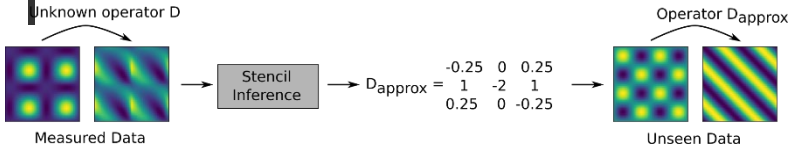
Hochleistungsrechnen
Mathematische Bibliotheken

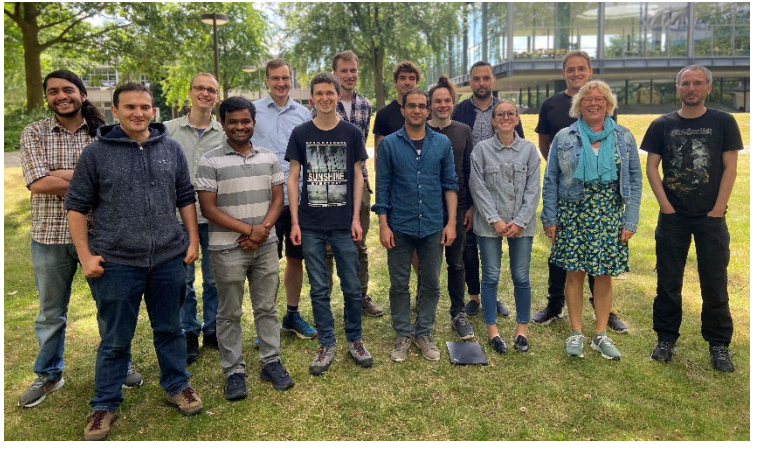
Skriptversion 17.01.2023

Prof. Dr. Philipp Neumann
Helmut-Schmidt-Universität/Universität der Bundeswehr Hamburg

Hiwi/Promotion?



- Themenbereiche
  - Codeoptimierung, Performanceanalysen, Energieeffizienzanalysen
  - Gekoppelte Mehrskalensimulation
  - Maschinelles Lernen <-> Simulation
  - Hardware-gewahre Implementierungen
- Anforderungen
  - Programmierkenntnisse (C/C++/Python)
  - Interesse an Simulation, HPC und Code-Gefrickel ☺
- Rahmenbedingungen (Hiwi)
  - Keine festen Anwesenheitszeiten, flexible Arbeitsbedingungen
  - Start: jederzeit
  - PhD:
- Bei Interesse: Philipp Neumann, philipp.neumann@hsu-hh.de

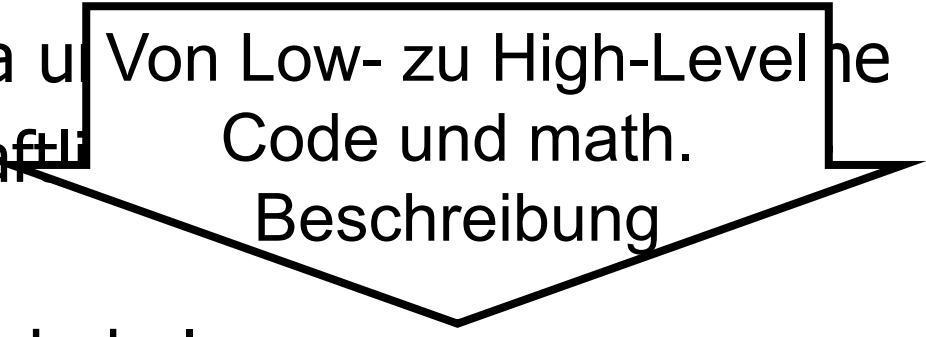


Überblick

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 u
3. Löserpakete für Wissenschaftli
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 → Maschinelles Lernen → Lineare Algebra“, siehe bspw. Götz und Anzt (2018) für Prä-Konditionierung

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

Effiziente Hardware-gewahre Realisierung

- 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** (successor of LINPACK)**, Scalapack**
- **NumPy**
- Hypre: Scalable Linear Solvers and Multigrid Methods
  - Weitere Informationen: https://computing.llnl.gov/projects/hypre-scalable-linear-solvers-multigrid-methods
- ParMETIS
  - MPI-parallele Bibliothek zur Partitionierung unstrukturierter Graphen
  - Weitere Informationen: https://github.com/KarypisLab/ParMETIS
- FFTW (Fastest Fourier Transform in the West)
  - Weitere Informationen: http://www.fftw.org/benchfft/

BLAS

- BLAS=Basic Linear Algebra Subprograms, eingeteilt in 3 Levels
  - Level 1: Vektor-Operationen
  - Level 2: Vektor-Matrix-Operationen
  - Level 3: Matrix-(Matrix-)Operationen
    → beinhaltet (GEMM=general matrix-matrix multiply)

  → 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
- Relevante Implementierungen
  - OpenBLAS
  - Intel oneAPI (previously: Math Kernel library (MKL)), Nvidia cuBLAS
  - BLIS (BLAS-like Library Instantiation Software)
  - Automatically tuned linear algebra software (ATLAS)
- Weitere Informationen: www.netlib.org/blas/

BLAS

- BLAS=Basic Linear Algebra Subprograms, eingeteilt in 3 Levels
  - Level 1: Vektor-Operationen
  - Level 2: Vektor-Matrix-Operationen
  - Level 3: Matrix-(Matrix-)Operationen
    → beinhaltet (GEMM=general matrix-matrix multiply)

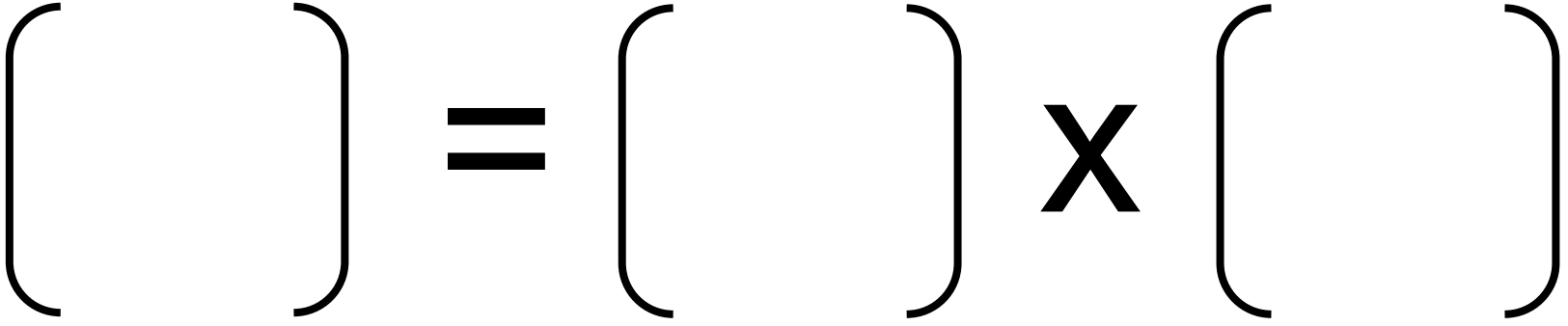
  → grundlegende Bausteine aus der linearen Algebra
    → Skalierung eines Vektors mit Konstante
    → Matrix-Vektor-Multiplikation
    → Matrix-Matrix-M…multiply)
    → Spezialisierunge…andmatrizen
- Relevante Implen…
  - OpenBLAS
  - Intel oneAPI (pre…uBLAS
  - BLIS (BLAS-like Libra…stantiation Software)
  - Automatically tuned linear algebra software (ATLAS)
- Weitere Informationen: www.netlib.org/blas/

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

ATLAS

- Ziel: Automatische Optimierung für spezifische Hardware
- Beispiel: Matrix-Matrix-Multiplikation

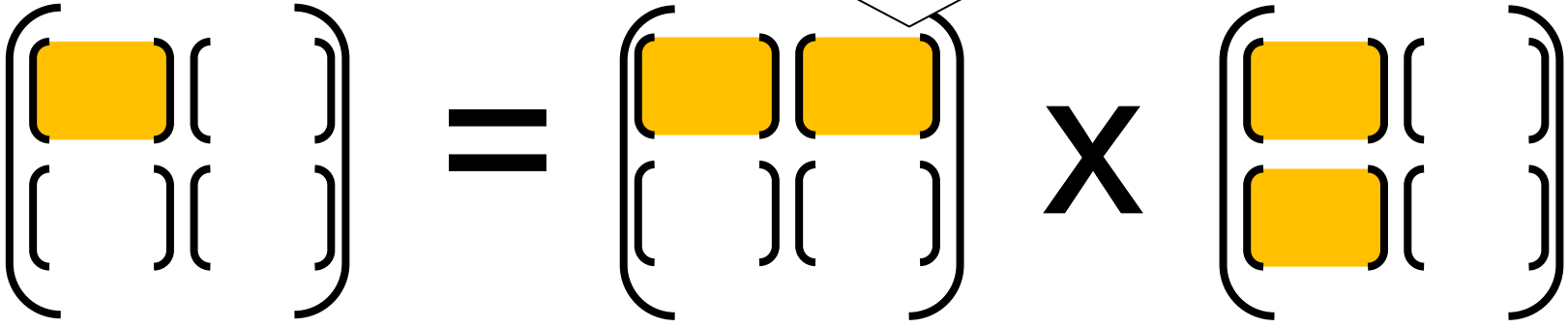
  → Optimales Cache-Blocking
- Weitere Informationen:
  - http://www.netlib.org/lapack/lawnspdf/lawn131.pdf
  - R.C. Whaley, J.J. Dongarra. Automatically tuned linear algebra software. SC ’98 proceedings, pp. 1-27, 1998



ATLAS

- Ziel: Automatische Optimierung für spezifische Hardware
- Beispiel: Matrix-Matrix-Multiplikation
  → Optimales Cache-Blocking
- Weitere Informatione
  - http://www.netlib.or pdf
  - R.C. Whaley, J.J. Dor linear algebra software. SC 1998

Tuning der Block-größe zur Verhinderung von Cache Misses

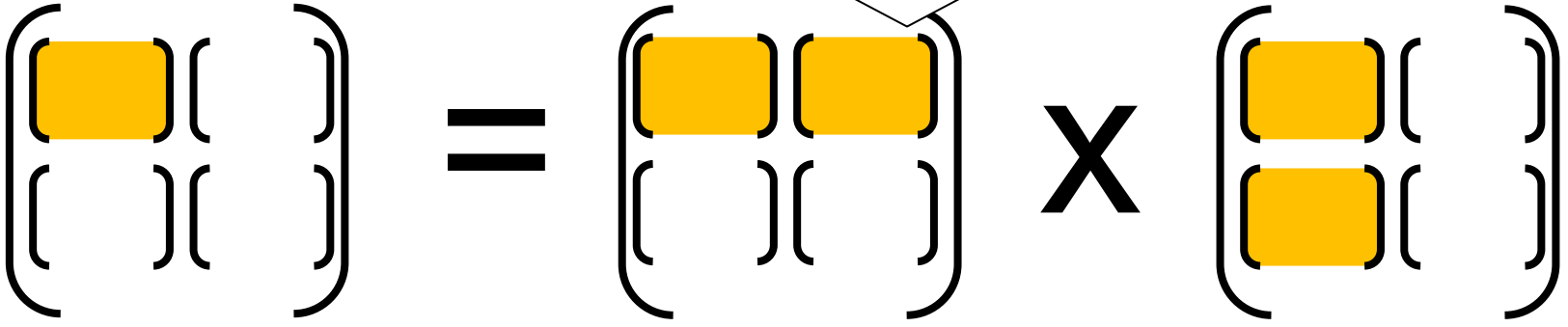


ATLAS

- Ziel: Automatische Optimierung ware
- Beispiel: Matrix-Matrix-Multiplika
  → Optimales Cache-Blocking
- Weitere Informatione
  - http://www.netlib.or pdf
  - R.C. Whaley, J.J. Dor linear algebra software. SC 1998

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
- OpenBLAS[1]: „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 MKL[2]: „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, as of 06/2020

2https://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/)

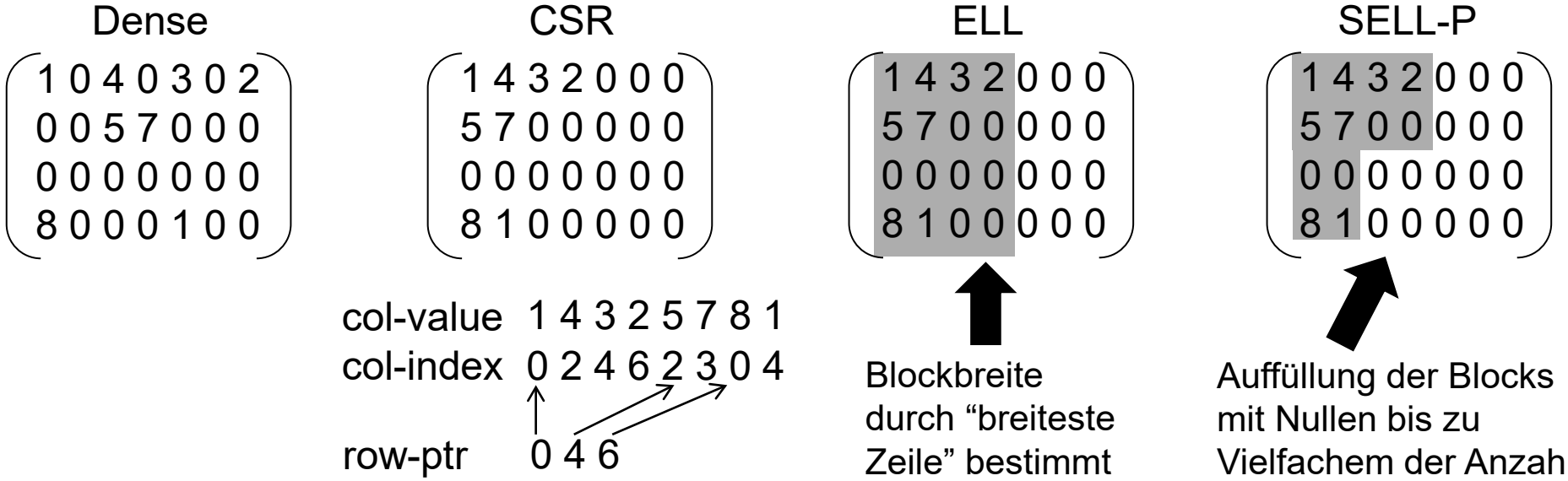
ScaLAPACK

- 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 line… distributed memory computers — desig… Computer Physics Communications 97(1…
- Ähnliche Entwicklungen:
  - PLASMA=Parallel Linear Algebra Softwa… Architectures (http://icl.cs.utk.edu/plasma/so…are/)
  - 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)

Dense

$$\begin{pmatrix} 1 & 0 & 4 & 0 & 3 & 0 & 2 \\ 0 & 0 & 5 & 7 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 8 & 0 & 0 & 0 & 1 & 0 & 0 \end{pmatrix}$$

GPU ☹
CSR

$$\begin{pmatrix} 1 & 4 & 3 & 2 & 0 & 0 & 0 \\ 5 & 7 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 8 & 1 & 0 & 0 & 0 & 0 & 0 \end{pmatrix}$$

col-value 1 4 3 2 5 7 8 1
col-index 0 2 4 6 2 3 0 4
row-ptr 0 4 6

GPU ☺
ELL

$$\begin{pmatrix} 1 & 4 & 3 & 2 & 0 & 0 & 0 \\ 5 & 7 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 8 & 1 & 0 & 0 & 0 & 0 & 0 \end{pmatrix}$$

Blockbreite durch “breiteste Zeile” bestimmt

GPU ☺
SELL-P

$$\begin{pmatrix} 1 & 4 & 3 & 2 & 0 & 0 & 0 \\ 5 & 7 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 8 & 1 & 0 & 0 & 0 & 0 & 0 \end{pmatrix}$$

Auffüllung der Blocks mit Nullen bis zu Vielfachem der Anzahl der Threads pro Zeile (hier: #threads=2)

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 → On-Process Vektor

      Off-Process Block x Off-Process Vektor → On-Process Vektor
  - Kommunikation von Off-Process Vektoreinträgen zwischen Prozessen

    → 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
- Simpel 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://petsc.org/release/

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)

- Native **MPI support**, OpenMP (more or less) recently
- PETSc GPU Roadmap, https://petsc.org/release/overview/gpu_roadmap/

PETSc code will include full implementations of vector and matrix operations (as well as other select operations) using each of:

Programming Model	Supporting Package	`Vec` Status	`Mat` Status	Supported GPU types
CUDA	cuBLAS/cuSparse	SUPPORTED	SUPPORTED	NVIDIA GPUs
HIP	Rocm	SUPPORTED	IN DEVELOPMENT	AMD GPUs
SYCL	MKL	NOT YET SUPPORTED	NOT YET SUPPORTED	NOT YET SUPPORTED
OpenCL	ViennaCL	SUPPORTED	SUPPORTED	NVIDIA, AMD, INTEL GPUs
Kokkos	Kokkos and Kokkos-Kernels	SUPPORTED	SUPPORTED	NVIDIA, AMD, INTEL GPUs

PETSc/Tao: Parallelisierung (2)

- Local-to-Global Mapping: Differenzierung zwischen lokaler und globaler Nummierung 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. Schranner, 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. Schranner, 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, [illegible] Mat, void*) = NULL;
computeMatrix = computeMatrix3D;
DMDACreate3d(
    PETSC_COMM_WORLD, bx, by, b
    parameters.geometry.sizeX+2, paran
    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 = computeM
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)(KS
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);
```

Parameter zur Randbehandlung

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], parame
   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)(KS
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);
```

Globale Größe des 3D-Systems

PETSc: Beispiel Initialisierung (1)

```
KSPCreate(PETSC_COMM_WORLD,&_ksp);
PCCreate(PETSC_COMM_WORLD,&_pc);
…
PetscErrorCode (*computeMatrix)(KS
computeMatrix = computeMatrix3D;
DMDACreate3d(
   PETSC_COMM_WORLD, bx, by, bz, D      A_STENCIL_STAR,
   parameters.geometry.sizeX+2, parame      s.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)(KS
computeMatrix = computeMatrix3D;
DMDACreate3d(
   PETSC_COMM_WORLD, bx, by,  DMDA_STENCIL_STAR,
   parameters.geometry.sizeX+2,  rameters.geometry.sizeY+2,
   parameters.geometry.siz  +2,
   parameters.parall  umProcessors[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)(KS                    L;
computeMatrix = computeMatrix3D;
DMDACreate3d(
   PETSC_COMM_WORLD, bx, by,      MDA_STENCIL_STAR,
   parameters.geometry.sizeX+2,      eters.geometry.sizeY+2,
   parameters.geometry.sizeZ+2
   parameters.parallel.numP      cessors[0],
   parameters.parallel.n      Processors[1],
   parameters.paralle      .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)(KS
computeMatrix = computeMatrix3D;
DMDACreate3d(
    PETSC_COMM_WORLD, bx, by, bz,  _STENCIL_STAR,
    parameters.geometry.sizeX+2, para  s.geometry.sizeY+2,
    parameters.geometry.sizeZ+2
    parameters.parallel.numPr  sors[0],
    parameters.parallel.nu  rocessors[1],
    parameters.parallel  umProcessors[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);
…
KSPSetFromOptions(_ksp);
KSPSetInitialGuessNonzero(_ksp,PETSC_TRUE);
KSPSetUp(_ksp);
…
```

Definition des Vorkonditionierers

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++){
            …
            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
```

Schleife über alle Zellen und Initialisierung der Matrix-Zeile

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);

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

Assemblierung der globalen Matrix-Information

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

PETSc: Beispiel solve()

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

// Then extract the information
PetscScalar ***array;
DMDAVecGetArray(_da, _x, &array);

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];
}}}
…
```

Definition von rechter Seite und Matrix

Endlich wird gerechnet ☺

Zurückschreiben der Lösung in Simulations-Datenstruktur

PETSc: Beispiel solve()

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

// Then extract the information
PetscScalar ***array;
DMDAVecGetArray(_da, _x, &array);

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];
```

Definition von rechter Seite und Matrix

Endlich wird gerechnet ☺

Zurückschreiben der Lösung in Simulations-Datenstruktur

Fazit: Erfolg = Expertenwissen + Interfaces + Programmierung

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

SciPy

- 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 Prorammierung (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

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

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

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

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

FEniCS

- Ziel: „creating easy, intuitive, efficient, and flexible software for solving partial differential equations (PDEs) using finite element methods”
  - H.P. Langtangen, A. Loog. 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. Loog. 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 Generie
  - Automatische Evaluie
  - Automatische Finite-E
  - Automatische adaptiv
  - Siehe: https://fenicsp phdcourse/lecture_01_fenics_intro

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

- FEniCS-HPC: Hybrid MPI/OpenMP-parallelisierte FEniCS-Komponenten

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:

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

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

FEniCS: Beispiel Poisson-Gleichung (2)

```
# https://fenicsproject.org/olddocs/dolfin/1.3.0/python/
# _downloads/demo_poisson.py
…
mesh = UnitSquareMesh(32, 32)
V = FunctionSpace(mesh, "Lagrange", 1)

# 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)
```

Geometrie: Einheits-quadrat, 32x32 Elemente

Lagrange-Elemente erster Ordnung

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:

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

$$a(u,v) = \int_\Omega \nabla u \cdot \nabla v \,\mathrm{d}x,$$

$$L(v) = \int_\Omega f v \,\mathrm{d}x + \int_{\Gamma_N} g v \,\mathrm{d}s.$$

deal.II

- 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/

Wer ist Super-HPCler? ☺

A Nase anfassen

B Klatschen

C Augen zukneifen

D Winken

Wer ist Super-HPCler? ☺

A Nase anfassen

B Klatschen

C Augen zukneifen

D Winken

Wie funktioniert das Speicherformat ELLPACK?

A Komprimieren der Matrix in einen großen Block, Padding mit Nullen

B Komprimieren der Matrix in verschieden große Blöcke, Padding mit Nullen

C Indirekte Indexierung der Matrixeinträge

D Das ist ein ELLbogen-SchutzPACK beim Skaten

Wer ist Super-HPCler? ☺

A Nase anfassen

B Klatschen

C Augen zukneifen

D Winken

Wie funktioniert das Speicherformat ELLPACK?

A Komprimieren der Matrix in einen großen Block, Padding mit Nullen

B Komprimieren der Matrix in verschieden große Blöcke, Padding mit Nullen

C Indirekte Indexierung der Matrixeinträge

D Das ist ein ELLbogen-SchutzPACK beim Skaten

Wer ist Super-HPCler? ☺

A Nase anfassen

B Klatschen

C Augen zukneifen

D Winken

Was ist BLAS?

A Eine Aufforderung zum Pusten

B Ein Bib-Ansatz für LinAlg-Komponenten

C Ausgestoßene Atem-luft von Walen

D Ein Löseralgorithmus für Gleichungssysteme

Wer ist Super-HPCler? ☺

A Nase anfassen

B Klatschen

C Augen zukneifen

D Winken

Was ist BLAS?

A Eine Aufforderung zum Pusten

B Ein Bib-Ansatz für LinAlg-Komponenten

C Ausgestoßene Atem-luft von Walen

D Ein Löseralgorithmus für Gleichungssysteme

Wer ist Super-HPCler? ☺

A Nase anfassen

B Klatschen

C Augen zukneifen

D Winken

Welche der folgenden Abkürzungen spielt in PETSc eine Rolle?

A DAAD

B MDAD

C DAMD

D DMDA

Wer ist Super-HPCler? ☺

A Nase anfassen

B Klatschen

C Augen zukneifen

D Winken

Welche der folgenden Abkürzungen spielt in PETSc eine Rolle?

A DAAD

B MDAD

C DAMD

D DMDA

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