
 - dune-istl (iterative solver template library)
 - dune-local functions (Ansatzfunktionen)
 - Numerische Strömungsmechanik: OpenFOAM
 - Wettersimulation: OpenIFS

Finite Elemente Methode



Finite Elemente Methode



$$-\frac{d^2 u}{dx^2} = f(x), \quad u(0) = u(1) = 0$$

$$-\int_0^1 \frac{d^2 u}{dx^2} \cdot v \, dx = \int_0^1 f \cdot v \, dx \quad \forall v \in V$$

Testfunktionen

$$\int_0^1 \frac{du}{dx} \cdot \frac{dv}{dx} \, dx = \int_0^1 f \cdot v \, dx \quad \forall v \in V$$

Schwache Formulierung

$$u(x) := \sum_i \alpha_i w_i(x)$$

Ansatzfunktionen

$$\sum_i \alpha_i \int_0^1 \frac{dw_i}{dx} \cdot \frac{dv}{dx} \, dx = \int_0^1 f \cdot v \, dx \quad \forall v \in V$$

- Ziel: „creating easy, intuitive, efficient, and flexible software for solving partial differential equations (PDEs) using finite element methods“
 - H.P. Langtangen, A. Logg. Solving PDEs in Python. The FEniCS Tutorial I, Springer, 2016
 - Weitere Informationen: <https://fenicsproject.org/>
- C++-Backend (DOLFIN)
- C++- und Python-Schnittstellen
- Schnittstellen zu (u.a.) PETSc, Trilinos, ParMETIS
- Features:
 - Automatische Generierung von Basisfunktionen
 - Automatische Evaluierung von schwachen Formulierungen
 - Automatische Finite-Element-Assemblierung
 - Automatische adaptive Fehlerkontrolle
 - Siehe: https://fenicsproject.org/pub/course/lectures/2017-nordic-phdcourse/lecture_01_fenics_introduction.pdf
- FEniCS-HPC: Hybrid MPI/OpenMP-parallelisierte FEniCS-Komponenten

FEniCS

- Ziel: „creating easy, intuitive, efficient, and flexible software for solving partial differential equations (PDEs) using finite element methods“
 - H.P. Langtangen, A. Logg. Solving PDEs in Python. The FEniCS Tutorial I, Springer, 2016
 - Weitere Informationen: <https://fenicsproject.org/>
- C++-Backend (DOLFIN)
- C++- und Python-Schnittstellen
- Schnittstellen zu (u.a.) PETSc, Trilinos, ParMETIS
- Features:
 - Automatische Generierung von Code
 - Automatische Evaluierung
 - Automatische Finite-Elemente-Mesh-Generierung
 - Automatische adaptive Mesh-Refinement
 - Siehe: https://fenicsproject.org/PhDcourse/lecture_01_fenics_introduction.pdf
- FEniCS-HPC: Hybrid MPI/OpenMP-parallelisierte FEniCS-Komponenten

u.a. Turbulente
Strömungssimulation auf 5000
Rechenkernen mit PETSc-Backend

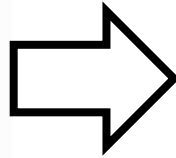
FEniCS: Beispiel Poisson-Gleichung (1)

- Weitere Informationen:

<https://fenicsproject.org/olddocs/dolfin/1.3.0/python/demo/documented/poisson/python/documentation.html>

- Transformation in schwache Formulierung:

$$\begin{aligned} -\nabla^2 u &= f && \text{in } \Omega, \\ u &= 0 && \text{on } \Gamma_D, \\ \nabla u \cdot n &= g && \text{on } \Gamma_N. \end{aligned}$$



$$\begin{aligned} a(u, v) &= L(v) \quad \forall v \in V, \\ a(u, v) &= \int_{\Omega} \nabla u \cdot \nabla v \, dx, \\ L(v) &= \int_{\Omega} f v \, dx + \int_{\Gamma_N} g v \, ds. \end{aligned}$$

FEniCS: Beispiel Poisson-Gleichung (2)

```
# https://fenicsproject.org/olddocs/dolfin/1.3.0/python/
```

```
# _downloads/demo_poisson.py
```

```
...
```

```
mesh = UnitSquareMesh(32, 32)
```

Geometrie: Einheits-
quadrat, 32x32 Elemente

```
V = FunctionSpace(mesh, "Lagrange", 1)
```

Lagrange-Elemente
erster Ordnung

```
# Define Dirichlet boundary (x = 0 or x = 1)
```

```
def boundary(x):
```

```
    return x[0] < DOLFIN_EPS or x[0] > 1.0 -  
    DOLFIN_EPS
```

```
u0 = Constant(0.0)
```

```
bc = DirichletBC(V, u0, boundary)
```

FEniCS: Beispiel Poisson-Gleichung (3)

```
# Define variational problem
```

```
u = TrialFunction(V)
```

```
v = TestFunction(V)
```

```
f = Expression("10*exp(-(pow(x[0] - 0.5, 2) + pow(x[1] - 0.5, 2)) / 0.02)")
```

```
g = Expression("sin(5*x[0])")
```

```
a = inner(grad(u), grad(v))*dx
```

```
L = f*v*dx + g*v*ds
```

```
# Compute solution
```

```
u = Function(V)
```

```
solve(a == L, u, bc)
```

Ansatzfunktionen

Testfunktionen

Löse:

$$a(u, v) = L(v) \quad \forall v \in V,$$

$$a(u, v) = \int_{\Omega} \nabla u \cdot \nabla v \, dx,$$

$$L(v) = \int_{\Omega} f v \, dx + \int_{\Gamma_N} g v \, ds.$$

- deal.II=successor of Differential Equations Analysis Library
- C++-Bibliothek
- Features:
 - Lokal verfeinerte Gitter, adaptive Verfeinerung mit Fehlerschätzern
 - h-,p-,hp-Verfeinerung
 - Continuous vs. Discontinuous elements
 - Eigenständige Bibliothek für lineare Algebra
 - Schnittstellen zu Trilinos, PETSc, METIS
- Weitere Informationen: <https://www.dealii.org/>

Zusammenfassung

- Mathematische Bibliotheken für numerische lineare Algebra, zur Lösung von PDEs/ODEs, Datenanalyse, etc.
- Numerische lineare Algebra
 - BLAS für grundlegende Operationen
 - LAPACK/Scalapack für Gleichungssystemlöser, Eigenwertprobleme, etc.
 - Exkurs: Dünnbesetzte Matrizen versus HPC-Implementierung
- Löserpakete für PDEs (Beispiel: PETSc)
 - Nutzung von Low-Level Bibs (wie bspw. BLAS)
 - Breite Funktionalität: Zeitschrittverfahren, (nicht-)lineare Löser, Gebietszerlegungen, etc.
- High-Level Frameworks (Beispiel: FEniCS): Anwendungs- und problemnahe Formulierung und Implementierung