Numerical simulation of two-phase flows using a combined VOF / Levelset method

NWO Computational Science 2005

Delft University of Technology

Research team

- Sander van der Pijl, Kees Vuik & Guus Segal

  Department of Applied Mathematical Analysis

  Faculty of Electrical Engineering, Mathematics and Computer Science

- Benjamin Vrolijk & Frits Post

  Department of Mediamatics

  Faculty of Electrical Engineering, Mathematics and Computer Science

- Emil Coyajee & Bendiks Jan Boersma

  Laboratory of Aero- & Hydrodynamics

  Faculty of Mechanical Engineering

Outline

- Introduction
- Theory
- Numerical Method
- Parallel Implementation
- Visualization
- Closing remarks

Introduction

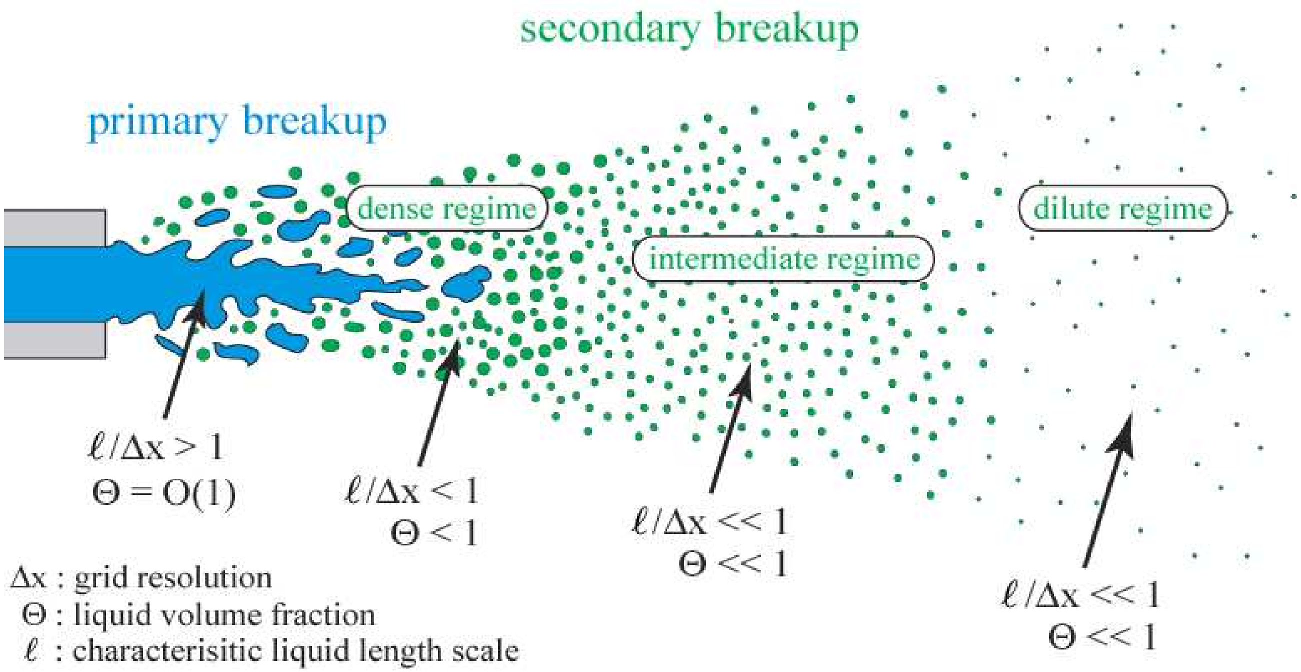
Incompressible two-phase flows

- gas-liquid
- liquid-liquid

Applications

• Chemical industry	(separation, boiling, ..)
• Combustion	(fuel injectors)
• Printing industry	(inkjets)
• Coating	(spray paint, ..)
• Maritime application	(green water loading, water waves)

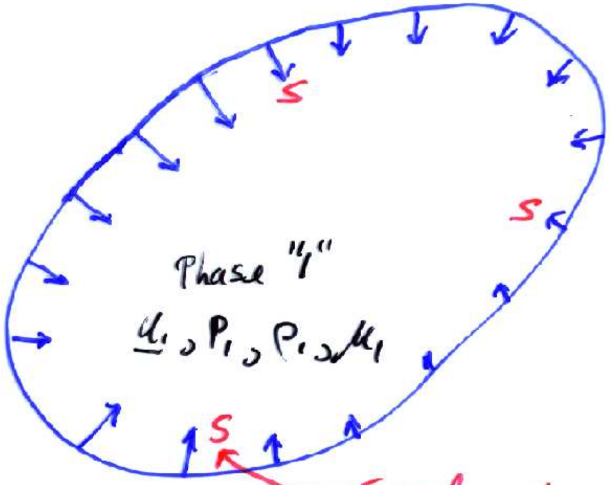
Applications: Spray combustion



(Ham et al. 2003)

Theory

Gas bubble in water



Phase "o"

$\underline{u}_o$ = velocity
P_o = Pressure
ρ_o = density
μ_o = dynamic viscosity

Surface tension = f(curvature)

Theory (cont'd.)

- Conservation of mass in phase "0" and "1"

$$\nabla \cdot \mathbf{u_{0,1}} = 0$$

- Conservation of momentum in phase "0" and "1"

$$\frac{\partial \mathbf{u_{0,1}}}{\partial t} + \mathbf{u_{0,1}} \cdot \nabla \mathbf{u_{0,1}} = -\frac{1}{\rho_{0,1}} \nabla p_{0,1} + \nabla \mu_{0,1} \left(\nabla \mathbf{u_{0,1}} + \nabla \mathbf{u_{0,1}}^{T} \right)$$

- Coupling between phase "0" and "1" through interface conditions

Interface conditions

- Continuity of velocity

$$\mathbf{u}_0 = \mathbf{u}_1$$

- Continuity of stresses

$$\mu_0(\nabla \mathbf{u_0} + \nabla \mathbf{u_0}^T) \cdot \mathbf{t} = \mu_1(\nabla \mathbf{u_1} + \nabla \mathbf{u_1}^T) \cdot \mathbf{t}$$

$$\mu_0(\nabla \mathbf{u_0} + \nabla \mathbf{u_0}^T) \cdot \mathbf{n} = \mu_1(\nabla \mathbf{u_1} + \nabla \mathbf{u_1}^T) \cdot \mathbf{n} + (p_1 - p_0) + \underbrace{\sigma\kappa \cdot \mathbf{n}}_{surf.\ tens.}$$

Interface conditions (cont'd).

- In principle μ is discontinuous and thus also $\mathbf{u}$
- Regularization of μ gives $\nabla \mathbf{u_0} = \nabla \mathbf{u_1} \rightarrow \nabla \mathbf{u}$
- Reduction of interface conditions to

$$(p_1 - p_0) + \sigma\kappa \cdot \mathbf{n} = 0$$

- Surface tension force regularized into a volume force (Brackbill et al., 1992):

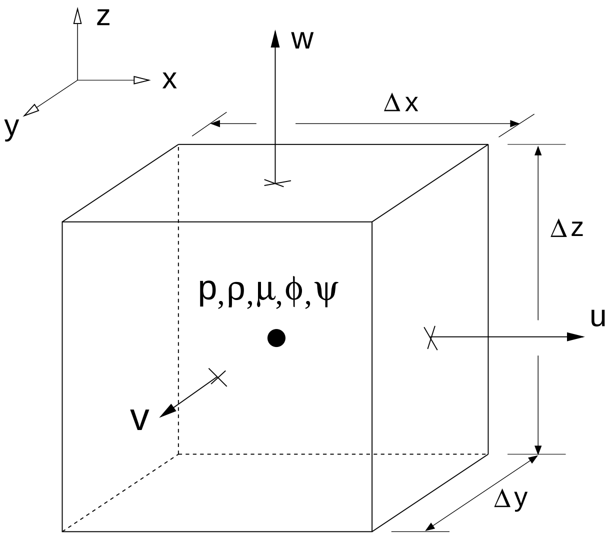
$$\int\int \sigma\kappa\mathbf{n}dS = \int\int\int \sigma\kappa\nabla H dV$$

- Interface normal $\mathbf{n}$ and curvature κ have to be known away from the interface

Computational method

Spatial discretization, with variable μ and ρ:

- Cartesian and uniform mesh
- Marker and Cell layout
- Discontinuous density, water/air $\rho_0/\rho_1 = 1000/1$
- Regularization of viscosity
- Continuous surface force approach → no interface conditions



Explicit time-integration for fluid flow and interface advection equations

- Navier-Stokes: pressure correction method

Surface representation

Required for calculation of ρ, μ and κ

Front tracking:

- Marker particles (Tryggvason and coworkers)

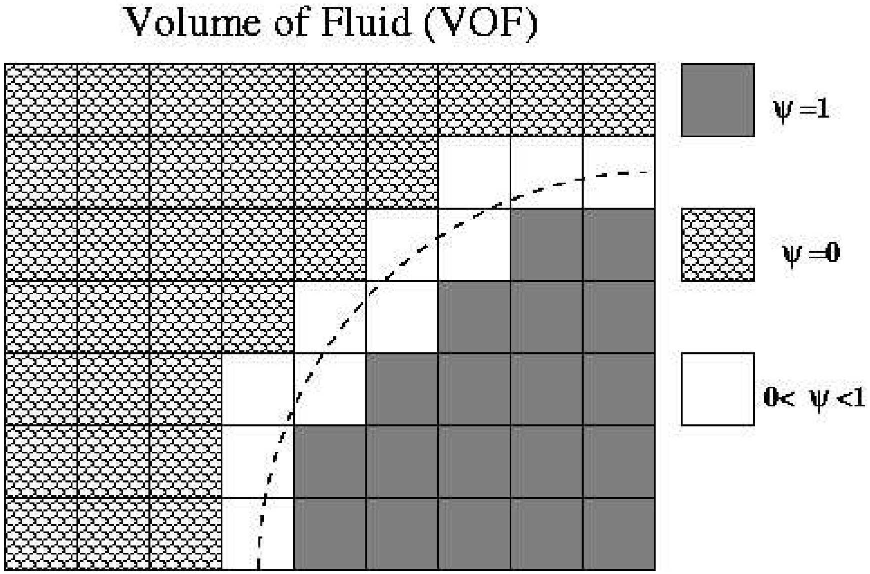
Front capturing:

- Volume of Fluid (VOF) (Rider & Kothe 1998, Scardovelli & Zaleski 1999, Renardy & Renardy 2002, Pilliod & Puckett 2004)
- Levelset (LS) (Sussman et al. 1994, Chang et al. 1996, Sethian 1999)
- LS/VOF (Sussman & Puckett 2000)

Front tracking/capturing:

- LS/Marker particles (Enright et al. 2003)

Volume of Fluid



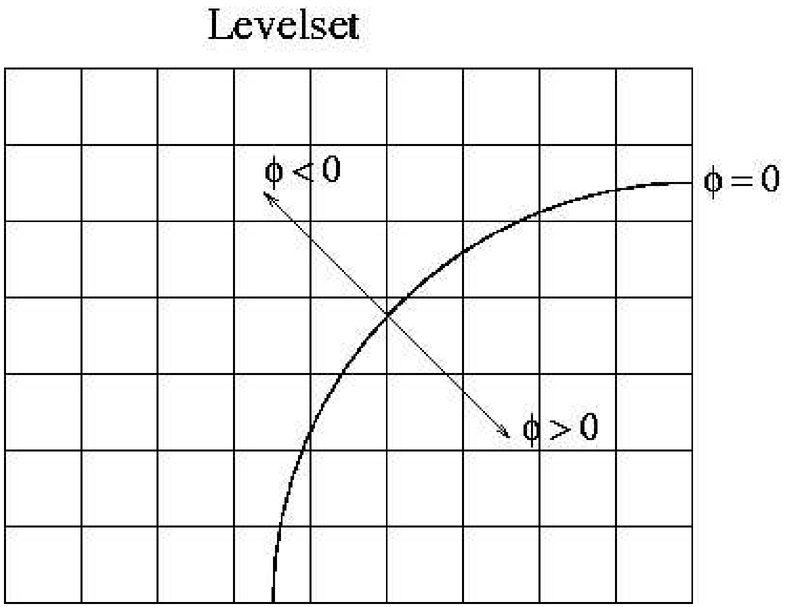
Advantage

- Mass conserving interface advection (numerically by construction)

Disadvantage

- Elaborate reconstruction of interface position and curvature, i.e. density, viscosity and surface tension.

Levelset



Advantage

- Straightforward extraction of interface position, computation of curvature, i.e. density, viscosity and surface tension

Disadvantage

- Numerical implementation of interface advection is not mass conserving

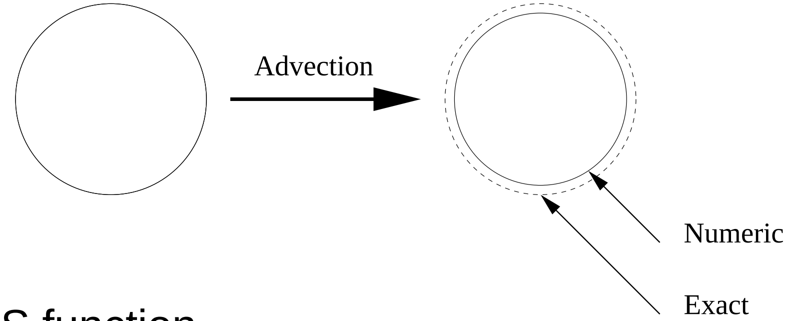
LS advection

$$\frac{\partial \phi}{\partial t} + \mathbf{u}\nabla\phi = 0$$

- Numerical implementation → dissipation

$$\frac{\phi^* - \phi^n}{\Delta t} = -u\frac{\phi_i^n - \phi_{i-1}^n}{\Delta x}$$

- Numerical dissipation → Mass loss/gain



- Apply small corrections to LS function

$$\phi^{n+1} = \phi^* + \delta\phi$$

How to calculate $\delta\phi$

- VOF reconstructed from LS

$$\psi^n = f(\phi^n, \nabla\phi^n)$$

- VOF advection is mass conserving by construction

$$\psi^n \rightarrow \psi^{n+1}$$

- Invert (Newton-Raphson) with ϕ^* as initial guess

$$\psi^{n+1} = f(\phi^{n+1}, \nabla\phi^{n+1})$$

- Mass conservation, up to a specified ϵ

Comp. Meth. Overview

- Velocity update $u^n, \phi^n \rightarrow u^*$
- LS advection $\phi^n, u^n \rightarrow \phi^*$
- VOF advection $\psi^n, \phi^n u^n \rightarrow \psi^{n+1}$
- LS correction $\phi^*, \psi^{n+1} \rightarrow \phi^{n+1}$
- Poisson equation

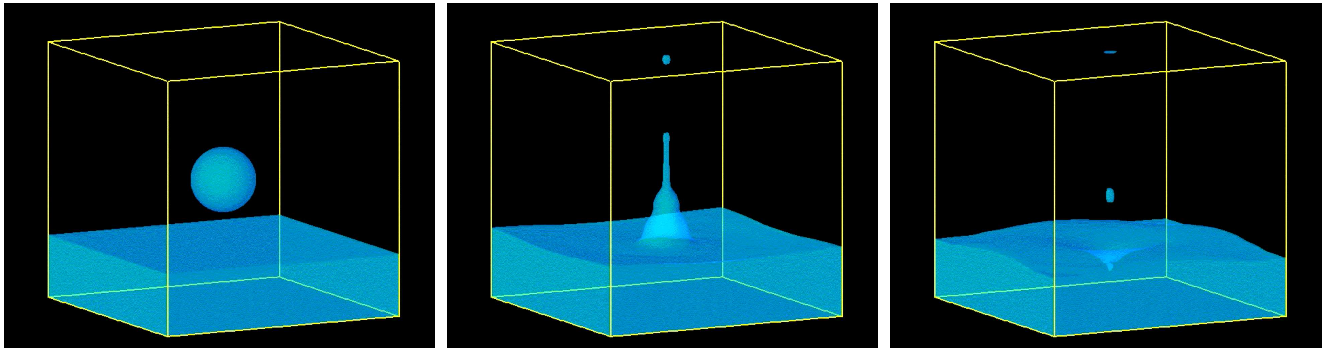
$$\nabla \frac{1}{\rho} \nabla p = \nabla \cdot u^*$$

- Solution with PCG
- Pressure correction $u^* \rightarrow u^{n+1}$

A Mass-Conserving Level-Set (MCLS) Method for Modeling of Multi-Phase Flows, S.P. van der Pijl, A. Segal, C.Vuik, & P. Wesseling (accepted: Int. Jour. for Num. Meth. in Fluids)

Results: Falling water drop

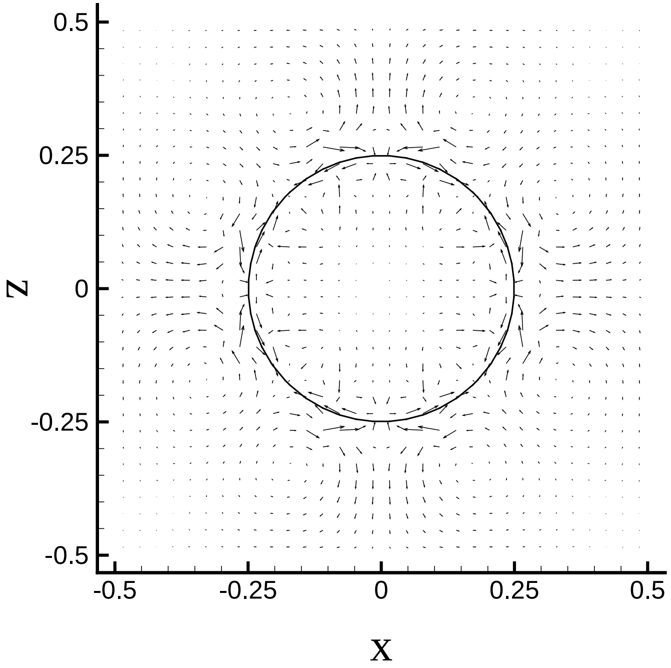
Serial code, numerical resolutions possible up to 128^3 gridpoints



Relative mass error < $1 \cdot 10^{-4}$

Stationary bubble/Laplace problem

- <u>Exact solution:</u> Pressure constant, velocity zero
- <u>Numerical solution:</u> Pressure perturbed, velocity non-zero



Surface tension

- Similar implementation/problems for all structured-grid methods
- Surface tension in N.S. equations: $\sigma\kappa\nabla H$ (Brackbill et al. 1992)
- Sources of error:
  1. Delta function approximation of the discontinuity
  2. Computation of curvature: $\kappa = \nabla \cdot \dfrac{\nabla\phi}{|\nabla\phi|}$ effect of $\delta\phi$
- Resulting symptoms: Parasitic currents for a stationary bubble (Laplace problem)

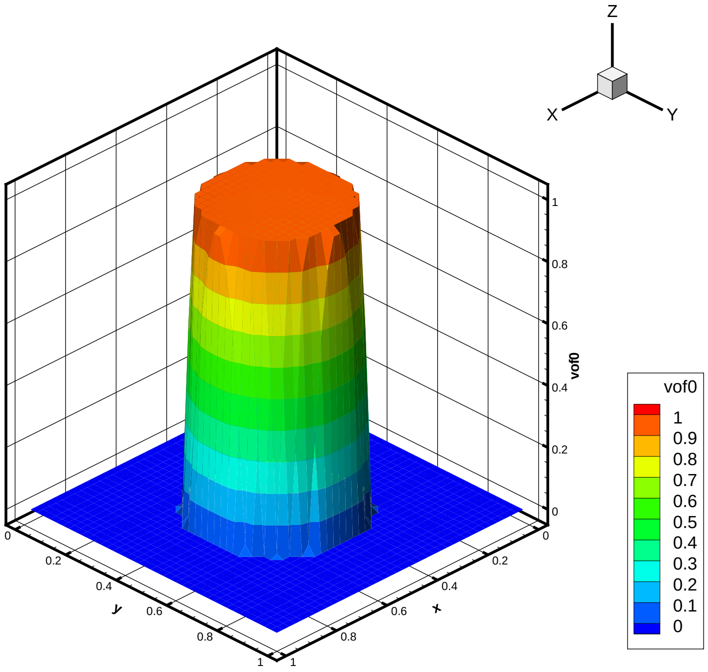
Complete VOF/Level-Set reconstruction (1)

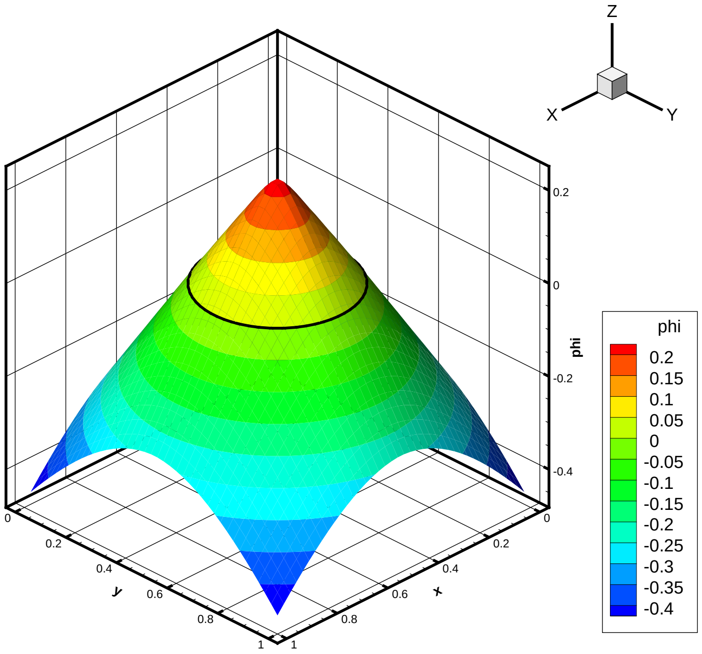
Simultaneously solve:

$$\psi = f(\phi, \nabla\phi) \quad \text{and} \qquad \underbrace{|\nabla\phi| = 1}_{distance\ function}$$

- $|\nabla\phi| = 1$ solved by 1st order Fast Marching method (Sethian 1999)
- $\psi = f(\phi, \nabla\phi)$ solved up to machine-precision

Complete VOF/Level-Set reconstruction (2)





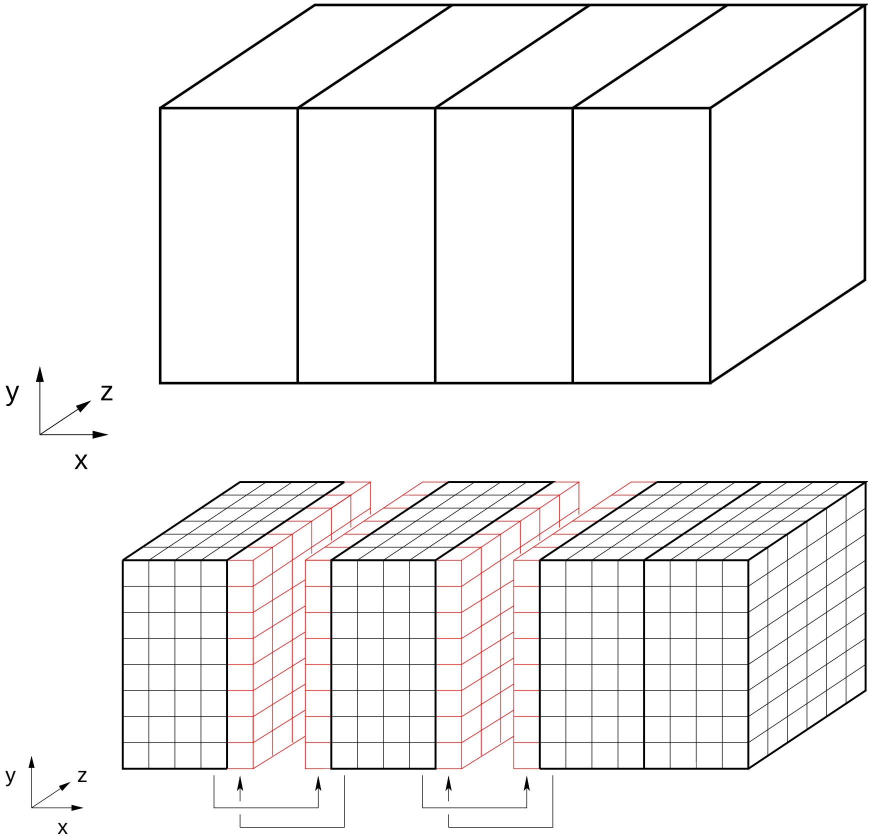
Remarks

- $|\nabla\phi| = 1$ ensures a unique solution for Level-Set function
- 'Classic' re-initialization (Sussman 1994) no longer required
- Surface tension representation improved, but not sufficient yet
- Immediate future: obtain higher order solution to $|\nabla\phi| = 1$

Parallelization of the code

- Parallel code required for meshes larger than 128^3 up to 512^3
- Approach: Message Passing Interface (MPI) library with domain decomposition
- Parallel Poisson solver
  - CG <u>without</u> pre-conditioner
  - Quality of initial guess important
- Code runs on SGI Origin 3800 or SGI Altix 3700 (Teras & Aster) at Sara

Parallelization: Domain decomposition



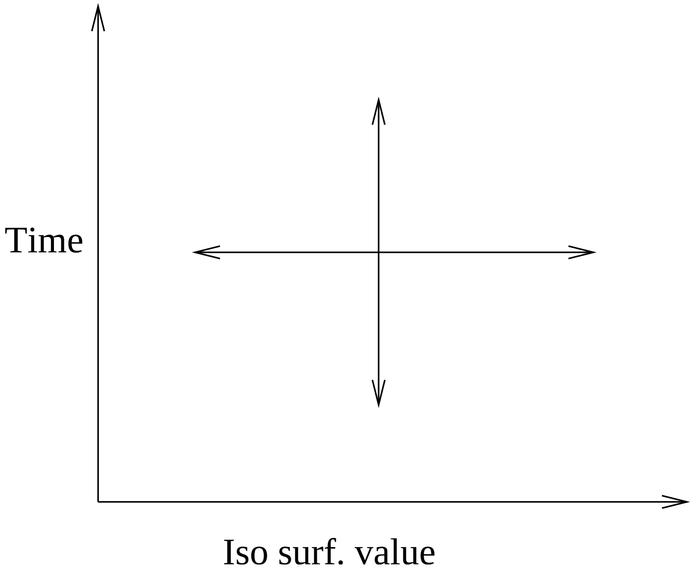
Parallel performance 128^3

- "Home made Beowulf cluster" with gigabit over copper
- Supercomputers TERAS/ASTER

#NCPU	Beowulf	Aster
1		44
2		40
4		30
8		17
16		7
32		8

Visualization

- Visualization of very large time-dependent data sets is a huge problem.
- To visualize the boundaries between fluids (*phase fronts*) we need
    - interactive isosurface extraction and rendering of large time-varying data sets.

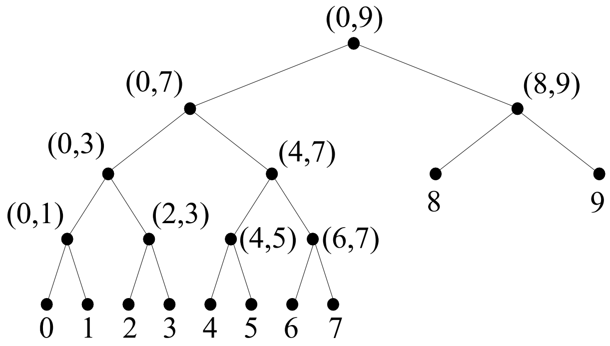


Data structure

A data structure for

- Interactive isosurface extraction
- Time-dependent data sets
- “Incremental” surfaces
- Use of temporal coherence
- Fast rendering
- No need to keep original data in memory

Temporal Hierarchical Index Tree (Shen, 1998)



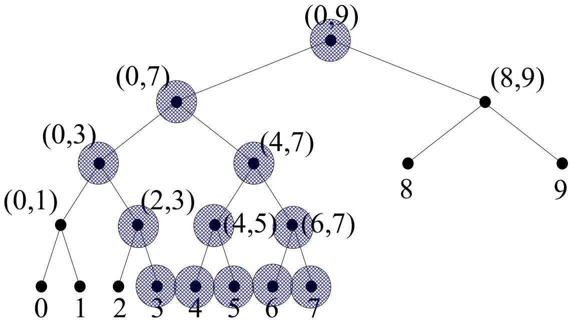
- Each node represents a certain time *range*.
- Each node contains “constant” cells for that time range.
- Cells in one node need not be stored below that node.
- The difference between (consecutive) time steps can be found by backtracking up the tree.
- In each node, a (possibly large) number of cells must be stored.

Out-of-core tree building

- During creation of an index tree, we need the entire *temporal evolution* of every cell, because we want to make use of temporal coherence as much as possible.
- Instead of using (x, y, z)-files, with each file representing a different time step, we use (x, y, t)-files
- All time-dependent data for a cell is in one single file.
- Split the data set in z-direction and create multiple trees.
- For example, for a 256^3 data set, we could create 8 trees of $256 \times 256 \times 32$.

Out-of-core visualization

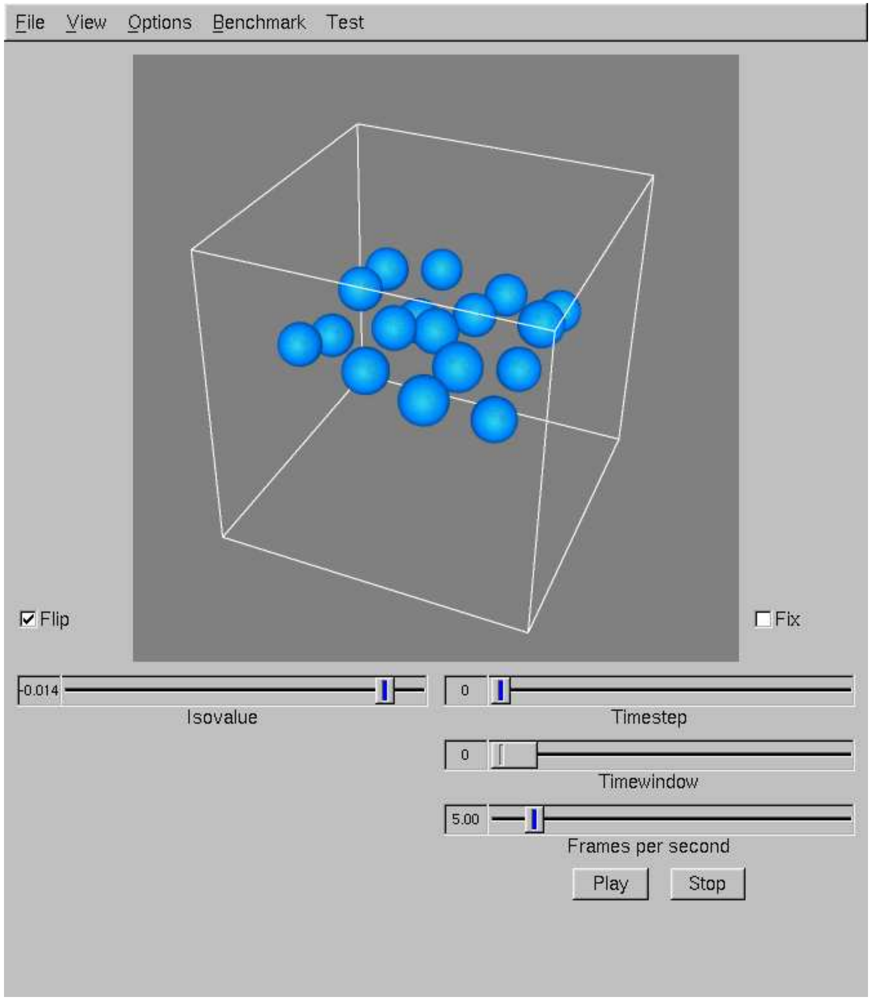
- During visualization, all sub-trees have to be read to reconstruct the entire *spatial* domain, but not complete.
- A *time window* in is kept main memory, centered around the current time step.
- This approach, alleviates the huge memory requirements for the visualization



Data sets

Data set	Bubbles		Clouds	
Resolution	256 × 256 × 256		128 × 128 × 80	
# Time steps	39		600	
Raw data size	4 992 MB		3 000 MB	
# THI Trees	16	8	6	8
xy-resolution	256 × 256	256 × 256	128 × 128	128 × 128
z-resolution	16	32	80	10
# Time steps	39	39	100	600
Total size	3 170 MB	1 630 MB	824 MB	750 MB

Visualization tool



☑ Flip

☐ Fix

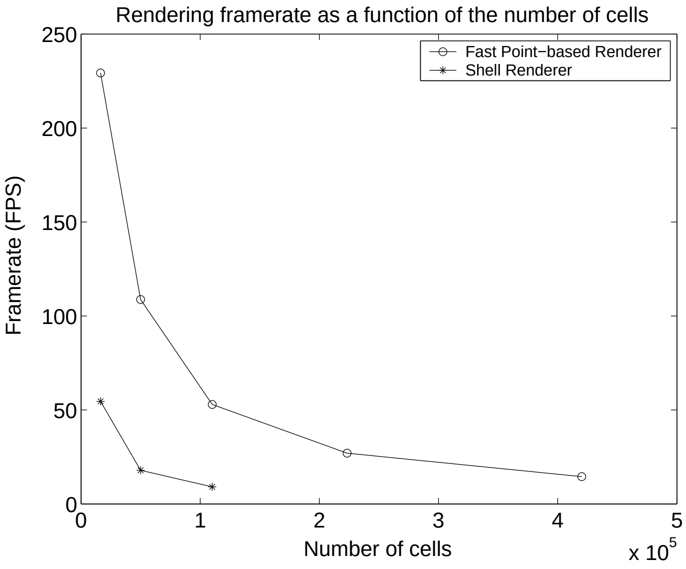
-0.014 Isovalue

0 Timestep

0 Timewindow

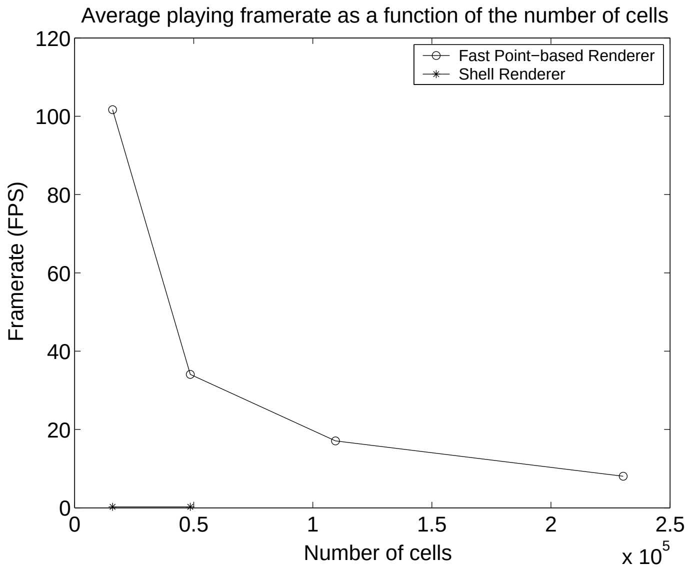
5.00 Frames per second

Rendering benchmarks



- 53 FPS for over $110,000$ cells
- 14 FPS for over $420,000$ cells
- Even 230 FPS for $16,000$ cells

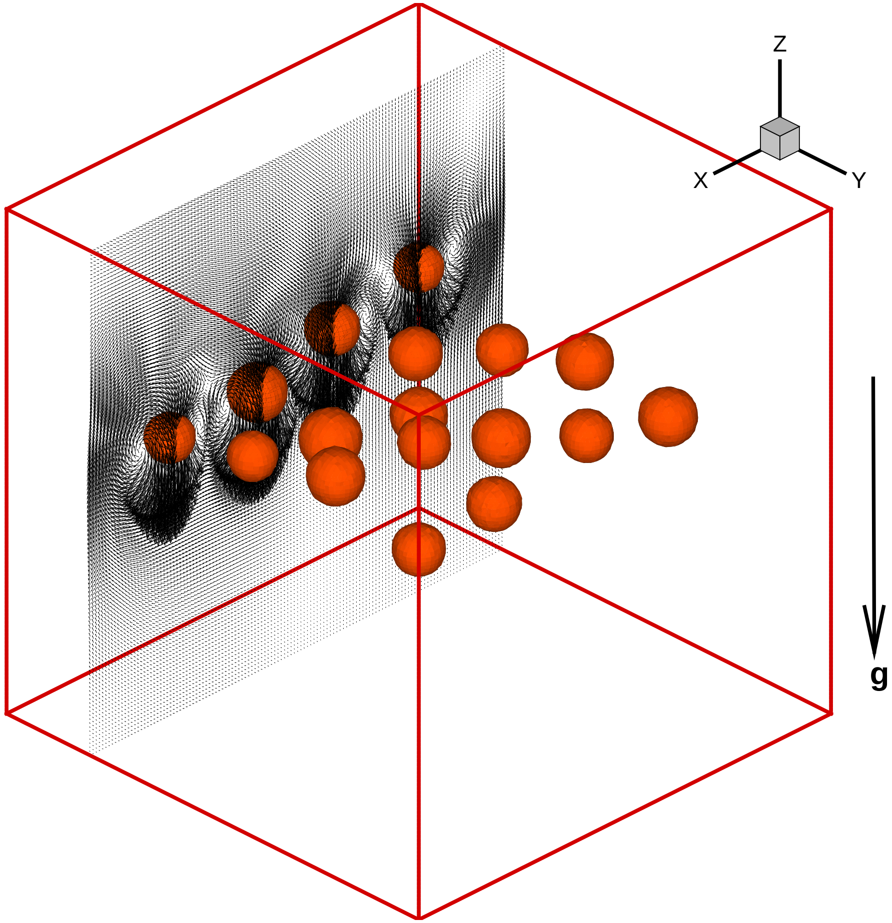
Playing benchmarks



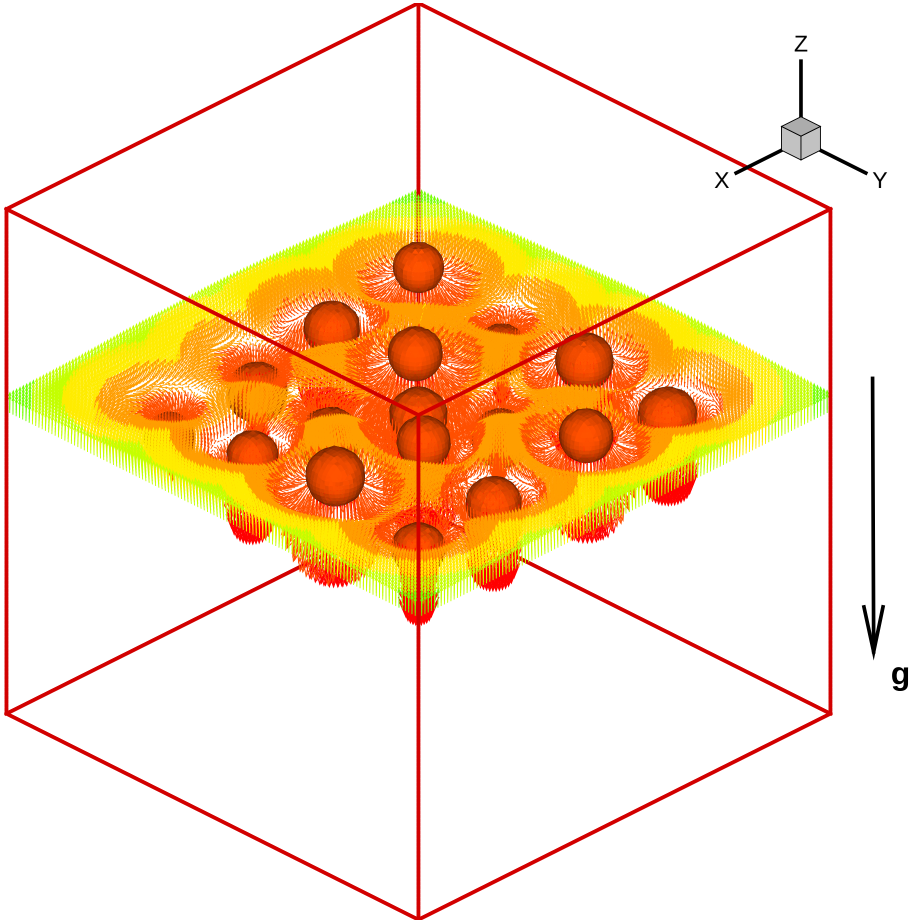
Extraction and rendering from all time steps

- 17 FPS (= time steps per second) for over $110,000$ cells
- 8 FPS (= time steps per second) for over $230,000$ cells
- As high as 101 FPS (= time steps per second) for $16,000$ cells

Some high resolution results 256^3 (1)



Some high resolution results 256^3 (2)



Closing remarks

- A mass conserving VOF/LS method has been developed
- Large scale flow simulations of complicated two-phase problems can be performed
- Collaboration between three disciplines has been very productive
- Scientific "freedom" of the NWO-CS program is very stimulating, new research lines can be developed.