The 3D DSC in Fluid Simulation

Marek K. Misztal

Informatics and Mathematical Modelling, Technical University of Denmark
mkm@imm.dtu.dk

DSC 2011 Workshop
Kgs. Lyngby, 26th August 2011

Governing Equations

- The general equation governing fluid dynamics is the **Navier-Stokes** equation:

$$\rho\left(\frac{\partial\mathbf{u}}{\partial t}+(\mathbf{u}\cdot\nabla)\mathbf{u}\right)=-\nabla p+\nabla\cdot\mathbb{T}+\mathbf{f}.$$

Governing Equations

- The general equation governing fluid dynamics is the **Navier-Stokes** equation:

$$\rho\left(\frac{\partial \mathbf{u}}{\partial t}+(\mathbf{u}\cdot\nabla)\mathbf{u}\right)=-\nabla p+\nabla\cdot\mathbb{T}+\mathbf{f}.$$

- In many cases we are only interested in *incompressible flow*. Assumption that $\nabla\cdot\mathbf{u}=0$ leads to simpler equation:

$$\rho\left(\frac{\partial \mathbf{u}}{\partial t}+(\mathbf{u}\cdot\nabla)\mathbf{u}\right)=-\nabla p+\mu\nabla^2\mathbf{u}+\mathbf{f}.$$

Governing Equations

- The general equation governing fluid dynamics is the **Navier-Stokes** equation:

$$\rho\left(\frac{\partial \mathbf{u}}{\partial t}+(\mathbf{u}\cdot\nabla)\mathbf{u}\right)=-\nabla p+\nabla\cdot\mathbb{T}+\mathbf{f}.$$

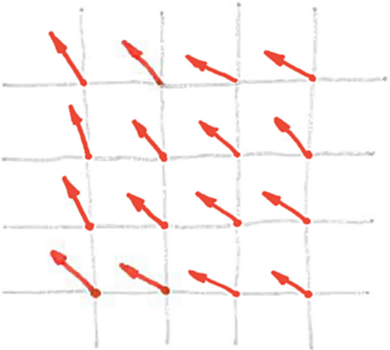
- In many cases we are only interested in *incompressible flow*. Assumption that $\nabla\cdot\mathbf{u}=0$ leads to simpler equation:

$$\rho\left(\frac{\partial \mathbf{u}}{\partial t}+(\mathbf{u}\cdot\nabla)\mathbf{u}\right)=-\nabla p+\mu\nabla^2\mathbf{u}+\mathbf{f}.$$

- Dropping viscosity term simplifies the equation even further, leading to **Euler equation**:

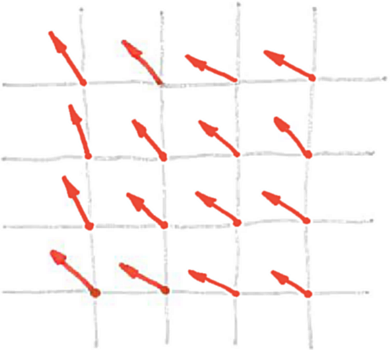
$$\rho\left(\frac{\partial \mathbf{u}}{\partial t}+(\mathbf{u}\cdot\nabla)\mathbf{u}\right)=-\nabla p+\mathbf{f}.$$

Eulerian Approach



- The free surface of the fluid is being tracked using *level set method*.

Eulerian Approach



- The free surface of the fluid is being tracked using *level set method*.
- N-S equation is usually solved using *fractional step method*.

Eulerian Approach

- This approach suffers from numerous drawbacks:

Eulerian Approach

- This approach suffers from numerous drawbacks:
    - rapid volume loss due to the numerical diffusion,

Eulerian Approach

- This approach suffers from numerous drawbacks:
  - rapid volume loss due to the numerical diffusion,
  - difficulty in handling solid boundaries which are not aligned with the grid,

Eulerian Approach

- This approach suffers from numerous drawbacks:
    - rapid volume loss due to the numerical diffusion,
    - difficulty in handling solid boundaries which are not aligned with the grid,
    - numerical errors introduce significant viscosity,

Eulerian Approach

- This approach suffers from numerous drawbacks:
  - rapid volume loss due to the numerical diffusion,
  - difficulty in handling solid boundaries which are not aligned with the grid,
  - numerical errors introduce significant viscosity,
  - lack of explicit free surface representantion (and hence, difficulty to include the surface energy),

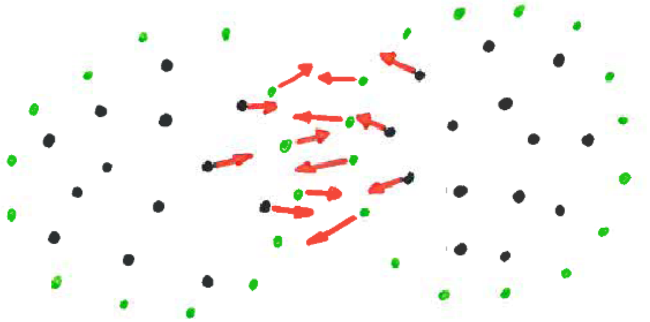
Eulerian Approach

- This approach suffers from numerous drawbacks:
  - rapid volume loss due to the numerical diffusion,
  - difficulty in handling solid boundaries which are not aligned with the grid,
  - numerical errors introduce significant viscosity,
  - lack of explicit free surface representantion (and hence, difficulty to include the surface energy),
  - lack of support of multiple phases.

Eulerian Approach

- This approach suffers from numerous drawbacks:
  - rapid volume loss due to the numerical diffusion,
  - difficulty in handling solid boundaries which are not aligned with the grid,
  - numerical errors introduce significant viscosity,
  - lack of explicit free surface representantion (and hence, difficulty to include the surface energy),
  - lack of support of multiple phases.
- Most of these drawbacks have been addressed in recent years, at the expense of simplicity of the original method.

Lagrangian Approach



Our Method

- Our finite element-based fluid solver is built upon an original, Lagrangian method for deformable interface tracking, the *deformable simplicial complex* (DSC).

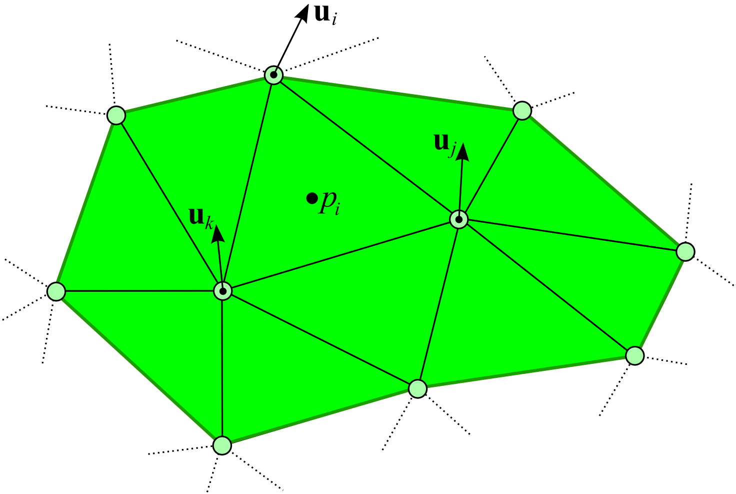
Our Method

- Our finite element-based fluid solver is built upon an original, Lagrangian method for deformable interface tracking, the *deformable simplicial complex* (DSC).
- Unlike in other fluid solvers using unstructured grid, the computational grid is not fixed or rebuilt at every time step, but it *evolves* over time, maintaining the fluid interface as a subcomplex (triangle mesh embedded in the grid).

Our Method

- Our finite element-based fluid solver is built upon an original, Lagrangian method for deformable interface tracking, the *deformable simplicial complex* (DSC).
- Unlike in other fluid solvers using unstructured grid, the computational grid is not fixed or rebuilt at every time step, but it *evolves* over time, maintaining the fluid interface as a subcomplex (triangle mesh embedded in the grid).
- We are going to show that our solver is intrinsically free of the shortcomings of the regular grid-based methods.

Setup



Method

- We treat the mesh triangles/tetrahedra as *conforming*, linear elements. Hence, the velocity field is defined as:

$$\mathbf{u}(\mathbf{x}) = \sum_{i=1}^{N_V} \mathbf{u}_i \cdot \phi_i(\mathbf{x}),$$

  where N_V is the number of vertices and ϕ_i is the linear interpolant (*hat* function).

Method

- We treat the mesh triangles/tetrahedra as *conforming*, linear elements. Hence, the velocity field is defined as:

$$\mathbf{u}(\mathbf{x}) = \sum_{i=1}^{N_V} \mathbf{u}_i \cdot \phi_i(\mathbf{x}),$$

  where N_V is the number of vertices and ϕ_i is the linear interpolant (*hat* function).
- Our solver resembles fractional step method in the sense that we separate advection from the other terms.

Method

- We treat the mesh triangles/tetrahedra as *conforming*, linear elements. Hence, the velocity field is defined as:

$$\mathbf{u}(\mathbf{x}) = \sum_{i=1}^{N_V} \mathbf{u}_i \cdot \phi_i(\mathbf{x}),$$

  where N_V is the number of vertices and ϕ_i is the linear interpolant (*hat* function).
- Our solver resembles fractional step method in the sense that we separate advection from the other terms.
- Advection is performed in purely Lagrangian way: velocity values “travel” together with the vertices.

Incompressibility

- We aim at keeping the velocity field divergence-free in each element:

$$\nabla \cdot \mathbf{u} = \sum_{i=1}^{N_V} \mathbf{u}_i \cdot \nabla \phi_i(\mathbf{x}) = 0.$$

$\nabla \phi_i$ is constant over every element.

Incompressibility

- We aim at keeping the velocity field divergence-free in each element:

$$\nabla \cdot \mathbf{u} = \sum_{i=1}^{N_V} \mathbf{u}_i \cdot \nabla \phi_i(\mathbf{x}) = 0.$$

  $\nabla \phi_i$ is constant over every element.
- By putting together incompressibility conditions for all tetrahedra we obtain a system of linear equations:

$$\mathbf{Pu} = \mathbf{0},$$

  where **P** is a discrete divergence operator (matrix).

Incompressibility

- Since the velocity field $\tilde{\mathbf{u}}$ after the advection step might not be divergence-free, we introduce pressure field $\mathbf{p}$, such that:

$$\mathbf{u} = \tilde{\mathbf{u}} + \mathbf{M}^{-1}\mathbf{P}^{T}\mathbf{p},$$

where $\mathbf{u}$ is divergece-free and $\mathbf{M}$ is the lumped mass matrix.

Incompressibility

- Since the velocity field $\tilde{\mathbf{u}}$ after the advection step might not be divergence-free, we introduce pressure field $\mathbf{p}$, such that:

$$\mathbf{u} = \tilde{\mathbf{u}} + \mathbf{M}^{-1}\mathbf{P}^T\mathbf{p},$$

  where $\mathbf{u}$ is divergece-free and $\mathbf{M}$ is the lumped mass matrix.
- Incompressibility condition $\mathbf{Pu} = \mathbf{0}$ yields that:

$$(\mathbf{PM}^{-1}\mathbf{P}^T)\mathbf{p} = -\mathbf{P}\tilde{\mathbf{u}}.$$

Surface Tension

- In order to make our fluid simulation more plausible we include *surface tension*. Surface tension is derived from surface energy U defined as:

$$U = \gamma A,$$

where γ is the surface energy density (material constant) and A is the free surface area.

Surface Tension

- In order to make our fluid simulation more plausible we include *surface tension*. Surface tension is derived from surface energy U defined as:
$$U = \gamma A,$$
where γ is the surface energy density (material constant) and A is the free surface area.
- Surface tension forces alone yield a highly divergent velocity field and our experiments have shown that integrating them before enforcing incompressibility step can give very poor results.

Optimization-based Approach

- In order to correctly incorporate surface tension forces we fully couple them with incompressibility by solving the optimization problem:

$$\begin{aligned} \text{minimize} \qquad & \frac{1}{2}(\mathbf{u}-\tilde{\mathbf{u}})^T \mathbf{M}(\mathbf{u}-\tilde{\mathbf{u}}) + U(\mathbf{x}+\mathbf{u}\Delta t), \\ \text{subject to} \qquad & \mathbf{P}\mathbf{u} = \mathbf{0}, \end{aligned}$$

where U is the surface energy.

Optimization-based Approach

- In order to correctly incorporate surface tension forces we fully couple them with incompressibility by solving the optimization problem:

$$\begin{array}{ll} \text{minimize} & \frac{1}{2}(\mathbf{u}-\tilde{\mathbf{u}})^T\mathbf{M}(\mathbf{u}-\tilde{\mathbf{u}})+U(\mathbf{x}+\mathbf{u}\Delta t), \\ \text{subject to} & \mathbf{P}\mathbf{u}=\mathbf{0}, \end{array}$$

  where U is the surface energy.
- The optimization problem is consistent, as the 1st order KKT condition is the backward Euler step:

$$\mathbf{u}=\tilde{\mathbf{u}}+\Delta t\mathbf{M}^{-1}\nabla U(\mathbf{u})+\mathbf{M}^{-1}\mathbf{P}^T\mathbf{p},$$

  where the pressure $\mathbf{p}$ plays the role of Lagrange multipliers.

Optimization-based Approach contd.

- By using the second order approximation of surface energy:

$$U(\mathbf{x}+\mathbf{u}\Delta t) \approx U(\mathbf{x}) + \Delta t \nabla U \cdot \mathbf{u} + \frac{\Delta t^2}{2}\mathbf{u}^T \nabla^2 U \mathbf{u}$$

we can further simplify the optimization problem to the form:

$$\begin{array}{ll} \text{minimize} & \frac{1}{2}\mathbf{u}^T(\mathbf{M}+\Delta t^2\nabla^2 U)\mathbf{u} + (-\tilde{\mathbf{u}}^T\mathbf{M}+\Delta t\nabla U)\mathbf{u}, \\ \text{subject to} & \mathbf{Pu} = \mathbf{0}, \end{array}$$

Optimization-based Approach contd.

- By using the second order approximation of surface energy:

$$U(\mathbf{x}+\mathbf{u}\Delta t) \approx U(\mathbf{x}) + \Delta t \nabla U \cdot \mathbf{u} + \frac{\Delta t^2}{2}\mathbf{u}^T \nabla^2 U \mathbf{u}$$

we can further simplify the optimization problem to the form:

$$\begin{aligned} &\text{minimize} && \frac{1}{2}\mathbf{u}^T(\mathbf{M}+\Delta t^2\nabla^2 U)\mathbf{u} + (-\tilde{\mathbf{u}}^T\mathbf{M} + \Delta t \nabla U)\mathbf{u}, \\ &\text{subject to} && \mathbf{P}\mathbf{u} = \mathbf{0}, \end{aligned}$$

- Now we can write the 1st order KKT condition as:

$$(\mathbf{M}+\Delta t^2\nabla^2 U)\mathbf{u} + \mathbf{P}^T\mathbf{p} = \mathbf{M}\tilde{\mathbf{u}} - \Delta t(\nabla U)^T.$$

KKT system

- Let us denote:

$$\begin{aligned}\mathbf{A} &= \mathbf{M} + \Delta t^2 \nabla^2 U, \\ \mathbf{b} &= \mathbf{M}\tilde{\mathbf{u}} - \Delta t (\nabla U)^T.\end{aligned}$$

KKT system

- Let us denote:

$$\begin{aligned}\mathbf{A} &= \mathbf{M} + \Delta t^2 \nabla^2 U,\\ \mathbf{b} &= \mathbf{M}\tilde{\mathbf{u}} - \Delta t (\nabla U)^T.\end{aligned}$$

- The KKT conditions for our *quadratic optimization problem* form a linear equation:

$$\begin{bmatrix} \mathbf{A} & \mathbf{P}^\mathrm{T} \\ \mathbf{P} & \mathbf{0} \end{bmatrix} \begin{bmatrix} \mathbf{u} \\ \mathbf{p} \end{bmatrix} = \begin{bmatrix} \mathbf{b} \\ \mathbf{0} \end{bmatrix}.$$

KKT system

- Let us denote:

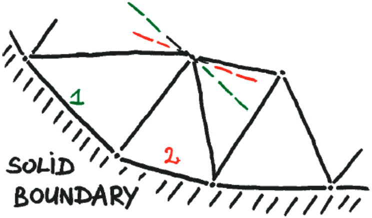
$$\begin{aligned} \mathbf{A} &= \mathbf{M} + \Delta t^2 \nabla^2 U, \\ \mathbf{b} &= \mathbf{M}\tilde{\mathbf{u}} - \Delta t (\nabla U)^T. \end{aligned}$$

- The KKT conditions for our *quadratic optimization problem* form a linear equation:

$$\begin{bmatrix} \mathbf{A} & \mathbf{P}^\mathrm{T} \\ \mathbf{P} & \mathbf{0} \end{bmatrix} \begin{bmatrix} \mathbf{u} \\ \mathbf{p} \end{bmatrix} = \begin{bmatrix} \mathbf{b} \\ \mathbf{0} \end{bmatrix}.$$

- Note that the KKT matrix is indefinite.

Locking



Locking means inability of a given finite element space to offer good approximate solutions, due to the fact that volume constraint on each tetrahedron may leave us with a solution space of very low dimension.

Pressure Stabilization

- To prevent it, we add a pressure stabilization term **S** to the equation:

$$\begin{bmatrix} \mathbf{A} & \mathbf{P}^T \\ \mathbf{P} & -\mathbf{S} \end{bmatrix} \begin{bmatrix} \mathbf{u} \\ \mathbf{p} \end{bmatrix} = \begin{bmatrix} \mathbf{b} \\ \mathbf{0} \end{bmatrix}. \qquad (1)$$

such that:

$$\mathbf{S}_{ij} = \begin{cases} -\delta \cdot a_{ij} & \text{if } i \neq j \\ \delta \cdot \sum_{k \neq i} a_{ik} & \text{if } i = j \end{cases}, \qquad (2)$$

where δ is a positive stabilization parameter and a_{ij} is the area of the face shared by tetrahedra i and j.

Pressure Stabilization

- To prevent it, we add a pressure stabilization term **S** to the equation:

$$\begin{bmatrix} \mathbf{A} & \mathbf{P}^T \\ \mathbf{P} & -\mathbf{S} \end{bmatrix} \begin{bmatrix} \mathbf{u} \\ \mathbf{p} \end{bmatrix} = \begin{bmatrix} \mathbf{b} \\ \mathbf{0} \end{bmatrix}. \tag{1}$$

  such that:

$$\mathbf{S}_{ij} = \begin{cases} -\delta \cdot a_{ij} & \text{if } i \neq j \\ \delta \cdot \sum_{k \neq i} a_{ik} & \text{if } i = j \end{cases}, \tag{2}$$

  where δ is a positive stabilization parameter and a_{ij} is the area of the face shared by tetrahedra i and j.
- Such stabilization term smoothens the pressure field in exchange for slight violation of incompressibility constraint, while globally preserving volume.

Pressure Stabilization

- To prevent it, we add a pressure stabilization term **S** to the equation:

$$\begin{bmatrix} \mathbf{A} & \mathbf{P}^T \\ \mathbf{P} & -\mathbf{S} \end{bmatrix} \begin{bmatrix} \mathbf{u} \\ \mathbf{p} \end{bmatrix} = \begin{bmatrix} \mathbf{b} \\ \mathbf{0} \end{bmatrix}. \qquad (1)$$

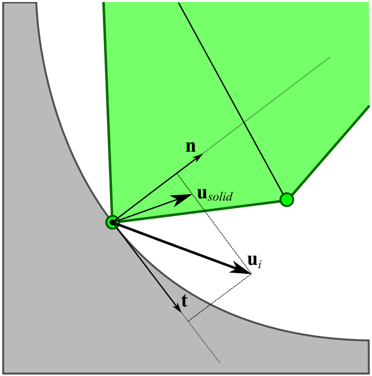
such that:

$$\mathbf{S}_{ij} = \begin{cases} -\delta \cdot a_{ij} & \text{if } i \neq j \\ \delta \cdot \sum_{k \neq i} a_{ik} & \text{if } i = j \end{cases}, \qquad (2)$$

where δ is a positive stabilization parameter and a_{ij} is the area of the face shared by tetrahedra i and j.

- Such stabilization term smoothens the pressure field in exchange for slight violation of incompressibility constraint, while globally preserving volume.
- In our simulations $\delta = \Delta t \cdot k^2$, where k is the speed of sound in the medium.

Solid boundaries



Solid boundaries put extra constraints on vertex velocity values. If the vertex v_i is in contact with the solid, we force its normal coordinate to match the normal coordinate of the solid velocity at the point of collision.

Viscosity

- More rigorous derivation of our finite element scheme can be found in: K. Erleben, M. K. Misztal and J. A. Bærentzen. *Mathematical Foundation of the Optimization-based Fluid Animation Method*. SCA 2011.

Viscosity

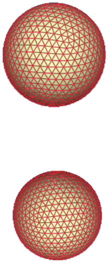
- More rigorous derivation of our finite element scheme can be found in: K. Erleben, M. K. Misztal and J. A. Bærentzen. *Mathematical Foundation of the Optimization-based Fluid Animation Method*. SCA 2011.
- Applying Galerkin method to the Navier-Stokes equation allows us to introduce viscosity into our scheme.

Viscosity

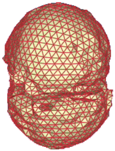
- More rigorous derivation of our finite element scheme can be found in: K. Erleben, M. K. Misztal and J. A. Bærentzen. *Mathematical Foundation of the Optimization-based Fluid Animation Method*. SCA 2011.
- Applying Galerkin method to the Navier-Stokes equation allows us to introduce viscosity into our scheme.
- This can be done by adding a $\Delta t\mathbf{D}$ term to the $\mathbf{A}$ matrix, where:

$$\mathbf{D}_{ij} = \int_{V_{fluid}} \mu(\nabla\phi_i{}^T\nabla\phi_j\mathbf{I} + \nabla\phi_i\nabla\phi_j{}^T)dV.$$

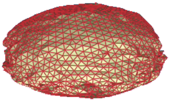
Examples



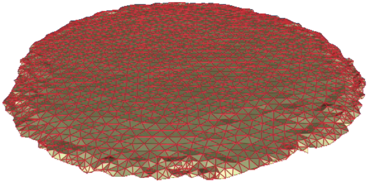
Examples



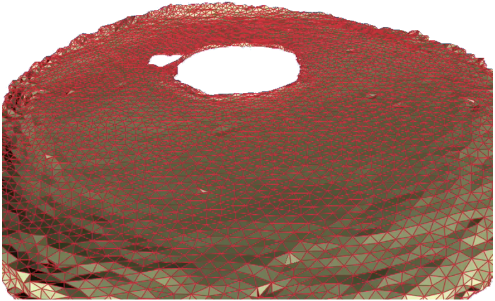
Examples



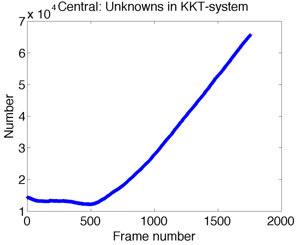
Examples

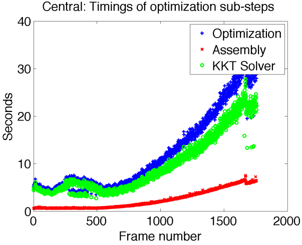


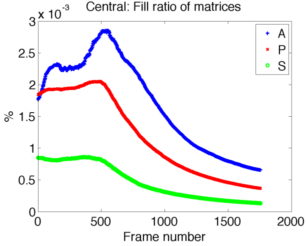
Examples



Performance







Implementation

- In the previous version of our fluid solver we had been using CHOLMOD (a Cholesky solver) in order to solve the KKT system.

Implementation

- In the previous version of our fluid solver we had been using CHOLMOD (a Cholesky solver) in order to solve the KKT system.
- Unfortunatelly, the complexity of Cholesky decomposition for sparse matrices is at best $O(n^{3/2})$, where n is the size of the matrix. This way, solving the KKT system quickly becomes the bottleneck.

Implementation

- In the previous version of our fluid solver we had been using CHOLMOD (a Cholesky solver) in order to solve the KKT system.
- Unfortunatelly, the complexity of Cholesky decomposition for sparse matrices is at best $O(n^{3/2})$, where n is the size of the matrix. This way, solving the KKT system quickly becomes the bottleneck.
- In order to avoid that we have recently switched the method to an iterative one: generalized minimal residual (GMRES) method.

Implementation

- In the previous version of our fluid solver we had been using CHOLMOD (a Cholesky solver) in order to solve the KKT system.
- Unfortunatelly, the complexity of Cholesky decomposition for sparse matrices is at best $O(n^{3/2})$, where n is the size of the matrix. This way, solving the KKT system quickly becomes the bottleneck.
- In order to avoid that we have recently switched the method to an iterative one: generalized minimal residual (GMRES) method.
- We are using CUDA-based implementation of GMRES, available in CUSP library.

Implementation

- In the previous version of our fluid solver we had been using CHOLMOD (a Cholesky solver) in order to solve the KKT system.
- Unfortunatelly, the complexity of Cholesky decomposition for sparse matrices is at best $O(n^{3/2})$, where n is the size of the matrix. This way, solving the KKT system quickly becomes the bottleneck.
- In order to avoid that we have recently switched the method to an iterative one: generalized minimal residual (GMRES) method.
- We are using CUDA-based implementation of GMRES, available in CUSP library.
- In order to improve the conditioning of the system, we multiply $\mathbf{A}$ and $\mathbf{b}$ by:

$$2 \cdot \frac{\|\mathbf{P}\|_\infty}{\|\mathbf{M}\|_\infty}.$$

Preconditioning

- Let us consider a linear system $\mathbf{K}\mathbf{x} = \mathbf{c}$

Preconditioning

- Let us consider a linear system $\mathbf{Kx} = \mathbf{c}$
- The convergence rate of many iterative solvers depends on the condition number κ of the matrix **K**:

$$\kappa = \left| \frac{\lambda_{\max}(\mathbf{K})}{\lambda_{\min}(\mathbf{K})} \right|,$$

where $\lambda_{\max}$ and $\lambda_{\min}$ are the maximum and minimum (by moduli) eigenvalues of **K**.

Preconditioning

- Let us consider a linear system $\mathbf{Kx} = \mathbf{c}$
- The convergence rate of many iterative solvers depends on the condition number κ of the matrix $\mathbf{K}$:

$$\kappa = \left| \frac{\lambda_{\text{max}}(\mathbf{K})}{\lambda_{\text{min}}(\mathbf{K})} \right|,$$

  where λ_{max} and λ_{min} are the maximum and minimum (by moduli) eigenvalues of $\mathbf{K}$.
- We can improve the condition number of the system by applying a preconditioner $\mathbf{T}$:

$$\mathbf{TKx} = \mathbf{Tc}.$$

Preconditioning

- Let us consider a linear system $\mathbf{Kx} = \mathbf{c}$
- The convergence rate of many iterative solvers depends on the condition number κ of the matrix $\mathbf{K}$:
$$\kappa = \left| \frac{\lambda_{\text{max}}(\mathbf{K})}{\lambda_{\text{min}}(\mathbf{K})} \right|,$$
where λ_{max} and λ_{min} are the maximum and minimum (by moduli) eigenvalues of $\mathbf{K}$.
- We can improve the condition number of the system by applying a preconditioner $\mathbf{T}$:
$$\mathbf{TKx} = \mathbf{Tc}.$$
- For our KKT system we are using the diagonal approximation of Murphy's block preconditioner (based on the Schur complement method):
$$\mathbf{T}^{-1} = \begin{bmatrix} \operatorname{diag}(\mathbf{M}) & 0 \\ 0 & \operatorname{diag}(\mathbf{P}^T\mathbf{M}^{-1}\mathbf{P} + \mathbf{S}) \end{bmatrix}$$

Future work

- Finishing implementation of viscosity.

Future work

- Finishing implementation of viscosity.
- More aggressive mesh quality improvement.

Future work

- Finishing implementation of viscosity.
- More aggressive mesh quality improvement.
- Parameter studies, in particular experimental evaluation of the time-step restrictions.