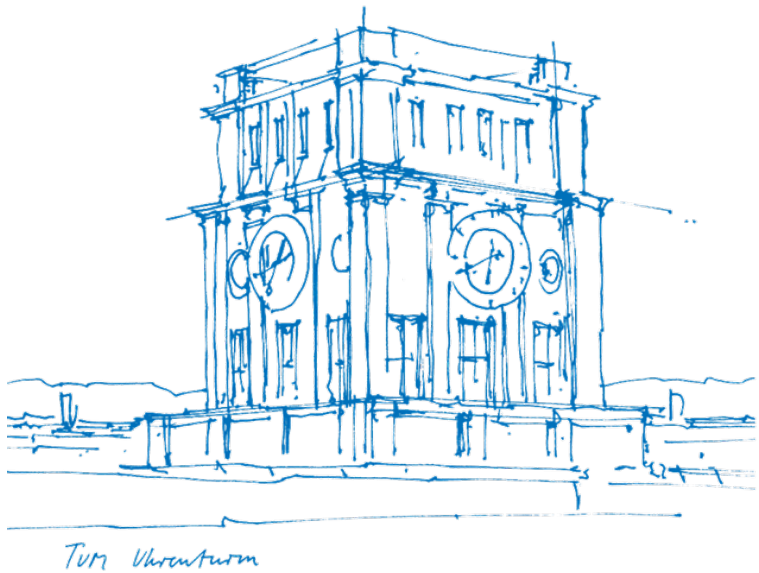
HPC Lab

Session 5: Project

Carsten Uphoff, Chaulio Ferreira, Michael Bader
TUM – SCCS

18 December 2017



Linear acoustics

Partial differential equations[1]

$$\frac{\partial p}{\partial t} + K_0 \frac{\partial u}{\partial x} + K_0 \frac{\partial v}{\partial y} = 0$$
$$\frac{\partial}{\partial t} u + \frac{1}{\rho_0} \frac{\partial p}{\partial x} = 0$$
$$\frac{\partial}{\partial t} v + \frac{1}{\rho_0} \frac{\partial p}{\partial y} = 0$$

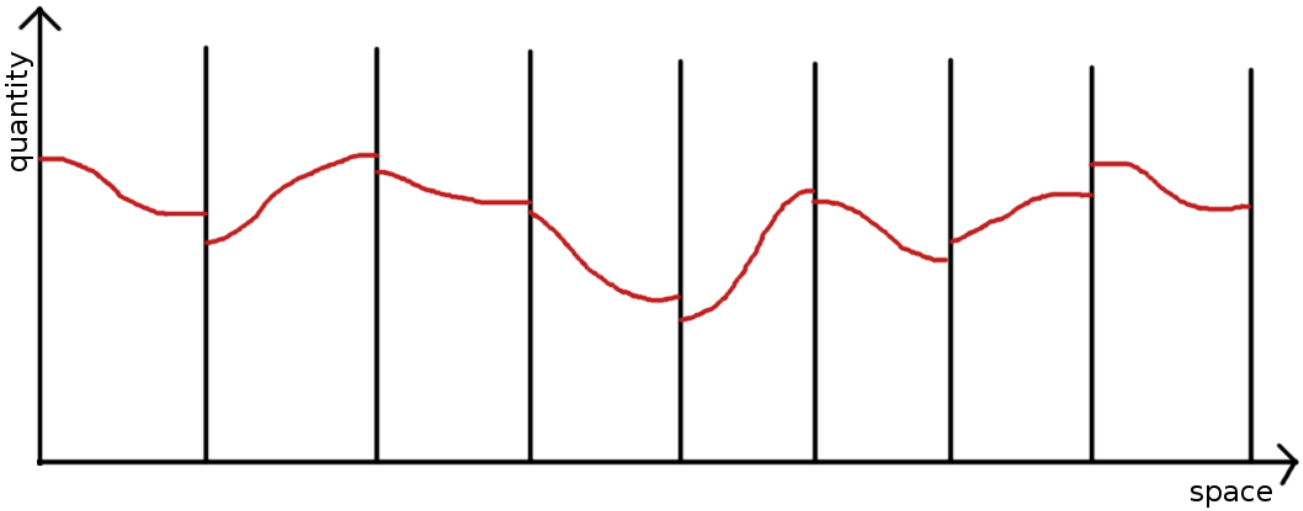
In matrix-vector notation

$$\frac{\partial Q}{\partial t} + A \frac{\partial Q}{\partial x} + B \frac{\partial Q}{\partial y} = 0$$

Goal: Find functions $Q_k(x, y, t)$, with $Q_k(x, y, 0) = Q_k^0(x, y)$, that satisfy the partial differential equations.

[1] LeVeque, RJ. Finite Volume Methods for Hyperbolic Problems (p.425). Cambridge, 2002.

Linear acoustics: Discretisation ideas



- Subdivide domain into finite elements.
- Represent the quantity on every element as a polynomial basis expansion, i.e. $Q_k(x, y, t) = \sum_{l=1}^{N} \hat{Q}_{lk}(t)\phi_l(x, y)$.
- Leads to ordinary differential equations that may be solved with standard methods like Euler, Runge-Kutta. Here: ADER approach.
- Quantity may be discontinuous at element boundaries. Continuity is only enforced weakly via numerical fluxes.

Linear acoustics: Grid

Finite elements are rectangles in a cartesian grid on the domain $[0, 1] \times [0, 1]$. That is, the grid consists of $(N_x + 1)(N_y + 1)$ points

$$P_{i,j} = \begin{pmatrix} i \cdot h_x \\ j \cdot h_y \end{pmatrix}, i \in [0, N_x], j \in [0, N_y],$$

where $h_x = 1/N_x$ and $h_y = 1/N_y$. Then, there are $N_x N_y$ finite elements consisting of points $P_{i,j}, P_{i+,j}, P_{i,j+1}, P_{i+1,j+1}$.

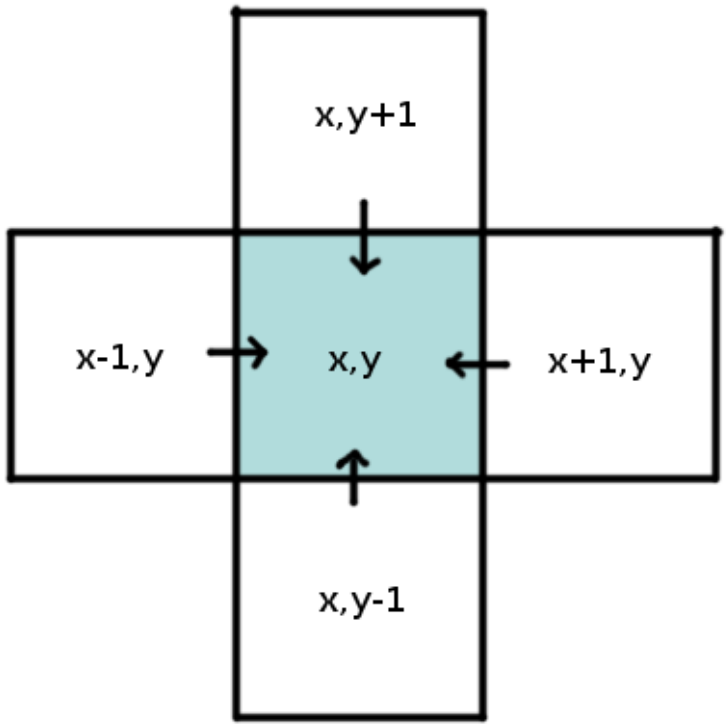
Local coordinate system: You might see ξ, η in the code. These are element-local coordinates defined as

$$\xi^{(m)}(x, y) = \frac{x - (P_m)_1}{h_x}, \quad \eta^{(m)}(x, y) = \frac{y - (P_m)_2}{h_y},$$

where $\xi^{(m)}, \eta^{(m)} \in [0, 1] \subset \mathbb{R}$.

Linear acoustics: Kernels

You find the main computation core in `src/Simulator.cpp`. The main loop consists of two parts: The first part are all operations that are local to an element. The second part interacts with neighboring according to the following 5 point stencil:



Linear acoustics: Kernels

The kernel operations itself consists almost entirely out of matrix-matrix products of the form

$$KQA,$$

where $K \in \mathbb{R}^{B \times B}$, $Q \in \mathbb{R}^{B \times 3}$, $A \in \mathbb{R}^{3 \times 3}$.

Note: K and A may be sparse.

Linear acoustics: Boundary conditions

Boundary conditions are always periodic. For example, the right neighbour of an element at the right boundary is the element at the left boundary on the same height.

Obtaining and running the code

```
git clone https://github.com/uphoffc/LinA.git
```

Build with “scons” (scons.org)

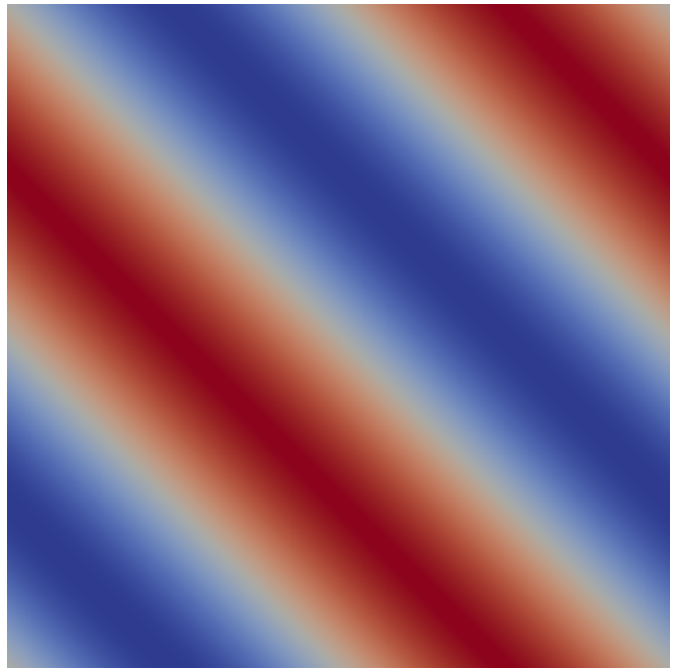
```
scons order=2-12 [compiler=gcc|intel] [compileMode=debug|release]
```

Usage

```
./build/lina [-t <double>] [-i <double>] [-o <string>] -y <int>
-x <int> -s <int> [--] [--version] [-h]
```

- -x: Number of cells in x direction.
- -y: Number of cells in y direction.
- -s: Scenario 0-3.
- -t: Final simulation time.
- -o: Base file name of wavefield output. Hint: Use directories, i.e. -o output/test. The file output/test.xdmf can be opened with ParaView.
- -i: Time interval of wavefield output.

Scenario 0: Plane wave in homogeneous medium



Used as a convergence test. In contrast to other scenarios, it analyses the error with respect to the analytical reference solution.
The errors should be the same in a parallel version. So if your errors are off, your code is wrong. (Don’t forget to Allreduce the error for MPI.)

Scenario 0: More on convergence

Compiled with order=6.

./build/lina -s 0 -x 10 -y 10

```
    L2 error analysis
=================
Pressue (p):    3.01142e-06
X-Velocity (u): 1.55155e-06
Y-Velocity (v): 1.55155e-06
```

./build/lina -s 0 -x 20 -y 20

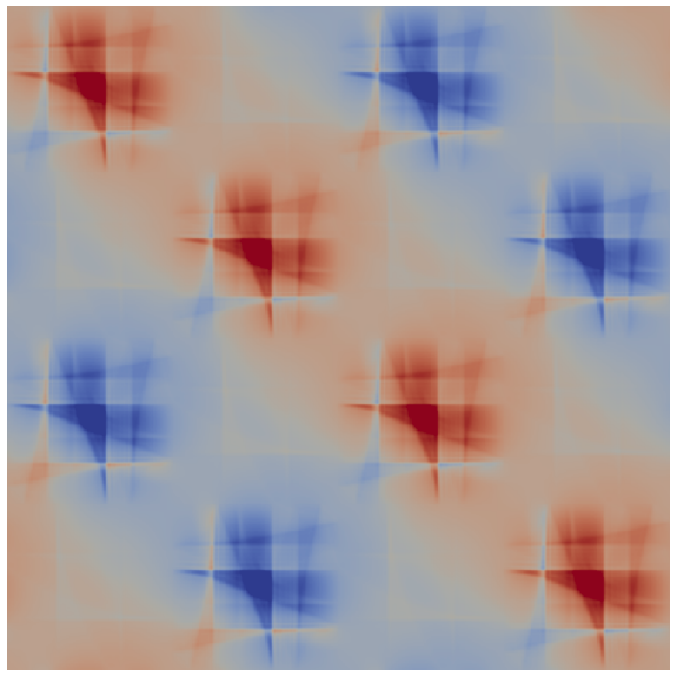
```
    L2 error analysis
=================
Pressue (p):    4.72039e-08
X-Velocity (u): 2.44428e-08
Y-Velocity (v): 2.44428e-08
```

Convergence rates are modeled by a power law, i.e. $\|error\| \leq Ch^p$. The convergence order p can be determined experimentally:

$$\hat{p} = \log(E_i/E_{i+1})/\log(h_{i+1}/h_i)$$

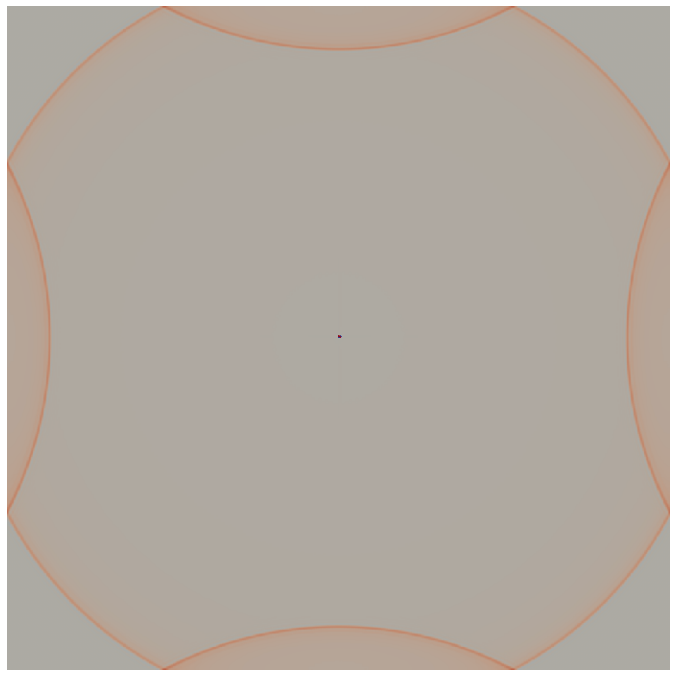
Example: $\hat{p} = \log(3.01142e\text{-}06/4.72039e\text{-}08)/\log(2) \approx 5.996$
(If you compile with order=6 $\hat{p}$ should be close to 6.)

Scenario 1: Plane wave in inhomogeneous medium



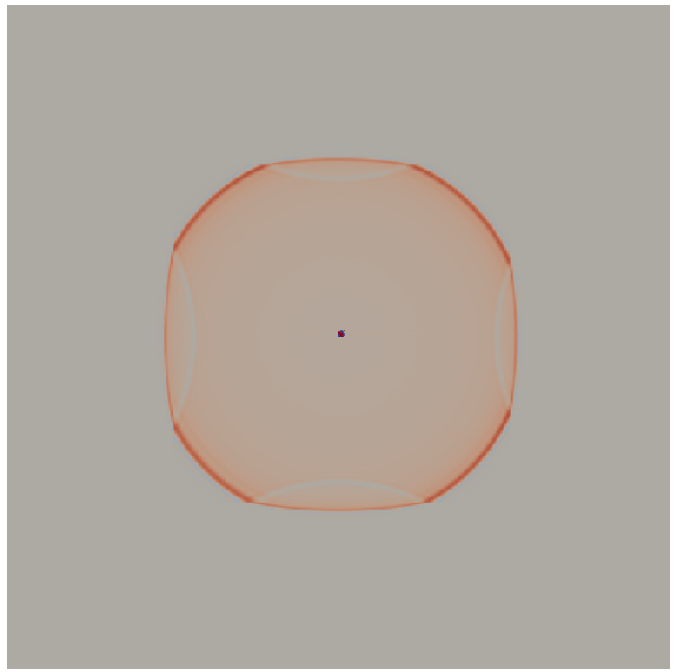
Medium with “checkerboard” material parameters. Your version should look the same and the minimum and maximum values should be the same.

Scenario 2: Point source in homogeneous medium



Pressure point source in the center which generates spherically symmetric waves. Don’t worry about the spot in the middle.
If a point source lies on the boundary of several cells you may arbitrarily assign it to a single cell. (Don’t forget that for MPI.)

Scenario 3: Point source in inhom. medium



Same as scenario 2 but with 2 material zones.

Project ideas

- Single-core optimization. E.g. precompute matrices, eliminate divisions, DGEMM optimization with AVX512, matrix padding...
- OpenMP parallelisation.
- MPI parallelisation. E.g. domain decomposition with copy/ghost layer, asynchronous communication...
- (Bonus) Parallel wavefield output. Consider libraries such as NetCDF or HDF5.

You do not need do optimise every order. Higher orders are likely to be compute bound whereas lower orders are likely to be memory bound. (Aim for higher orders.)

You can easily split tasks. E.g. single-node optimization is independent of MPI parallelisation.

Deliverables

- Optimization strategy (2 pages). You may send it to us upfront and we may give you some advice.
- Git-repository, which documents all of your code-changes
  $\rightarrow$ `http://gitlab.lrz.de`
- Documentation how to build and execute your code.
- Project report containing all graphs, results . . . created throughout the project phase.

Deadline: 29 January 2018

Hints

- Find measurements for performance.
- Consider all parallelization levels: Vectorisation, OpenMP, MPI.
- Target platform is the CoolMUC3 cluster.
- Document your improvements early.
- Use profilers to analyse the code.
- **Test convergence after every code change.**

Organisation

There will be no meeting on January 29, just hand in code and report.

We will also have a small competition: The group with the lowest time to solution on a 1000x1000 grid, 10th order, wavefield output off, on 16 KNL nodes wins a prize (gummy bears).