

# Store Separation with Sloshing

*The dynamic mesh and VOF models in FLUENT are used in this example to simulate the fall of a store partially filled with liquid. The trajectory and orientation of the falling body are computed using a six-degrees-of-freedom solver. A user-defined function is used to compute the changing center of gravity and moment of inertia throughout the fall. The sloshing of the liquid inside the store is shown to impact its motion.*

One of the interesting problems in the aerospace industry, especially for military vehicles, is the analysis of a store that is released from a high speed aircraft. In applications where the store contains either a liquid propellant or liquid cargo, the sloshing of the liquid inside the moving body becomes important. Due to sloshing, the moment of inertia (MI) of the body is no longer constant and has to be computed at each time step for an accurate trajectory calculation. In this example, a falling store half-filled with water is simulated. The volume of fluid (VOF) model in FLUENT is used to track the liquid interface inside the store. The dynamic mesh model with a special six-degrees-of-freedom (6DOF) solver is used to compute the trajectory and orientation of the object. An auxiliary user-defined function (UDF) is needed to account for the sloshing and its impact on the trajectory calculation.

The store is filled with air and water (50% of each by volume) and dropped at  $t=0$  from a moving

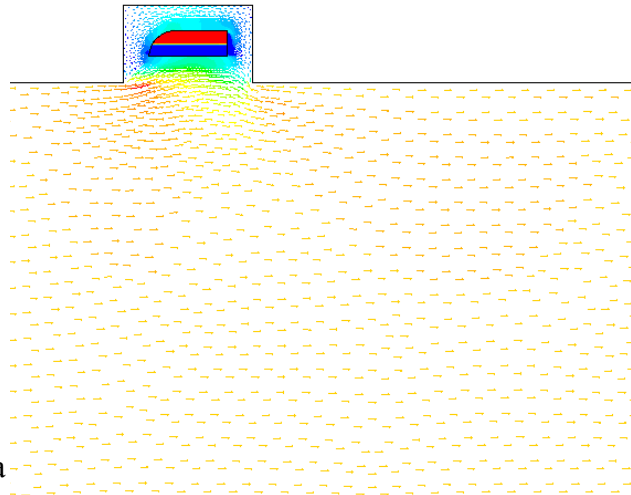


Figure 1: The initial steady state solution of the store and flowfield; blue corresponds to water inside the store

airplane flying at Mach 0.8. The initial condition used for this time-dependent simulation is the steady-state solution outside the store and wing before the time of release. As the object falls, it experiences a pressure force, a viscous drag force, and the gravitational force. These forces also create a moment on the object, causing it to rotate about its center of gravity (CG). The inputs for the 6DOF solver include the basic characteristics of the store, such as the location of the center of gravity, the store mass,

and components of the moment of inertia tensor. If the store has a constant density distribution, the components of the moment of inertia tensor are constant in the body-fixed coordinate system, and the CG does not change. In this case however, where sloshing takes place inside the store, additional calculations must be performed in order to update the MI and CG at each time step. It is for this purpose that the UDF is needed. This routine first calculates the center of gravity, using the distribution of water and air inside the store. Next, it computes the components of the moment of inertia tensor in the local coordinate system, with its origin at the CG.

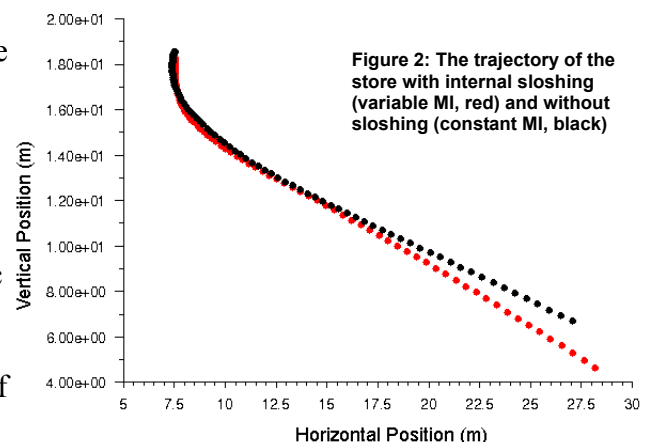


Figure 2: The trajectory of the store with internal sloshing (variable MI, red) and without sloshing (constant MI, black)

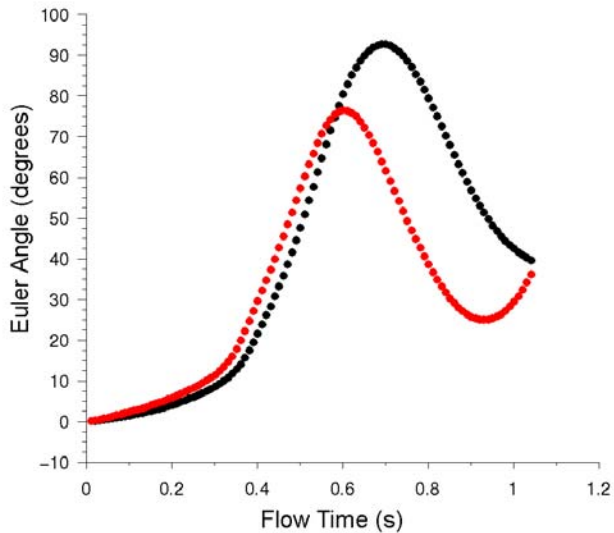


Figure 3: Euler angle as a function of time for the varying MI (red) and constant MI (black) cases

Using the updated CG location and MI values, a new position and orientation of the body are calculated with the 6DOF solver. For comparison, the trajectory calculations are also performed on the body with a constant moment of inertia, equal to that of the body half filled with water at  $t = 0$ .

Figure 1 shows the initial position of the store prior to its release. Inside the store, the volume fraction of air (with blue representing all water) is shown. The velocity distribution in the domain external to the store is also shown.

The trajectory of the store CG is illustrated in Figure 2 for the two cases considered. The case with liquid sloshing (and thus a variable MI) is shown in red, while the case without sloshing (and a constant MI) is shown in black. During the first half of the fall, sloshing does not have a strong impact on the trajectory, but during the second half of the time considered, the variable MI store advances farther than the

constant MI store in both the horizontal and vertical directions. The discrepancy in the trajectories for the two cases can be explained by the results shown in Figure 3, where the orientation of the store (Euler angle) as a function of time is plotted. Since this is a 2D case, the angle is that of the body about the z-axis, which is normal to the page. At

$t=0$ , the angle has a value of 0. As the store falls, it rotates in a counter-clockwise direction, corresponding to a positive Euler angle. The difference in the orientation of the body for the two cases results in a difference in the net drag, and this leads to differences in the trajectories. The orientation is similar during the first 0.6 seconds, causing the trajectories to be similar as well. After this time, however, the rotation of the sloshing store reverses direction, owing to the internal movement of the liquid.

The constant MI store continues to rotate until it reaches an angle of 90 degrees, after which its rotation direction reverses. During its fall, it spends more time oriented broadside to the direction

of travel, and therefore it experiences more drag.

Figure 4 shows the pressure distribution in the domain external to the store and the volume fraction of air (blue represents the water) inside the store at  $t = 0.6$  sec. To accurately capture the air-water interface, the geo-reconstruct scheme is used throughout the calculation. After 0.6 sec, the behavior of the two cases begins to diverge, and it is the result of the liquid motion in the store that causes this to happen.

In summary, a 6DOF calculation in FLUENT has been performed for a store separation example in which sloshing is taking place. The VOF model was used to track the air-liquid interface. A UDF was developed to compute the center of gravity and components of the moment of inertia tensor of the body at each time step. This information was then used to calculate the trajectory and orientation of the falling body.

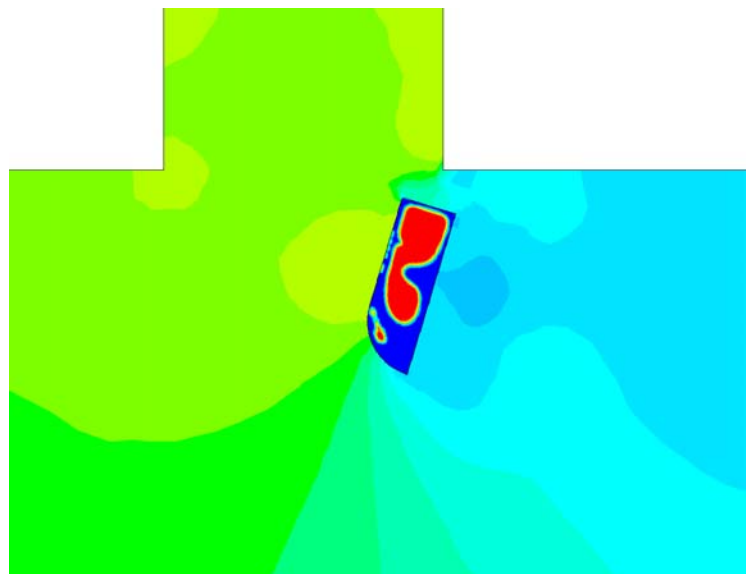


Figure 4 : Air volume fraction inside (blue represents water) and pressure contours outside the store