InterDyMFoam coupled with forces and mesh motion in 6-DoF

Erik Ekedahl

Outline

1 Introduction and motivation

2 Theory

3 Algorithm
- Usage
- Possible Alterations

4 Result

5 Discussion and Improvements

Motivation

- Volume-of-Fluid method and dynamic mesh with 6-degrees of freedom
- Marine Applications
- Base:interDyMFoam - Volume of Fluid.

Motivation

- Volume-of-Fluid method and dynamic mesh with 6-degrees of freedom
- Marine Applications
- Base:interDyMFoam - Volume of Fluid.

Motivation

- Volume-of-Fluid method and dynamic mesh with 6-degrees of freedom
- Marine Applications
- Base:interDyMFoam - Volume of Fluid.

Theory

- Two phases; gas and liquid in this case.
- Interface

$$\gamma = \begin{cases} 1 & \text{if in the water} \\ 0 < \gamma < 1 & \text{in the region around the surface} \\ 0 & \text{if in the air} \end{cases}$$

- Under-relaxation

$$\phi_{new} = \alpha\phi_{new} + (1-\alpha)\phi_{old} \tag{1}$$

Theory

- Two phases; gas and liquid in this case.
- Interface

$$\gamma = \begin{cases} 1 & \text{if in the water} \\ 0 < \gamma < 1 & \text{in the region around the surface} \\ 0 & \text{if in the air} \end{cases}$$

- Under-relaxation

$$\phi_{new} = \alpha\phi_{new} + (1-\alpha)\phi_{old} \tag{1}$$

Theory

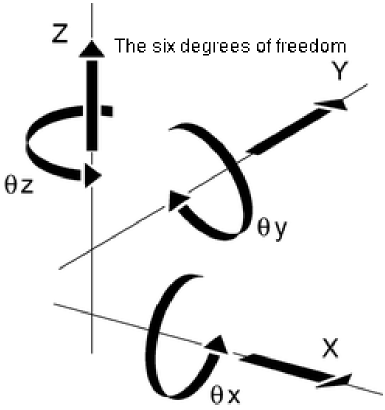
- Two phases; gas and liquid in this case.
- Interface

$$\gamma = \begin{cases} 1 & \text{if in the water} \\ 0 < \gamma < 1 & \text{in the region around the surface} \\ 0 & \text{if in the air} \end{cases}$$

- Under-relaxation

$$\phi_{new} = \alpha\phi_{new} + (1 - \alpha)\phi_{old} \tag{1}$$

Six Degrees of Freedom



ls

Figure: 6-DOF

Forces and Moments

-
$$\boldsymbol{F} = \int_S \boldsymbol{F}_{ext} + \boldsymbol{F}_{flow} \tag{2}$$
$$\boldsymbol{M} = \int_S \boldsymbol{M}_{ext} + \boldsymbol{M}_{flow} \tag{3}$$
- Forces from gravity, pressure and viscous effects

Forces and Moments

- $$\boldsymbol{F} = \int_S \boldsymbol{F}_{ext} + \boldsymbol{F}_{flow} \tag{2}$$
  $$\boldsymbol{M} = \int_S \boldsymbol{M}_{ext} + \boldsymbol{M}_{flow} \tag{3}$$
- Forces from gravity, pressure and viscous effects

Rotation and Translation

- *Quaternion* - Hamilton 1843
- Base: $\{1, i, j, k\}$ as $a1 + bi + cj + dk$
- Where a, b,c and d are real numbers and $i^2 = j^2 = k^2 = ijk = -1$
- Composed of one vector and one scalar in OF.
- Euler angles → quaternion and back.

Rotation and Translation

- *Quaternion* - Hamilton 1843
- Base: $\{1, i, j, k\}$ as $a1 + bi + cj + dk$
- Where a, b,c and d are real numbers and $i^2 = j^2 = k^2 = ijk = -1$
- Composed of one vector and one scalar in OF.
- Euler angles → quaternion and back.

Rotation and Translation

- *Quaternion* - Hamilton 1843
- Base: $\{1, i, j, k\}$ as $a1 + bi + cj + dk$
- Where a, b,c and d are real numbers and $i^2 = j^2 = k^2 = ijk = -1$
- Composed of one vector and one scalar in OF.
- Euler angles → quaternion and back.

Rotation and Translation

- *Quaternion* - Hamilton 1843
- Base: $\{1, i, j, k\}$ as $a1 + bi + cj + dk$
- Where a, b,c and d are real numbers and $i^2 = j^2 = k^2 = ijk = -1$
- Composed of one vector and one scalar in OF.
- Euler angles → quaternion and back.

Rotation and Translation

- *Quaternion* - Hamilton 1843
- Base: $\{1, i, j, k\}$ as $a1 + bi + cj + dk$
- Where a, b,c and d are real numbers and $i^2 = j^2 = k^2 = ijk = -1$
- Composed of one vector and one scalar in OF.
- Euler angles → quaternion and back.

mesh Deformation

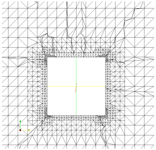


Figure: the *xy*-plane at time 0 seconds

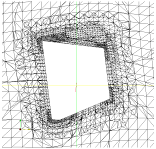


Figure: the *xy*-plane at time 16 seconds

Algorithm

```
while Time is running do
    Extract forces and moments from pressure, buoyancy.
    Extract forces and moments from viscous effects.
    Divide by mass for forces and inertia for momentum.
    Extract the acceleration by multiplication of the real time
    interval.
    Reset and move center of gravity.
    Translate patch to origo.
    Rotate in accordance to momentum.
    Translate back to origo.
    Translate according to forces.
end while
```

Algorithm

```
while Time is running do
    Extract forces and moments from pressure, buoyancy.
    Extract forces and moments from viscous effects.
    Divide by mass for forces and inertia for momentum.
    Extract the acceleration by multiplication of the real time interval.
    Reset and move center of gravity.
    Translate patch to origo.
    Rotate in accordance to momentum.
    Translate back to origo.
    Translate according to forces.
end while
```

Algorithm

```
while Time is running do
    Extract forces and moments from pressure, buoyancy.
    Extract forces and moments from viscous effects.
    Divide by mass for forces and inertia for momentum.
    Extract the acceleration by multiplication of the real time interval.
    Reset and move center of gravity.
    Translate patch to origo.
    Rotate in accordance to momentum.
    Translate back to origo.
    Translate according to forces.
end while
```

Algorithm

```
while Time is running do
    Extract forces and moments from pressure, buoyancy.
    Extract forces and moments from viscous effects.
    Divide by mass for forces and inertia for momentum.
    Extract the acceleration by multiplication of the real time
    interval.
    Reset and move center of gravity.
    Translate patch to origo.
    Rotate in accordance to momentum.
    Translate back to origo.
    Translate according to forces.
end while
```

Algorithm

```
while Time is running do
    Extract forces and moments from pressure, buoyancy.
    Extract forces and moments from viscous effects.
    Divide by mass for forces and inertia for momentum.
    Extract the acceleration by multiplication of the real time interval.
    Reset and move center of gravity.
    Translate patch to origo.
    Rotate in accordance to momentum.
    Translate back to origo.
    Translate according to forces.
end while
```

Algorithm

```
while Time is running do
    Extract forces and moments from pressure, buoyancy.
    Extract forces and moments from viscous effects.
    Divide by mass for forces and inertia for momentum.
    Extract the acceleration by multiplication of the real time
    interval.
    Reset and move center of gravity.
    Translate patch to origo.
    Rotate in accordance to momentum.
    Translate back to origo.
    Translate according to forces.
end while
```

Algorithm

```
while Time is running do
    Extract forces and moments from pressure, buoyancy.
    Extract forces and moments from viscous effects.
    Divide by mass for forces and inertia for momentum.
    Extract the acceleration by multiplication of the real time
    interval.
    Reset and move center of gravity.
    Translate patch to origo.
    Rotate in accordance to momentum.
    Translate back to origo.
    Translate according to forces.
end while
```

Algorithm

```
while Time is running do
    Extract forces and moments from pressure, buoyancy.
    Extract forces and moments from viscous effects.
    Divide by mass for forces and inertia for momentum.
    Extract the acceleration by multiplication of the real time interval.
    Reset and move center of gravity.
    Translate patch to origo.
    Rotate in accordance to momentum.
    Translate back to origo.
    Translate according to forces.
end while
```

Algorithm

```
while Time is running do
    Extract forces and moments from pressure, buoyancy.
    Extract forces and moments from viscous effects.
    Divide by mass for forces and inertia for momentum.
    Extract the acceleration by multiplication of the real time
    interval.
    Reset and move center of gravity.
    Translate patch to origo.
    Rotate in accordance to momentum.
    Translate back to origo.
    Translate according to forces.
end while
```

Algorithm

```
while Time is running do
    Extract forces and moments from pressure, buoyancy.
    Extract forces and moments from viscous effects.
    Divide by mass for forces and inertia for momentum.
    Extract the acceleration by multiplication of the real time
    interval.
    Reset and move center of gravity.
    Translate patch to origo.
    Rotate in accordance to momentum.
    Translate back to origo.
    Translate according to forces.
end while
```

Algorithm

```
while Time is running do
    Extract forces and moments from pressure, buoyancy.
    Extract forces and moments from viscous effects.
    Divide by mass for forces and inertia for momentum.
    Extract the acceleration by multiplication of the real time interval.
    Reset and move center of gravity.
    Translate patch to origo.
    Rotate in accordance to momentum.
    Translate back to origo.
    Translate according to forces.
end while
```

ForceFoamDict

forceBalance forceBalanceEnabled yes;

/* ———————————————————————— */

// Mass of the moving part //M M [1 0 0 0 0 0 0] 136.51; M M [1 0 0 0 0 0 0] 108000;

// Mass of the moving part inertia inertia [1 2 0 0 0 0 0] 648000;

// Limit the maximum acceleration on the system (Bandwidth limitation) accelLim accelLim [0 1 -2 0 0 0 0] 1.0e-2;

// Acceleration due to gravity g g [0 1 -2 0 0 0 0] 9.81;

// Direction vector of the acceleration due to gravity gVector (0 0 -1);

// Direction vector of the acceleration due to gravity CgCenter (0 0 0);

/* ———————————————————————— */

motionPatches (cube$_r$*egion*0);

Alterations

- Extract the archives.
- *my6DOFFoam*: run wclean then wmake
- *kubtest*: run application with 'my6DOFFoam'

Alterations

- Extract the archives.
- *my6DOFFoam*: run wclean then wmake
- *kubtest*: run application with 'my6DOFFoam'

Alterations

- Extract the archives.
- *my6DOFFoam*: run wclean then wmake
- *kubtest*: run application with ’my6DOFFoam’

Alterations

- Initial Values
- *accelLim* - Limits the motion due to the Forces and Moments.
- Under-relaxation in
- Mesh - *blockMesh* and *snappyHexMesh*

Alterations

- Initial Values
- *accelLim* - Limits the motion due to the Forces and Moments.
- Under-relaxation in
- Mesh - *blockMesh* and *snappyHexMesh*

Alterations

- Initial Values
- *accelLim* - Limits the motion due to the Forces and Moments.
- Under-relaxation in
- Mesh - *blockMesh* and *snappyHexMesh*

Alterations

- Initial Values
- *accelLim* - Limits the motion due to the Forces and Moments.
- Under-relaxation in
- Mesh - *blockMesh* and *snappyHexMesh*

Set up

- Domain - 40 meter kub with 6 meter cube in the origo
- BC's - slip and reflecting walls, movingWallVelocity for the cube
- Initial values

Set up

- Domain - 40 meter kub with 6 meter cube in the origo
- BC's - slip and reflecting walls, movingWallVelocity for the cube
- Initial values

Set up

- Domain - 40 meter kub with 6 meter cube in the origo
- BC's - slip and reflecting walls, movingWallVelocity for the cube
- Initial values

Result: cube at initial timestep

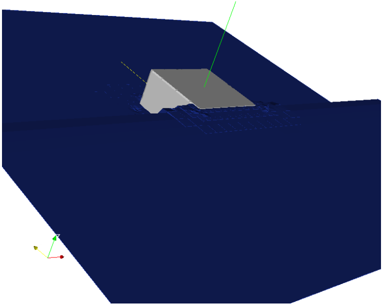


Figure: Time 0

Result: cube is tilted and begining to turn

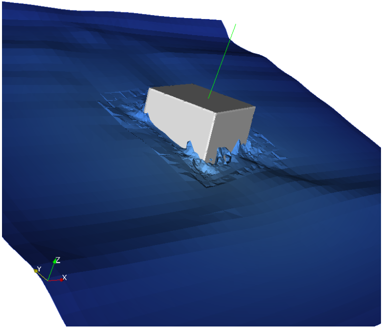


Figure: Time 6 seconds

Result: cube just before divergence

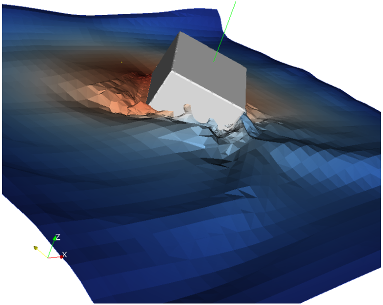


Figure: Time 10 seconds

Discussion

- This is a first draft
- Unstable without proper damping
- Skewed cells

Discussion

- This is a first draft
- Unstable without proper damping
- Skewed cells

Discussion

- This is a first draft
- Unstable without proper damping
- Skewed cells

Improvements

- Spring-damping
- Optimize under-relaxation
- Automate calculation of inertia and center of gravity

Improvements

- Spring-damping
- Optimize under-relaxation
- Automate calculation of inertia and center of gravity

Improvements

- Spring-damping
- Optimize under-relaxation
- Automate calculation of inertia and center of gravity