Indo-German Winter Academy
Course 2

High Performance Computing for Engineering Problems

Prof. Biswas, Prof. Buwa, Prof. Chakraborty,
Prof. Mittal, Prof. Mishra,
Prof. Poeschel, Prof. Ryssel
Prof. Rüde, Prof. Wellein

Roorkee, December 2009

Talks

- Talks should be approx. 45-50 minutes
- 30-45 min discussion
- Speak s-l-o-w-l-y!
- Discussion
- Interaction
- Ask questions!
- Enjoy the winter school!

Talks

- Challenges in Computational Engineering and Introduction to Fast Algorithms

Ulrich Rüde
Lehrstuhl für Systemsimulation

Indo-German Winter Academy 2009

Schedule

Sunday, December 13
17:30  Inaugural Session
18:30  Campus Tour
20:00  Dinner

Monday, December 14
09:00  Introduction to Course 2: Challenges in Computational Engineering and Introduction to Fast Algorithms (Prof. U. Rüde, Erlangen)
10:30  Coffee Break
11:00  Classification of partial differential equations and their solution characteristics (Shweta Jain Kothari, IITR, Tutors: Prof. V. Buwa, Prof. C. Mishra, Prof. U. Rüde)
12:30  Lunch
14:00  Fundamentals of finite difference methods (Sumit Somani, IIT B, Tutor: Prof. G. Biswas)
15:30  Coffee Break
16:00  Fundamentals of finite elements methods: variational methods for the Laplace and Poisson Equation (Vinay Prashanth Subbiah, IIT M, Tutor: Prof. S. Mittal)
17:30  Coffee Break
18:00  Fundamentals of molecular dynamics simulations (Rainer Hartmann, Erlangen, Tutors: Prof. S. Chakraborty, Prof. T. Poeschel)
19:30  Dinner
20:30  Special Lecture: Simulation of Multi Phase Flows (Prof. V.V. Buwa, IITD)
### Schedule

#### Tuesday, December 15

<table>
<thead>
<tr>
<th>Time</th>
<th>Activity</th>
</tr>
</thead>
<tbody>
<tr>
<td>09:00</td>
<td>Finite volume method: basic principles and examples</td>
</tr>
<tr>
<td></td>
<td>(Mainak Chowdhury, IIT K, Tutor: Prof. G. Biswas)</td>
</tr>
<tr>
<td>10:30</td>
<td>Coffee Break</td>
</tr>
<tr>
<td>11:00</td>
<td>Discretization of convection-diffusion type equations</td>
</tr>
<tr>
<td></td>
<td>(Piyush Kumar Sao, IIT M, Tutors: Prof. V. Buwa, Prof. S. Chakraborty)</td>
</tr>
<tr>
<td>12:30</td>
<td>Lunch</td>
</tr>
<tr>
<td>14:00</td>
<td>An introduction to the lattice Boltzmann method</td>
</tr>
<tr>
<td></td>
<td>(Soham Mehta, IIT B, Tutors: Prof. U. Rüde, Chr. Feichtinger)</td>
</tr>
<tr>
<td>15:30</td>
<td>Coffee Break</td>
</tr>
<tr>
<td>16:00</td>
<td>Rigid body dynamics of interacting particles</td>
</tr>
<tr>
<td></td>
<td>(Tobias Preclik, Erlangen, Tutors: Prof. Ruede Prof. T. Poeschel, K. Iglberger)</td>
</tr>
<tr>
<td>17:30</td>
<td>Coffee Break</td>
</tr>
<tr>
<td>18:00</td>
<td>Special Lecture: Granular Media</td>
</tr>
<tr>
<td></td>
<td>(Prof. T. Poeschel, Erlangen)</td>
</tr>
<tr>
<td>19:00</td>
<td>Dinner</td>
</tr>
<tr>
<td>20:30</td>
<td>Industrial Collaborator: Coal Gasification</td>
</tr>
<tr>
<td></td>
<td>(Mr. Hauser, Siemens Corporate Research, Erlangen)</td>
</tr>
</tbody>
</table>

#### Wednesday, December 16

<table>
<thead>
<tr>
<th>Time</th>
<th>Activity</th>
</tr>
</thead>
<tbody>
<tr>
<td>09:00</td>
<td>Fundamentals of Parallel Processing</td>
</tr>
<tr>
<td></td>
<td>(Dishant Ailawadi, IIT R, Tutors: Prof. Wellein. J. Götz)</td>
</tr>
<tr>
<td>10:30</td>
<td>Coffee Break</td>
</tr>
<tr>
<td>11:00</td>
<td>Supercomputer Architecture</td>
</tr>
<tr>
<td></td>
<td>(B V V Sri Raj Dutt, IIT R, Tutors: Prof. Wellein, M. Stürmer)</td>
</tr>
<tr>
<td>12:30</td>
<td>Lunch</td>
</tr>
<tr>
<td>14:00</td>
<td>Performance analysis and tuning of parallel programs</td>
</tr>
<tr>
<td></td>
<td>(Nitin Dua, IIT G, Tutors: Prof. Wellein, T. Gradl)</td>
</tr>
<tr>
<td>15:30</td>
<td>Coffee Break</td>
</tr>
<tr>
<td>16:00</td>
<td>Multi-Core Programming of a blood flow simulation on the IBM Cell Processor</td>
</tr>
<tr>
<td></td>
<td>(Kanav Goyal, IIT D, Tutors: Prof. Rüde and J. Götz)</td>
</tr>
<tr>
<td>17:30</td>
<td>Coffee Break</td>
</tr>
<tr>
<td>18:00</td>
<td>Special Lecture: Simulation in Microfluidics</td>
</tr>
<tr>
<td></td>
<td>(Prof. S. Chakraborty, IITKGP)</td>
</tr>
<tr>
<td>19:00</td>
<td>Break</td>
</tr>
<tr>
<td>20:30</td>
<td>Discussion of Participating Professors</td>
</tr>
</tbody>
</table>

#### Thursday, December 17

<table>
<thead>
<tr>
<th>Time</th>
<th>Activity</th>
</tr>
</thead>
<tbody>
<tr>
<td>09:00</td>
<td>Sightseeing</td>
</tr>
<tr>
<td>12:30</td>
<td>Lunch</td>
</tr>
<tr>
<td>14:00</td>
<td>Sightseeing/Shopping</td>
</tr>
<tr>
<td>19:30</td>
<td>Dinner</td>
</tr>
</tbody>
</table>

#### Friday, December 18

<table>
<thead>
<tr>
<th>Time</th>
<th>Activity</th>
</tr>
</thead>
<tbody>
<tr>
<td>09:00</td>
<td>Multigrid methods (Varun Bedi, IIT KGP, Tutors: Prof. Rüde and T. Gradi)</td>
</tr>
<tr>
<td>10:30</td>
<td>Coffee Break</td>
</tr>
<tr>
<td>11:00</td>
<td>Free surface LBM computations of bubbly flows - Interface tracking techniques</td>
</tr>
<tr>
<td></td>
<td>(Pritika Goyal, IIT D, Tutors: Prof. V. Buwa, Prof. Ruede, S. Donath)</td>
</tr>
<tr>
<td>12:30</td>
<td>Lunch</td>
</tr>
<tr>
<td>14:00</td>
<td>Computer Graphics and Scientific Visualization</td>
</tr>
<tr>
<td>15:30</td>
<td>Coffee Break</td>
</tr>
<tr>
<td>16:00</td>
<td>Possibilities to study in Germany</td>
</tr>
<tr>
<td>16:45</td>
<td>Invited Talk: Compressible Flows and Hyperbolic Conservation Laws</td>
</tr>
<tr>
<td></td>
<td>(Prof. S.V. Raghu Rama Rao, Indian Institute of Science)</td>
</tr>
<tr>
<td>18:00</td>
<td>Break</td>
</tr>
<tr>
<td>18:15</td>
<td>Algorithms for Image Processing</td>
</tr>
<tr>
<td></td>
<td>(Abhishek Bafna, IIT D, Tutors: Prof. Rüde and H. Köstler)</td>
</tr>
<tr>
<td>19:45</td>
<td>Dinner</td>
</tr>
</tbody>
</table>

#### Saturday, December 19

<table>
<thead>
<tr>
<th>Time</th>
<th>Activity</th>
</tr>
</thead>
<tbody>
<tr>
<td>09:00</td>
<td>Free for individual discussions</td>
</tr>
<tr>
<td>11:30</td>
<td>Feedback Session for Course 2</td>
</tr>
<tr>
<td>12:00</td>
<td>Valedictory Session</td>
</tr>
<tr>
<td>12:30</td>
<td>Lunch</td>
</tr>
</tbody>
</table>

### Before We Start

- **Questions?**
- **Introduction Round**
Motivation: Computational Science for the 21st Century

U. Rüde (LSS Erlangen, ruede@cs.fau.de)
In collaboration with many

Lehrstuhl für Informatik 10 (Systemsimulation)
Universität Erlangen-Nürnberg
www10.informatik.uni-erlangen.de

Indo-German Winter Academy 2009

The Two Principles of Science

Theory
Mathematical Models, Differential Equations, Newton

Experiments
Observation and prototypes empirical Sciences

Computational Science
Simulation, Optimization (quantitative) virtual Reality

How much is a PetaFlops?

- $10^8 = 1$ MegaFlops: Intel 486
  33MHz PC (~1989)
- $10^9 = 1$ GigaFlops: Intel Pentium III
  1GHz (~2000)
  - If every person on earth does one operation every 6 seconds, all humans together have 1 GigaFlops performance (less than a current laptop)
- $10^{12} = 1$ TeraFlops: HLRB-I
  1344 Proc., ~ 2000
- $10^{15} = 1$ PetaFlops
  >100 000 Proc. Cores
  Roadrunner/Los Alamos: Jun 2008
  - If every person on earth runs a 486 PC, we all together have an aggregate Performance of 6 PetaFlops.

Computational Science & Engineering

Science Engineering

Observation Experiment Building prototypes

Computer models in physics, chemistry, electrical engineering mechanical engineering material sciences chemical engineering, ...

Virtual experiments Virtual prototypes Virtual reality

Computer simulation

Theory

Computation

Computational Science
**Dynamics of many objects**
(composed objects)

**Granular Flows with Non-Spherical Particles**
and Frictional Elastic Collisions

64 Processes, 62658 particles, each composed of 2-5 overlapping spheres, approx. 13 hours runtime

ACM Transactions on Graphics 24:946-956, 2005

---

**Simulation of Metal Foams**

- Example application:
  - Engineering: metal foam simulations
- Based on LBM:
  - Free surfaces
  - Surface tension
  - Disjoining pressure to stabilize thin liquid films
  - Parallelization with MPI and load balancing
- Collaboration with C. Körner (Dept. of Material Sciences, Erlangen)
- Other applications:
  - Food processing
  - Fuel cells

---

**Virtual Fluidized Bed**

512 Processors
HLRB-II

Simulation Domain
Size: 180x198x360 cells of LBM
900 capsules and 1008 spheres = 1908 objects
Number time steps: 252,000
Run Time:
07h 12 min
Massively parallel simulations with many particles (RRZE woody cluster)

- xsize: 1400
- ysize: 400
- zsize: 640
- 358,4e6 lattice cells
- 129 processes/cores
- on woody cluster
- 1900 objects
- 20,000 timesteps
- 5:30h

Simulations with Fluid Control

Goal

- Solve the linear system

\[ Ax = b; \quad A \in \mathbb{R}^{n \times n}, \quad x, b \in \mathbb{R}^n. \]

- If the matrix is non-singular, the system has a unique solution \( x \)
- But … not all linear systems are created equal.
  - Developing specialised solvers for large finite element (finite volume, …) problems has been (and will stay) an active research topic for decades
  - The key is to exploit the special features of the systems

Recapitulation: Direct Linear Systems Solvers

This is THE Standard Algorithm: Gaussian Elimination

- direct method: i.e. it computes the exact result (except for roundoff errors) after a finite number of operations
- complexity (for dense matrices): \( O(n^3) \)
- how does it work:
  - for all \( O(n^2) \) elements of \( A \) below the diagonal: \( a_{ij} \)
    - perform a "row elimination step" by adding a suitable multiple of the "pivot" row to the row \( i \) with the goal of annihilating the element \( a_{ij} \)
  - Usually pivoting necessary (to keep roundoff errors low)
  - Must be done in the right order as not to destroy already created zeros
  - Elimination also for rhs \( b \)
  - or LU-decomposition (and forward substitution)
  - backward substitution
Gaussian Elimination for FE-Systems (1)?

- In its classical form, Gaussian Elimination is much too expensive for FE problems: Complexity $O(n^3)$
- Example: $n=10^6$:
  - $n^3 = 10^{18}$ - this corresponds to $10^6$ Seconds (on a 1 Gigaflops PC) = 31 years of compute time
  - The best known multigrid algorithms (for the 2D problem from this class) needs about $30n$ operations, so it should compute the solution in less about 30 milliseconds (which is NOT at all easy to do!)
  - Even with the fastest supercomputers Gaussian Elimination would be much slower than multigrid on a laptop.
- So Gaussian Elimination cannot be used for FE systems!
- Gaussian Elimination CAN be used - in fact it is state of the art in most commercial FE packages (such as NASTRAN)
- It must be modified to exploit existing zeros in the FE-matrix

Gaussian Elimination for FE-Systems (2)?

- For 1D (ODE) problems, the stiffness matrix is „tridiagonal“:
- The 0-s below the subdiagonal can be exploited: they need not be eliminated - they are never touched - they need not be stored - a row elimination reduces to a few elementary operations - the whole Gaussian Elimination then has reduced complexity of $O(n)$ - the algorithm is perfectly suited for this - no need to use an iterative method here.

Gaussian Elimination for FE-Systems (3)?

- For 2D (PDE) problems, the stiffness matrix has more complicated structure
  - The matrix is still sparse: e.g. only 5 or 9 entries in each row may be nonzero.
  - A matrix is called sparse, if it contains „many more“ zero entries than non-zero entries
  - typical: the number of nonzeros in each row is less than a constant (such as 10)
  - sparse matrices are common in many practical applications!
  - even if a matrix is sparse, its inverse is usually not: do not attempt to compute the inverse (actually computing the inverse is a bad idea in almost all cases, even for dense matrices)
  - we must develop and study algorithms that exploit sparsity (to save storage and compute time)

- For problems on a structured grid, the FE stiffness matrix will have „banded“ structure
  - in general, however, the FE stiffness matrix is unstructured
  - Elimination suffers from „fill-in“, that is, originally existing zeros are destroyed in the elimination process and must be re-eliminated
  - For 2D-Fe-problem with $n$ unknowns (and in lexicographic ordering), $O(n^{1.5})$ unknowns must be eliminated
  - better orderings exist: e.g. Minimum degree ordering, nested dissection ordering
    - finding orderings that reduce fill-in has become research topic in ist own right and is still active (especially e.g. for parallel computing!)
    - For 2D-problems, good elimination based algorithms are practically useful and are routinely used in practice, though they fail to be „optimal“
  - In (large) 3D problems - elimination based algorithms are not sufficiently efficient
Gaussian Elimination for banded FE-Systems

Fill in destroys zeros within „band“ when eliminating.

In our case \( n = N^2 \).

Iterative Linear System Solver Algorithms

- For large 2D and especially for 3D problems iterative linear systems solvers are preferable - because they can have (much) better complexity.

- Idea:
  - generate a sequence of approximate solution vectors
  - that approach the true solution.

\[ x^0, x^1, x^2, \ldots \text{ such that } x^* := \lim_{i \to \infty} x^{(i)} \text{ solves } Ax^* = b. \]

- Each iterate \( x^{(i)} \) must be easy (that is: cheap) to compute
  - low compute requirements
  - low memory requirements

Grid and Matrix

- Distinguish between the
  - grid (or finite element mesh)
  - and the matrix (finite element stiffness matrix)

- In 2D
  - a structured mesh (grid) may have \( N \times N \) unknowns that can be arranged (e.g.) in lexicographic order in a vector with dimension \( n = N \times N \)
  - The stiffness matrix has dimension \( N^2 \times N^2 \) and banded structure
  - In the following we will use the grid representation to visualize the vectors of unknowns in FE problems
Recapitulation: Gauss-Seidel Iteration
(The Mother of all Iterative Solvers)

see also
http://www-i1.informatik.rwth-aachen.de/~algorithmus/algo39.php

In Grid-Notation:

```c
real u[N+1][N+1];  /* initialisieren mit Randwerten, in Inneren
                   "O" oder Mittelwert der Randwerte */

for (int it=0; it<MAXIT; it++) {
  real udiff=0;
  for (int i=1; i<N; i++)
    for (int j=1; j<N; j++) {
      real un = 0.25*(u[i-1][j]+u[i+1][j]+u[i][j+1]+u[i][j-1])
      udiff += fabs(u[i][j]-un);
      u[i][j] = w*un + (1-w)*u[i][j];
    }
  if (udiff / N*N > TOL) break; /*
}
```

Model Problem: Dirichlet Boundary

Visualization of Convergence

before any iteration
after 1 iterations
after 2 iterations
after 10 iterations
after 100 iterations
after 1000 iterations
overlayed with true solution
overlayed with true solution
Linear Iterative Methods in Matrix Notation

- The classical choices for linear iterative methods are based on a matrix splitting
  \[ A = L + D + R \]
  - Jacobi Iteration:
    \[ x^{k+1} = x^k + D^{-1}(f - Ax^k) \]
  - Gauss-Seidel Iteration
    \[ x^{k+1} = x^k + (L + D)^{-1}(f - Ax^k) \]
  - SOR method (for \( 0 < \omega < 2 \))
    \[ x^{k+1} = x^k + (L + \frac{1}{\omega}D)^{-1}(f - Ax^k) \]

Clearly, the exact solution \( x^* = A^{-1}f \) is a fixed point of the iteration.

To check whether the iteration converges \( x^k = x^* + e^k \) and \( x^{k+1} = x^* + e^{k+1} \) may be inserted:

\[ e^{k+1} = (I - B^{-1}A)e^k \]

Convergence is obtained, when the error norm is reduced.

\[ ||e^{k+1}|| \leq ||I - B^{-1}A|| ||e^k|| \]

Convergence is obtained, when the error norm is reduced.

\[ ||I - B^{-1}A|| \leq 1 \]

Extreme (but useless) case: \( A = B \)

Design goal: good compromise between cost for each iteration and speed of convergence.
SOR-method in elementary form

For \( i = 1, \ldots, n \):

\[
x_i^{k+1} = (1 - \omega) x_i^k + \frac{\omega}{a_{ii}} \left( b_i - \sum_{j=1}^{i-1} a_{ij} x_j^{k+1} - \sum_{j=i+1}^{n} a_{ij} x_j^k \right)
\]

- For typical systems (such as the Laplace operator in 2D), an optimally chosen \( \omega \) can reduce the number of required iterations from \( O(n) \) to \( O(n^{0.5}) \).
- The overall complexity is then \( O(n^{1.5}) \) (instead of \( O(n^2) \)) - provided the sparsity of the matrix is exploited.
- The optimal parameter can in practice be determined experimentally.
- For model problems an analysis exists that allows to calculate the optimal \( \omega \) analytically.

Explicitly writing the linear system...

Here all \( x^i \) are brought to the „right hand side“, leading to a simple formula to compute the \( x^{i+1} \).
Efficient implementation of iterative methods

- We must not use the matrix data structure as a two-dimensional \( n \times n \) array.
- We must avoid to store and operate on all the zeros
- Banded matrices:
  - storing the diagonal bands in a collection of 1D-vectors is the standard technique.
  - The grid and corresponding matrix structure is "hard-coded" with this data structure
- Stencil based storage
  - In many applications - when element shapes and material parameters are identical, the matrix rows will be identical
  - No matrix needs to be stored, just a "stencil" that represents the entries in a matrix row for all (or at least many) rows of the matrix

Compressed row matrix data structure

- Make no assumption on sparsity structure
- they don't store any unnecessary elements
- are efficient (but not very efficient)
  - needing an indirect addressing step for every single scalar operation in a matrix-vector product
- CRS: \( \text{nnz} = \) number of nonzeros in \( A \).
  - puts the subsequent nonzeros of the matrix rows in contiguous memory locations
  - 3 vectors
    - one for floating-point numbers (\( \text{val} \))
    - other two for integers (\( \text{col_ind, row_ptr} \))
  - convention: \( \text{row_ptr}(n+1) = \text{nnz}+1 \)

Example of matrix in CRS format

\[
A = \begin{bmatrix}
10 & 0 & 0 & 0 & -1 & 0 \\
3 & 9 & 0 & 0 & 0 & 3 \\
0 & 7 & 8 & 7 & 0 & 0 \\
3 & 0 & 8 & 7 & 5 & 0 \\
0 & 8 & 0 & 9 & 9 & 13 \\
0 & 4 & 0 & 0 & 2 & -1
\end{bmatrix}
\]

\[
\begin{array}{ccccccccccccc}
\text{val} & 10 & -1 & 3 & 9 & 3 & 7 & 8 & 7 & 3 & \ldots & 9 & 13 & 4 & 2 & -1 \\
\text{col-ind} & 1 & 5 & 1 & 2 & 6 & 2 & 3 & 4 & 1 & \ldots & 5 & 6 & 2 & 5 & 6 \\
\text{row-ptr} & 1 & 3 & 6 & 9 & 13 & 17 & 20
\end{array}
\]

Properties of the CRS format

- Memory cost proportional to number nonzeros (1 int + 1 float/double)
- Matrix-vector multiply (or Gauss-Seidel iteration) costs proportional to number nonzeros
- adding new matrix coefficients is costly
- deleting matrix coefficients may be costly
- access to element \((i,j)\) requires search in row
- indirection in memory access may be slow on some machines
Compressed row matrix data structure

- Using CRS for iterative solver
- Example: matrix-vector multiply $y = A x$
  - initialize $y = 0$
  - for $i = 1$ to number_of_rows
    - for $j = row_ptr(i)$ to row_ptr($i+1$)-1
      - $y(i) = y(i) + val(j) * x(col_ind(j))$
  - Complexity: $O(nnz) << O(n^2)$

- Variants of CRS
  - store diagonal element first in each row.
  - store the inverse of the diagonal element instead of element itself
    - helps to replace repeated divisions by cheaper multiplications in Gauss-Seidel and similar iterative schemes

Summary and Extension

- Linear system solvers for simulation
  - direct methods
  - iterative methods
- Classical iterative methods
  - Gauss-Seidel
  - SOR
- Outlook
  - Conjugate gradient methods (Krylov space methods)
  - Multilevel/ multigrid methods

Questions?