Large-Scale Liquid Simulation on Adaptive Hexahedral Grids

Florian Ferstl, Rüdiger Westermann and Christian Dick
July 22, 2014
Introduction & Related Work

• Goal: Large-scale liquid simulation
  – Adaptive octree grid
  – Effective resolutions $\geq 1024^3$

• Challenges
  – Memory consumption
  – Consistent, adaptive discretization
  – Performance (bottleneck: pressure solve)

• Our method combines
  – Adaptive octree grid
  – Multigrid solver (for irregular, adaptive grid)
  – FEM discretization (hexahedral elements):
    
    **Element matrix based formulation**
Outline

- Adaptive Octree Grid
- FEM Discretization & element matrices
- Hanging Vertices
- Second Order Boundary Conditions
- Multigrid Solver
- Results
Adaptive Octree Grid

- Refinement strategy
  - Symmetric refinement band around surface (~5 cells wide)
  - Interior is held as coarse as possible
  - Octree is restricted to make resolution degrade smoothly towards interior
- Grid adapted after every surface advection step
Grid Example

- Uncoarsened octree (= uniform grid), Resolution: $256^3$

- Octree, Effective resolution: $256^3$
Outline

- Adaptive Octree Grid
- **FEM Discretization & element matrices**
- Hanging Vertices
- Second Order Boundary Conditions
- Multigrid Solver
- Results
Discretization

- **Navier-Stokes Equations**
  \[
  \dot{\mathbf{u}} = -\mathbf{u} \cdot \nabla \mathbf{u} + \frac{\mu}{\rho} \Delta \mathbf{u} - \frac{1}{\rho} \nabla p + \mathbf{g}
  \]

- **Pressure Poisson Equation**
  \[
  \frac{1}{\rho} \Delta p = \frac{1}{\Delta t} \nabla \cdot \mathbf{u}^* \quad \text{on } \Omega
  \]
  \[
  \rho \quad \text{prescribed on } \Gamma_{\text{Air}}
  \]
  \[
  \mathbf{n} \cdot \mathbf{u}_\Gamma \quad \text{prescribed on } \Gamma_{\text{Wall}}
  \]

- **Time splitting:**
  - **Advection:** Semi-Lagrange
  - **Diffusion (Optional):** FEM
  - **External forces:** FEM
  - **Projection:** FEM

- **Liquid surface tracked with (particle-) level-set**
Finite Element Discretization

- Tri-linear ansatz functions $\phi_i$ for $u$ and $p$
  $\rightarrow$ Co-located grid with all DOFs at cell vertices
- $\frac{1}{\rho} \Delta p = \frac{1}{\Delta t} \nabla \cdot \bm{u}^*$
  $\downarrow$
  $L_p = D\bm{u}^* + B(\bm{u}^* - \bm{u}_\Gamma)$
- Entries of linear operators FEM are given by integration over ansatz functions, e.g. for $L$:

$$(L)_{i,j} = \int_{\Omega} \nabla \phi_i \cdot \nabla \phi_j$$

$$= \sum_{e \in \Omega} \int_e \nabla \phi_i \cdot \nabla \phi_j$$

Contribution of $e$ to $L$
FEM Element Matrices

- Every FEM operator can be expressed as a sum of element matrices:

\[ L = \sum_e \hat{L}^e = \sum_e L^e \]

- \( L \) is sparse
- The \( \hat{L}^e \) are sparse and equal (up to a cell-size scaling factor and row/column permutation)
- Element matrices \( L^e \in \mathbb{R}^{8 \times 8} \)
  - A single representative can be analytically pre-computed
  - Entries of \( L \) can be assembled on the fly from this representative

\[ L = L^1 \oplus \ldots \oplus L^6 \]
Outline

- Adaptive Octree Grid
- FEM Discretization & element matrices
- Hanging Vertices
- Second Order Boundary Conditions
- Multigrid Solver
- Results
Hanging Vertices

- Hanging vertices at grid level transitions are constrained to tri-linearly interpolated values
  - New basis functions are linear combinations of original basis functions
  - Formally:
    - $I$: Interpolation operator (non-hanging vertices to all vertices)
    - $L'$: “unconstrained FEM operator” (treating hanging vertices as DOFs)
    - Hanging vertex elimination: $L = I^T L' I$

Split cell
Hanging Vertices

• Hanging vertex elimination: $L = I^T L' I$
• This can be done on element matrix level!
  - $L^{e'}$: relates the geometrically adjacent vertices of its cell
  - $L^e = (I^e)^T L^{e'} I^e$ (all $8 \times 8$ matrices in 3D)
  - $L^e$: relates the logically adjacent vertices of its cell

• Different hanging vertex configurations possible
  → Instead of one precomputed $L^e$, lookup table with 512 precomputed $L^e$'s
Outline

- Adaptive Octree Grid
- FEM Discretization & element matrices
- Hanging Vertices
- Second Order Boundary Conditions
- Multigrid Solver
- Results
Second Order Boundary Conditions

- Second order accurate BCs are essential for animation
- Ghost fluid method not applicable to the FEM discretization
- Our solution: An embedded interface method
  - Second order accurate
  - Currently only $\Gamma_{Air}$ treated with second order accuracy
Boundaries: Formulation

- Do not fix pressure at any vertices
- Add **penalty term** to enforce $p = 0$ at $\Gamma_{\text{Air}}$ in a variational sense
- **Restrict integration** to fluid filled portion of cells and the corresponding boundary:

\[
(L)_{i,j} = \int_{\Omega} \nabla \phi_i \cdot \nabla \phi_j + \int_{\Gamma_{\text{Air}}} [...], \quad (B)_{i,j} = \int_{\Gamma_{\text{Wall}}} [...]
\]

First order

Second order $\Gamma_{\text{Air}}$
Boundaries: Computation

- Changes only affect element matrices in boundary cells
- Compute those at runtime
  - Triangulate / tetrahedralize boundary cells using an extended, marching cubes style lookup table
  - Numerically integrate terms over the resulting tets / triangles
- Store element matrices explicitly (only) for boundary cells
Second Order Boundaries
Outline

- Adaptive Octree Grid
- FEM Discretization & element matrices
- Hanging Vertices
- Second Order Boundary Conditions
- Multigrid Solver
- Results
Multigrid Solver

- Galerkin-based coarsening
  - Define coarse grids
  - Define interpolation operators $I_{2h}^h$ (tri-linear interpolation)
  - Compute coarse grid operators according to
    \[ L_{2h} = (I_{2h}^h)^T L_h I_{2h}^h \]

- Solver performs standard V-Cycles
  - Gauss-Seidel smoother, PCG on coarsest level
  - MG efficient and stable if used as standalone solver, but even faster if used as preconditioner for CG
Element Matrix Coarsening

- Galerkin-based coarsening can be done on element matrix level!
  - All coarse grid operators can be written as sum of element matrices $L_i^e$
  - On coarser levels, these are purely mathematic constructs

- Each coarse grid element matrix can be computed from its children fine grid element matrices

$$L_0^0 = I_0^0 L_0^0 I_0^0 + I_1^0 L_0^1 I_1^0 + I_2^0 L_0^2 I_2^0$$
A few more things...

• Optional MG extension: Cell duplication
  – Complex, branching domains are often represented poorly on coarse grids (→ bad solver convergence)
  – Duplicate cells (and vertices) where necessary using a connectivity graph to track topology
• Parallelization: 90% parallelized (in terms of run-time on a single-core)
Outline

- Adaptive Octree Grid
- FEM Discretization & element matrices
- Hanging Vertices
- Second Order Boundary Conditions
- Multigrid Solver
- Results
Solver Convergence

- Simple Domain: Standing water in a box

- Complex Domain:
More Results

- **Fountain**: up to 34M adaptive cells, 25GB memory, effective resolution $1024^2 \times 3072$, < 5 min per time-step
- **4000 Drops**: up to 33M adaptive cells, 22GB memory, effective resolution $1024^3$, < 5 min per time step
Conclusion

- **Liquid simulation:**
  - Adaptive octree grid
  - Multigrid solver
  - FEM discretization

- **Limitations & future work**
  - Accurate wall boundaries
  - Grid refinement strategy
  - ...

Additional example: FLIP simulation (over 100M particles) with octree background grid
Thats all, thanks!