INTERNATIONAL JOURNAL FOR NUMERICAL METHODS IN FLUIDS, VOL. 7, 535–550 (1987)

# APPLICATION OF THE VOF METHOD TO THE SLOSHING OF A FLUID IN A PARTIALLY FILLED CYLINDRICAL CONTAINER

I. S. PARTOM

*RAFAEL (A.D.A), P.O. Box 2250, Haifa, Israel*

## SUMMARY

In a previous work we solved numerically the steady-state motion of an ideal fluid that fills a moving cylindrical container with partitions, and were able to compute the equivalent moments of inertia.

Here we extend this work in two steps. First we introduce time dependence and then free surfaces, and are able to compute the transient motion of the fluid not filling the container. The main body of the work has to do with the treatment of free surfaces. Our approach is an extention to three dimensions of the volume of fluid method of Hirt and Nichols. The solution algorithm is outlined, and two examples that demonstrate its capability are presented.

KEY WORDS Slosh Cylindrical Container Free Surface Volume of Fluid Numerical Method

## INTRODUCTION

In Reference 1 we developed a three-dimensional numerical code to solve the steady irrotational flow of an ideal fluid inside a moving cylindrical container with partitions. In this work we extend our code in two steps. We first introduce time dependence, and can simulate the fluid motion for time-varying container movement. We then introduce ullage, which means that the fluid does not fill the container, so that free surfaces are present. This would make our code useful for analysing practical situations. The time-dependent equations and their solution algorithm are outlined briefly in Reference 2. In the next section we repeat and expand the presentation. The main challenge of this work is, however, the inclusion of a free surface separating two immiscible fluids. We therefore devote the main part of the presentation to the free surface algorithm and examples.

The numerical simulation of fluid motion with a free surface is known to be difficult. The best way to simulate the motion of a boundary surface is by using the Lagrangian approach (the co-ordinate net moves with the fluid). But Lagrangian schemes are known to fail when the fluid body undergoes large distortions. For sloshing problems the Eulerian approach (the co-ordinate net is fixed with respect to the container) is therefore usually used. In trying to consider fluid motion with a free surface with Eulerian co-ordinates three main difficulties are encountered:

1. It is not obvious how the free surface (or for that matter, any other boundary surface) geometry should be described with respect to the fixed net. For two-dimensional problems several simple schemes suggest themselves. But the geometry becomes rather messy for three-dimensional problems.

0271–2091/87/060535–16$08.00

*Received 30 September 1985*
*Revised 1 August 1986*

2. It is not obvious how difference equations should be written for cells that are partly filled with fluid.
3. It is possible that for certain container motions, several free surfaces may exist in the fluid simultaneously. Also, new cavities may be formed and existing cavities may disappear. The solution algorithm should therefore be able not only to follow the initial free surface, but also to watch for the appearance of new surfaces and the disappearance of existing ones.

These difficulties were challenged for many years by several groups of workers. One approach was to use Lagrangian markers in an Eulerian net. This is called the 'marker and cell' (MAC) method.[3] Its main drawback is that it requires a very large computer. Another group of workers, headed by Hirt and Nichols from LANL, developed a different approach, named the volume of fluid (VOF) method.[4–7] In this approach the kinematics of boundary surfaces are not defined and computed exactly. Instead, a quantity **F** that measures the relative volume of fluid, is defined for each net cell. For full cells $F = 1$, for empty cells $F = 0$, and for boundary cells $0 < F < 1$. It is clear that the values of $F$ in a boundary cell and in neighbouring cells determine roughly the position and orientation of the boundary there. It is obvious that if interest is focused on the close neighbourhood of the free surface, the VOF approach would yield a very rough and probably unacceptable approximation. However, in the case of the internal sloshing motion under consideration, practical applications focus interest mainly on the gross motion of the whole body of fluid, and on the resultant force and moment it exerts on the container. We therefore conclude that it is appropriate to apply the VOF approach to our problem.

In their work Hirt and Nichols applied the VOF method to a two-dimensional geometry. We apply the method to a three-dimensional cylindrical geometry, which is far more complex. Much of the effort has therefore been devoted to algorithms that control the filling and emptying of cells without violating mass conservation and without diffusing the boundary.

In the next section we summarize the governing equations. Then the computational algorithms are outlined with emphasis on those related to free boundaries. The results of several fundamental test problems were computed in order to help debug the numerical code as well as the computational algorithms.

The results of several examples of computations are shown. From these it is possible to acquire a feeling for the capabilities and drawbacks of the VOF method.

## THEORY

Denoting the absolute velocity field of the fluid by **V** and the pressure field by $P$, Euler's equations for fluid motion are

$$\frac{\partial \mathbf{V}}{\partial t} + (\mathbf{V}\cdot\operatorname{grad})\mathbf{V} = -\frac{1}{\rho}\operatorname{grad} P + \mathbf{G}, \tag{1}$$

where **G** is the gravitational acceleration field, and

$$\operatorname{div}\mathbf{V} = 0. \tag{2}$$

Transforming to the container frame of reference we have the momentum equation

$$\frac{\partial \mathbf{U}}{\partial t} + (\mathbf{U}\cdot\operatorname{grad})\mathbf{U} = -\frac{1}{\rho}\operatorname{grad} P - \mathbf{A}_b - 2(\boldsymbol{\omega}\times\mathbf{U}) + \mathbf{G} \tag{3}$$

and the continuity equation

$$\operatorname{div}\mathbf{U} = 0, \tag{4}$$

$$\mathbf{U} = \mathbf{V} - \mathbf{V}_b, \tag{5}$$

where $\boldsymbol{\omega}$ is the angular velocity vector of the container and $\mathbf{V}_b$ is the velocity of a point moving with the container:

$$\mathbf{V}_b = \mathbf{V}_0 + \boldsymbol{\omega} \times \mathbf{R}, \tag{6}$$

where $\mathbf{R}$ is the radius vector and $\mathbf{V}_0$ is the translation velocity of the container.

$\mathbf{A}_b$ is the acceleration of a point moving with the container given by

$$\mathbf{A}_b = \mathbf{A}_0 + \boldsymbol{\gamma} \times \mathbf{R} + \boldsymbol{\omega} \times (\boldsymbol{\omega} \times \mathbf{R}), \tag{7}$$

where $\mathbf{A}_0$ is the translation acceleration and $\boldsymbol{\gamma}$ the angular acceleration of the container.

Taking the divergence of equation (3) and using equation (4) we have

$$\operatorname{div} \operatorname{grad} P = S_p(\mathbf{U}), \tag{8}$$

where

$$S_p(\mathbf{U}) = -\rho \operatorname{div} [(\mathbf{U} \cdot \operatorname{grad}) \mathbf{U} + \mathbf{A}_b + 2(\boldsymbol{\omega} \times \mathbf{U})]. \tag{9}$$

We refer to equation (8) as the pressure equation. The pressure equation is used to compute the pressure field once the velocity field has been obtained.

To start a flow problem we have to assume an initial flow field that is compatible with the boundary conditions. This flow field is generated in the following way. First, an initial arbitrary velocity field $\mathbf{V}_1$ is assumed. The usual assumption is that the fluid is at rest with respect to the inertial reference frame, i.e.

$$\mathbf{V}_1 = 0; \quad \mathbf{U}_1 = -\mathbf{V}_b. \tag{10}$$

Generally $\mathbf{U}_1$ does not satisfy the boundary conditions. Therefore a correction field $\mathbf{U}_c$ is added, so that

$$\mathbf{U} = \mathbf{U}_1 + \mathbf{U}_c. \tag{11}$$

By the continuity equation (4) we then have

$$\operatorname{div} \mathbf{U}_c = -\operatorname{div} \mathbf{U}_1 = -S_u, \tag{12}$$

where generally $S_u \neq 0$. We further assume that $\mathbf{U}_c$ is derivable from a potential $\phi_c$. This is equivalent to a rapid application of the boundary motion so that vorticity would not have enough time to develop. We thus have

$$\mathbf{U}_c = \operatorname{grad} \phi_c; \quad \operatorname{div} \operatorname{grad} \phi_c = -S_u. \tag{13}$$

Equation (13) is referred to as the velocity equation.

We also use the velocity equation during time integration as a means to control fluid mass (or volume) conservation. At the end of each time step we obtain a new flow field $\mathbf{U}'$. Because of various sources of error $\mathbf{U}'$ would not generally conserve fluid mass exactly, so that generally

$$S_u = \operatorname{div} \mathbf{U}' \neq 0 \tag{14}$$

To compensate for the unbalanced mass flow a correction field $\mathbf{U}_c$ is added, so that

$$\mathbf{U} = \mathbf{U}' + \mathbf{U}_c, \quad \operatorname{div} \mathbf{U} = 0, \quad \operatorname{div} \mathbf{U}_c = -\operatorname{div} \mathbf{U}' = -S_u, \tag{15}$$

and we solve for $\mathbf{U}_c$ by using the velocity equation (13).

For time integration an explicit scheme is used. This certainly bears upon stability and precision but simplifies numerical problems considerably. A time step starts by using equation (3) to compute the acceleration field $\partial \mathbf{U}/\partial t$ from the old velocity field and the old pressure field. From the

accelerations a new velocity field is obtained. This is corrected for exact mass conservation by applying the velocity equation (13). The corrected velocity field is then used in the pressure equation (8) to compute the new pressure field.

The velocity and pressure equations are Poisson's equations, and so need boundary conditions. For a full container the boundary conditions are straightforward:

$$(\mathbf{U}_c)_n = \frac{\partial \phi_c}{\partial n} = 0 \tag{16}$$

for the velocity equation, where $n$ is the direction normal to the container wall, and

$$(\text{grad } P)_n = -\rho[(\mathbf{U}\cdot\text{grad})\mathbf{U} + \mathbf{A}_b + 2(\boldsymbol{\omega}\times\mathbf{R})]_n. \tag{17}$$

For an unfilled container, mass conservation adjustment is enforced only for completely full cells, and the boundary conditions are somewhat different. When there is a full cell near the container wall we allow for the possibility that the fluid moves inwards (and a new free surface is about to be formed). This is done by taking $\phi_c = 0$ at a virtual cell outside the container whenever $u_n$ at the boundary points inwards. Similarly, when a full cell is near a boundary cell (a partly filled or empty cell) $u_n \neq 0$ and we take $\phi_c = 0$ at the boundary cell.

For an unfilled container we solve the pressure equation for full as well as for partly filled cells. To do this we modify the pressure equation (9) by introducing the fluid volume function $F$. We thus use

$$\text{div grad } P = FS_p \tag{18}$$

and we take $P = 0$ at the empty cells. Whenever a partly filled cell is near the container wall we take $P = 0$ at a virtual cell outside the container.

The normal velocity components on cell boundaries between two partly filled cells are treated in the same way as those between full cells. Whenever there is a partly filled cell near an empty cell the normal velocity component on their common boundary is set to zero. This enables us to compute the gradients of velocity components needed in the pressure and momentum equations.

Finally, we need to decompose the gravity vector $\mathbf{G}$ into its components in the reference frame of the moving container. We do this by using the transformation equation

$$\left(\frac{\partial \mathbf{G}}{\partial t}\right)_{\text{inertial system}} = \left(\frac{\partial \mathbf{G}}{\partial t}\right)_{\text{moving system}} + \boldsymbol{\omega}\times\mathbf{G}. \tag{19}$$

But

$$\left(\frac{\partial \mathbf{G}}{\partial t}\right)_{\text{inertial system}} = 0. \tag{20}$$

Therefore

$$\left(\frac{\partial \mathbf{G}}{\mathrm{d}t}\right)_{\text{moving system}} = -\boldsymbol{\omega}\times\mathbf{G} = \mathbf{G}\times\boldsymbol{\omega} \tag{21}$$

We time integrate equation (21) and obtain the time variations of the components $G_x$, $G_y$ and $G_z$ in the container reference system.

## VOLUME OF FLUID UPDATING

As mentioned in the introduction, the VOF method developed by Hirt and Nichols has been presented by them for two-dimensional geometries.[4] In this work it is adapted to three-dimensional geometries.

Following Hirt and Nichols we define for each cell a volume of fluid function $F$:

$$F = \frac{m_c}{\rho V_c} \tag{22}$$

where $m_c$ is the mass of fluid in the cell and $V_c$ is the volume of the cell. Clearly, for an empty cell $F = 0$, for a full cell $F = 1$ and for a partly filled cell $0 < F < 1$. A partly filled cell is called a boundary cell.

For two-dimensional geometries Hirt and Nichols approximate the fluid free surface by a straight line crossing the boundary cell. They show how to determine the position and orientation of the boundary line in a cell from the values of $F$ in that cell and in neighbouring cells.

This is done by first approximating grad $F$ from the values of $F$ in neighbouring cells. The orientation of the normal to the boundary line coincides with the direction of grad $F$. The position of the boundary line is then determined from the geometry of a line of known orientation cutting a given cell.

This procedure can be extended to three-dimensional geometry, where a boundary plane replaces the boundary line. The complexity and amount of computations would, however, be much larger.

Hirt and Nichols compute the position and orientation of the boundary line in a boundary cell for only one purpose: to determine by interpolation the pressure at the centre of a boundary cell from the pressure boundary condition and the pressure in a nearby full cell. We chose not to adopt this approach for our three-dimensional geometries, because interpolation in a three-dimensional net is not straightforward, and because it calls for a large volume of computations. Instead we chose to compute the pressure by modifying the pressure equation according to equation 18. In this way we automatically take care of the three-dimensional interpolation, and also avoid the necessity of determining the position and orientation of the boundary plane in each boundary cell. We did not, however, compare the two approaches in terms of accuracy. This comparison has yet to be performed.

The most intricate part of the VOF method is the algorithm for updating $F$ at or near boundary cells. Generally the updating is effected by the conservation of fluid volume (or mass) equation:

$$\frac{\Delta F}{\Delta t} = \frac{1}{V_c} \sum_{\text{on cell boundaries}} F_d u_n A_d, \tag{23}$$

where $F_d$ is the volume of fluid transmitted through the cell boundary relative to the considered cell volume, $u_n$ is the average fluid velocity normal to the cell boundary, $A_d$ is the cell boundary area and $V_c$ is the cell volume. But it is not obvious what to take for $F_d$.

Hirt and Nichols showed how to determine $F_d$ for two-dimensional geometry in a way that would solve two problems: (a) ensure fluid volume conversation and (b) avoid dispersion of the boundary. In our preliminary computations we realized that boundary dispersion is indeed a severe problem. If not strictly checked by proper choice of $F_d$ the boundary tends to disperse, and all the empty cells very quickly become partly filled.

We extended Hirt and Nichols' approach for determining $F_d$ to three-dimensional geometry as follows. Consider two adjacent cells as in Figure 1. $F_d$ for the common boundary of these two cells depends first on the value of $F_1$ (the donor cell): for $F_1 = 0$ we have $F_d = 0$; for $F_1 = 1$ we have $F_d = 1$. The case of $0 < F_1 < 1$ is subdivided into three branches according to the value of $F_2$.

For $F_2 = 0, F_d$ is determined according to the general direction of the fluid boundary in cell 1. The general direction could be either 'parallel' or 'perpendicular' to the cell boundary, as shown in Figure 2.

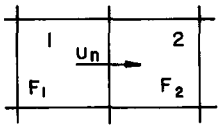


Figure 1. Schematics of two adjacent cells

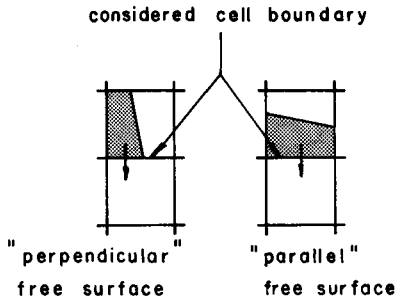


Figure 2. 'Parallel' and 'perpendicular' general directions of the fluid boundary

The general direction of the fluid boundary in cell 1 is determined from the values of $F$ in its four neighbouring cells, as shown in Figure 3.

When one or more of the neighbouring cells is empty the fluid boundary in cell 1 is 'perpendicular' to the considered cell boundary. When none of the neighbouring cells is empty, the fluid boundary in cell 1 is 'parallel' to the considered cell boundary.

For the 'perpendicular' general direction we thus have $F_d = F_1$, and for the 'parallel' $F_d = 0$. For $F_2 = 1$ we always have $F_d = F_1$.

For the case $0 < F_2 < 1$ we also determine first the general direction of the fluid boundary in cell 1. Then, for the 'perpendicular' general direction we have $F_d = F_1$ and for the 'parallel' direction $F_d = 1$.

All this does not prevent boundary dispersion in all situations. To avoid numerical dispersion it is necessary to prevent fluid in a partly filled cell from spilling into an empty cell. We do this by simply setting to zero the velocity $u_n$ normal to the boundary between two such cells. In doing this we must take care, however, to exclude two rarely occurring cases: (a) when the fluid is just crossing a cell edge and (b) when the fluid boundary is just crossing a cell corner. These two situations are represented in Figure 4. We see that for these two rarely occurring cases it is essential not to prevent the fluid from spilling into the empty cell.

The complete algorithm for checking whether an empty cell can be filled or a full cell can be emptied, from the point of view of avoiding boundary dispersion, calls for considerable programming sophistication. We chose not to outline the details of this algorithm here.

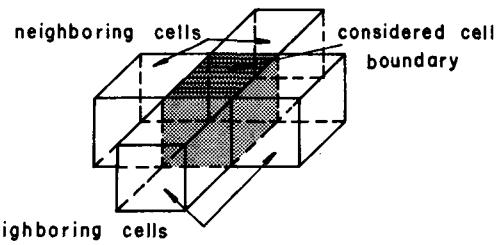


Figure 3. Representation of the four neighbouring cells to a given cell boundary

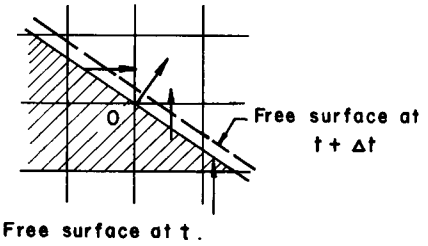


Figure 4. Schematics of a case where the fluid boundary is just crossing a cell edge or corner

Finally, there is the problem of overfilling and overemptying. It may happen that during a time step $\Delta t$ a cell that is almost full would be overfilled. What is done in such a case (and also in the parallel case of overemptying) is to spill the overfill into neighbouring empty or partly filled cells.

## NUMERICAL SCHEME

The finite difference net is the same one described in Reference 1 (Figure 1 there). Briefly, the cells are formed by the intersection of radial planes, horizontal planes and cylindrical surfaces. Along the container axis there are cylindrical cells. The scalar functions $P$ and $\phi_c$ are defined at the cell centres. The velocity components are defined on cell boundaries, so that we have a staggered three-dimensional network.

To write the finite difference approximation for the momentun equation (3) we first define

$$\mathbf{F} = \mathbf{A}_b + 2(\boldsymbol{\omega} \times \mathbf{U}), \tag{24}$$

$$\mathbf{T} = (\mathbf{U} \cdot \text{grad})\mathbf{U}. \tag{25}$$

The component of $\mathbf{F}$ and $\mathbf{T}$ in cylindrical co-ordinates are given in Reference 1 as well as their difference approximations. The difference approximation for the term grad $P$ is the usual central difference expression.

For the cylindrical cells at the centre Cartesian co-ordinates are used. A certain amount of care is needed to match the Cartesian components in the inner cell to the cylindrical components in the outer cells.

As mentioned above, a time cycle starts by using the momentun equation to compute the fluid acceleration and from it the velocity field. The velocity field is then corrected to enforce exact mass conservation by solving the velocity equation in full cells. The finite difference form of the velocity equation is given in Reference 1, where its solution procedure is also outlined. It is an iterative line relaxation with a single fixed point. What is different when there are unfilled cells is that the lines (in the $z$-direction) are partly incomplete. This difficulty is easily overcome by inserting virtual equations ($\phi_c = 0$) for the partly filled and empty cells.

After the corrected velocity field is obtained, the VOF function $F$ is updated by the procedures described previously. If there is an overspilling or underemptying in the course of updating we once again go through the velocity field correction procedure to ensure exact mass conservation.

The next step is to compute the pressure field by solving the pressure equation. The difference approximation of the right-hand side of the pressure equation is given in detail in Reference 1, as well as the finite difference representation of the equation itself. The solution procedure is also outlined there. Similar to that of the velocity equation, it is solved by line relaxation in the axial direction. To avoid divergence in the singular case of a full container, a constant value is subtracted from all the cells after each iteration. We do this by looking for the lowest pressure value in the field,

and this value is subtracted from all the cells. Recall that the pressure equation is solved only for full and partly filled cells. The pressure at the container wall is extrapolated from the pressure values in the cell. We then integrate over the container wall and obtain the resultant force and moment exerted by the fluid on the container. The detailed equations for the procedure are also outlined in Reference 1. At this stage the computation is ready for a new time cycle.

The scheme just outlined was programmed for a CDC Cyber-175 computer. Because of limitations in fast memory capacity, the maximum net used was $6 \times 14 \times 60$ cells in the $r, \theta$ and $z$ directions, respectively. Many computations were run with a sparse net of $3 \times 6 \times 16$ cells, and we were able to show that even for such a sparse net the results are acceptable from the point of view of stability and accuracy.

In a typical run with the sparse net a time cycle requires about 6 s of CPU time. In the examples that follow around 100 time cycles were needed to complete a problem, which amounts to about 600 s CPU time per problem.

In the next section several examples are described and discussed.

## COMPUTED EXAMPLES

To check the performance of the computer code, several simple problems, which we call 'zero computations', were run. In these computations the boundary conditions are such that almost nothing happens to the fluid, and it is easy to check whether indeed nothing happens. Some of the 'zero computations' were

(a) a full container and steady boundary condition, and comparison with the results obtained in Reference 1
(b) a partly filled container, where the free surface is in the $z$-direction, no gravitation and the container translates with a constant velocity in the $z$-direction
(c) a half filled container where the free surface is in the $z$-plane, no gravitation, and the container rotates with a constant angular velocity around the $z$-axis
(d) a partly filled unmoving container in the $z$-direction and the fluid free-falls by the influence of gravity.

In all these zero computations misbehaviours of the flow, especially near boundaries, were located, and the numerical algorithm corrected accordingly.

We then proceeded to compute meaningful examples. In what follows we present results for two of them.

*Example 1: fluid motion in a horizontal unmoving container by the influence of gravity*

The initial conditions for this problem are shown in Figure 5. There are two cases. In case a the container is half filled, and in case b three-quarters filled.

The container is horizontal and gravitation is in the $x$-direction. At $t = 0$ the diaphragm holding the fluid is removed and the fluid flows into the empty part of the container. The problem has symmetry with respect to the $xz$-plane, which enables us to check to what extent the code preserves this symmetry. In the preliminary runs this symmetry was not preserved. However, after the causes of asymmetry were corrected we obtained identical results on the two sides to within five significant digits.

The results obtained for case a are shown in Figure 6, in which two sections of the container are shown at five different times. In the boundary cells we inserted the values of the VOF function $F$. With the aid of the $F$ values we were able to draw the approximate location of the free surface.

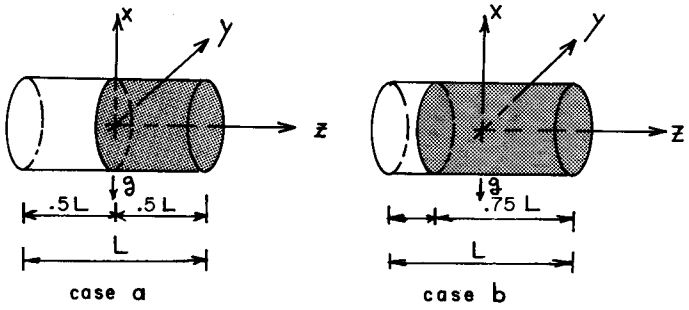


Figure 5. Initial conditions for example 1: (a) half-filled container; (b) three-quarters filled container

It is possible to see that the fluid surface is not 'smooth'. There are some bubbles under the surface and some splashes above it. We do not claim that these bubbles and splashes are real. They may well be numerical artefacts that result from deficiencies in our VOF algorithm. We intend to look into this more closely in the future. However, it should be borne in mind that usually the main interest is in computing the resultant force and moment exerted by the fluid in the container. It would seem that these integral quantities are not influenced by numerical debris at the free surface.

In Figure 8 we show the fluid free surface in the $zx$-plane at five different times. It is possible to deduce from this Figure the speed of spilling of the fluid by the influence of gravity.

The results obtained for case b are shown in Figure 7. In this case it is possible to observe the interaction of the moving fluid with the container end boundary without spending too much computer time. In Figure 10 the free surface contours at different times in a longitudinal section are shown.

As mentioned before, we are interested mainly in the resultant force **Q** and moment **M** on the container. The components different from zero in the example under consideration are $Q_x, Q_z$ and $M_y$. Their values for cases a and b are plotted in Figures 9 and 11, respectively.

From simple considerations it can be deduced that when steady state is attained (by the aid of viscosity neglected in our computations) $Q_z$ and $M_y$ tend to zero by symmetry, and $Q_x$ to the total fluid weight. This tendency is indeed observed in the results, especially for case b.

*Example 2: the container moves around a horizontal axis*

In case a the container has its upper layer of cells empty and its motion is given by $\omega_x = 1$ rad/s for $t > 0$. Gravity is ignored. The force and moment components different from zero are, in this case, $Q_x, Q_z$ and $M_y$. Longitudinal and transverse cross-sections at different times with values of $F$ and the free surface line are shown in Figure 12. Curves of $Q_x, Q_z$ and $M_y$ are given in Figure 13.

In case b the motion is $\dot{\omega}_x = 1$ rad/s$^2$ for $t > 0$, and gravity is ignored. The results of the initial time cycle enable us (as in Reference 1) to compute the equivalent moment of inertia by

$$I_{xe} = M_x/\dot{\omega}_x. \tag{26}$$

These results are plotted as a function of the degree of filling $V_r$(volume of fluid/volume of container), in terms of the relative moment of inertia $I_{xr}$, in Figure 14. $I_{xr}$ is defined by

$$I_{xr} = I_{xe}/I_{xx}, \tag{27}$$

where $I_{xx}$ is the moment of inertia of a rigid fluid. The $L/D$ ratio in this case is 16/7. We see that the case of the full container is singular. For a partly filled container there is a shallow maximum around $V_r = 0{\cdot}6$, and $I_{xr}$ there is around 0·62.

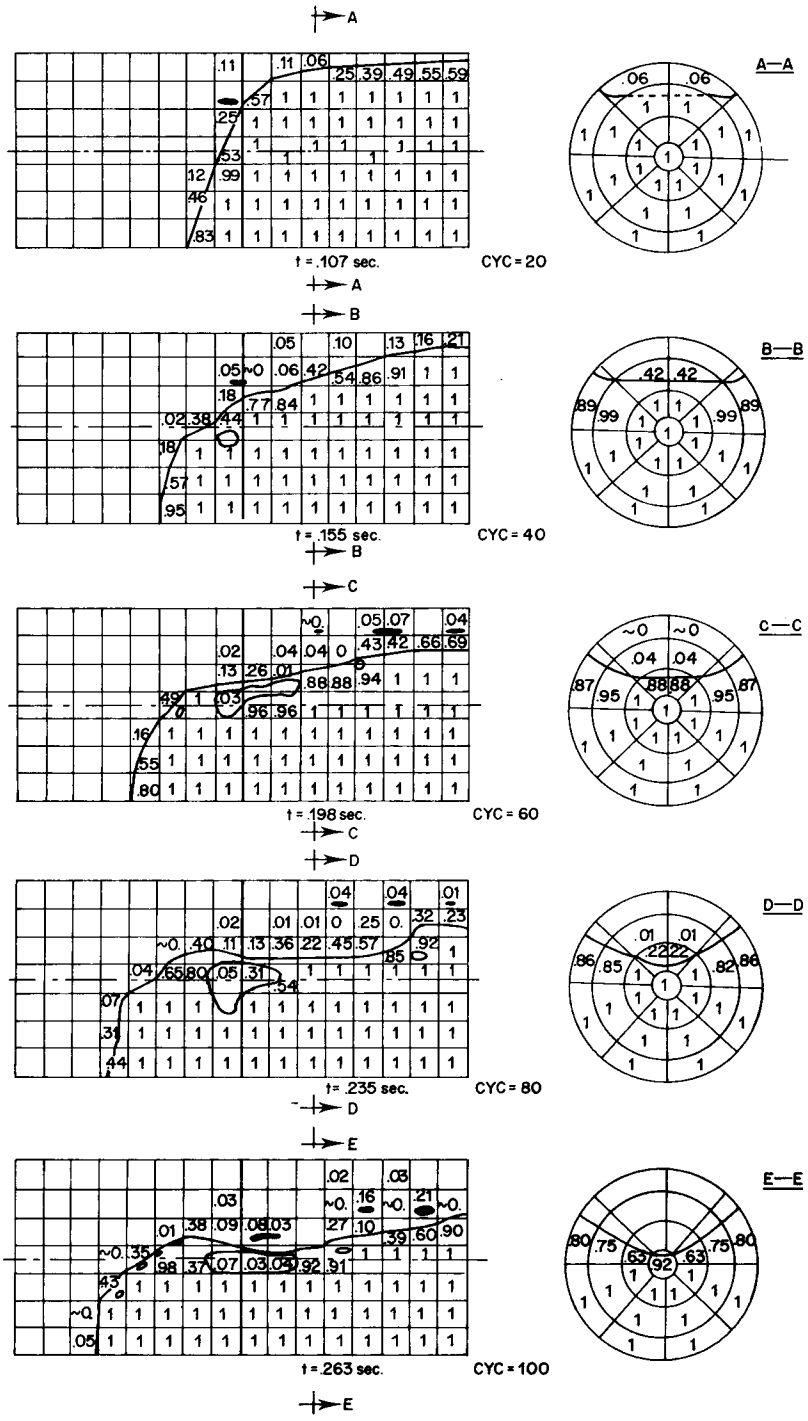


Figure 6. Example 1, case a. Fluid surface and $F$ values in longitudinal and transverse sections, at different times

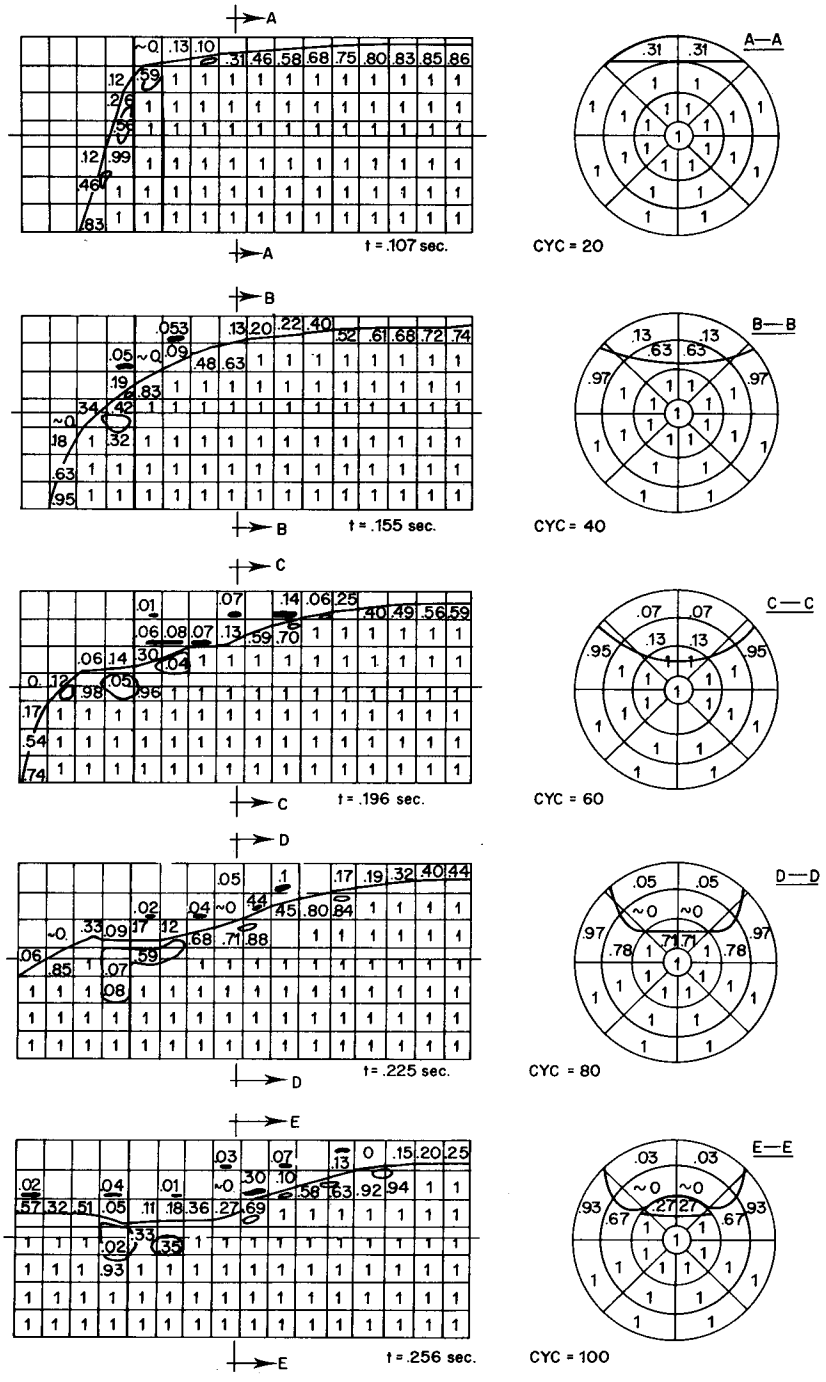


Figure 7. Example 1, case b. Fluid surface and $F$ values in longitudinal and transverse sections, at different times

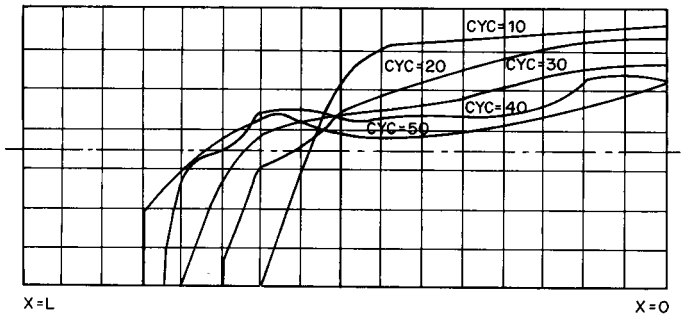


Figure 8. Example 1, case a. Fluid surface in a longitudinal section at different times

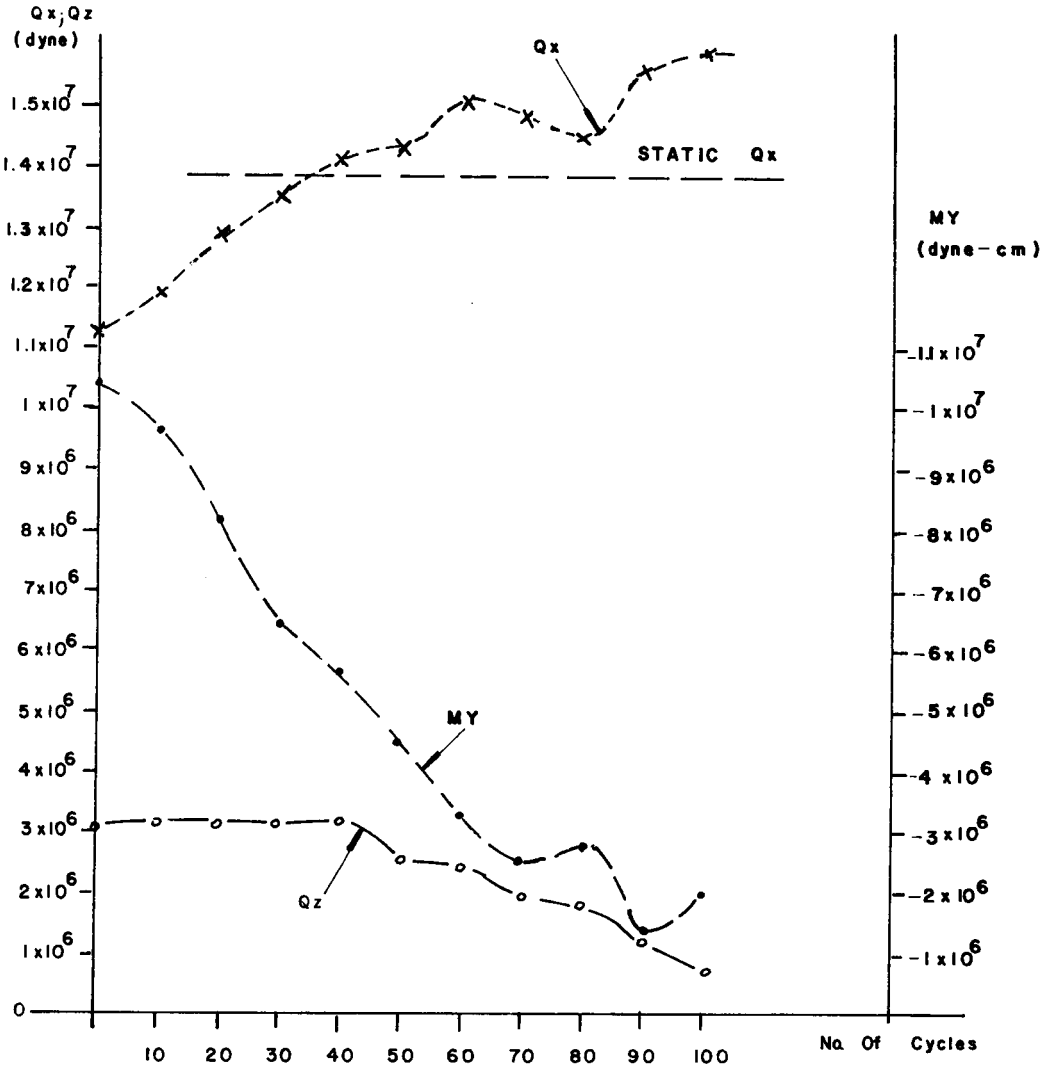


Figure 9. Example 1, case a. Resultant forces and torque exerted by the fluid on the container

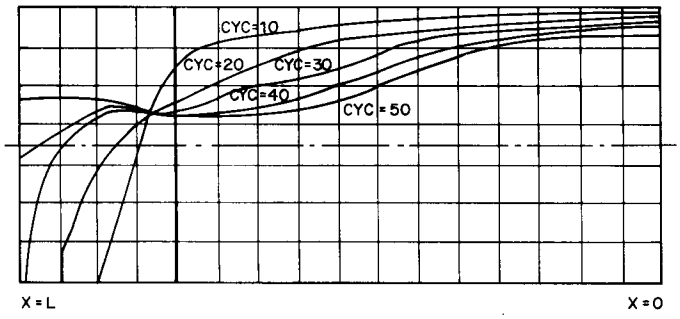


Figure 10. Example 1, case b. Fluid surface in a longitudinal section at different times

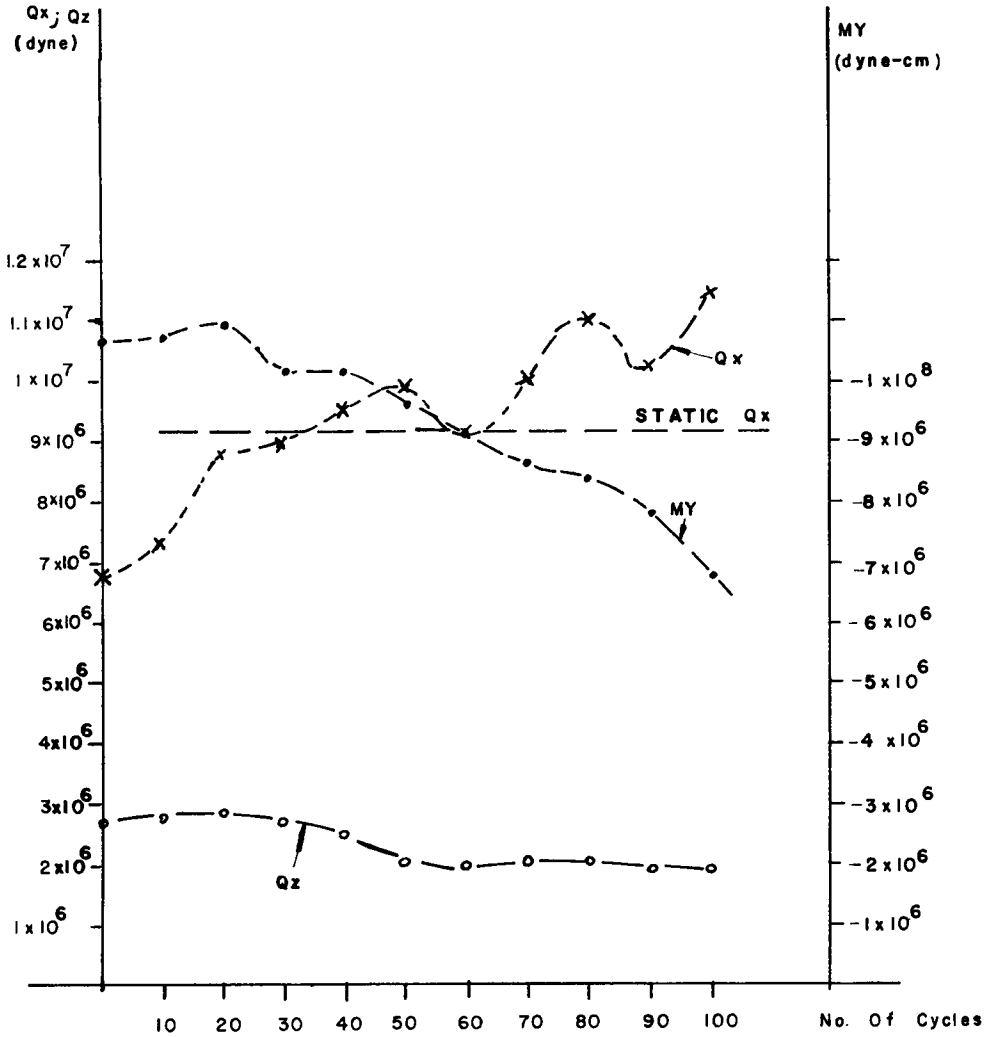


Figure 11. Example 1, case b. Resultant forces and torque exerted by the fluid on the container

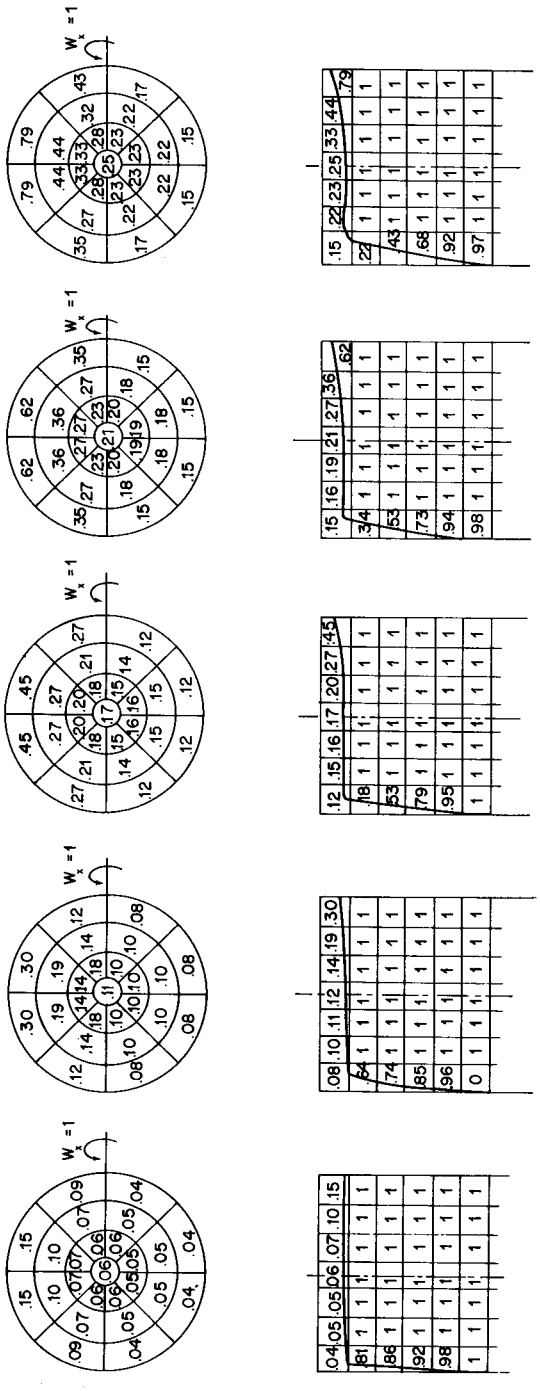


Figure 12. Example 2. Fluid surface and $F$ values in a longitudinal section and a transverse section through the upper layer of cells, at different times

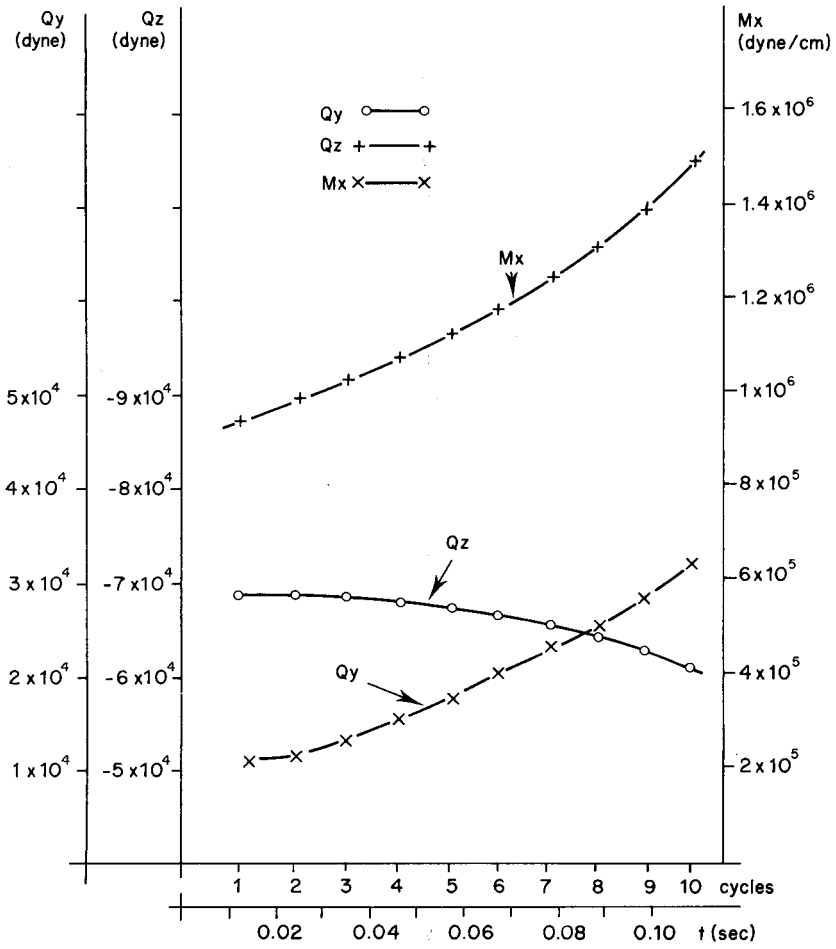


Figure 13. Example 2. Resultant forces and torque exerted by the fluid on the container

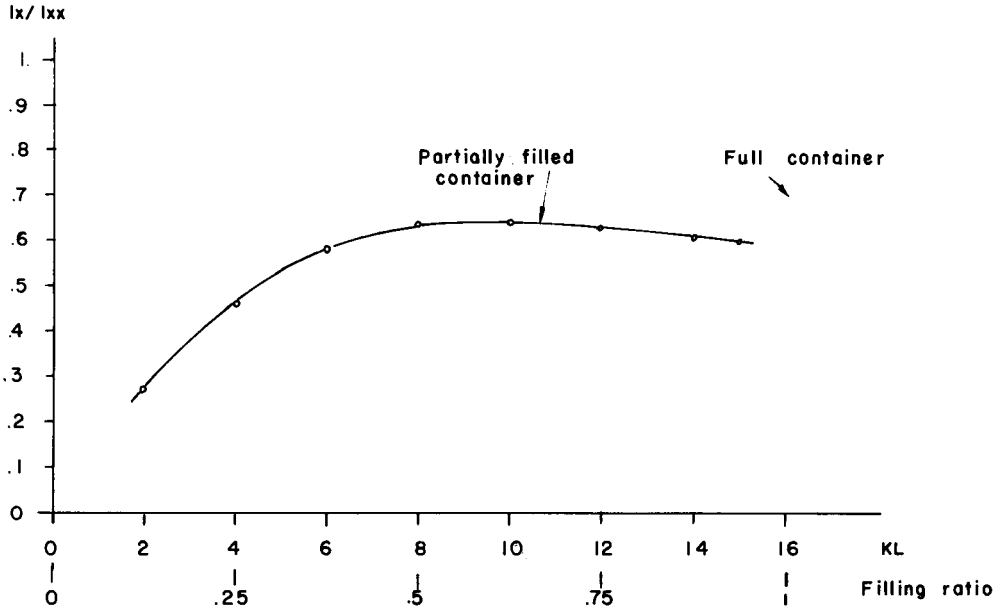


Figure 14. Example 2. Relative equivalent moment of inertia as a function of the degree of filling

## REFERENCES

1. I. S. Partom, 'Numerical calculation of equivalent moment of inertia for a fluid in a cylindrical container with partitions', *Int. j. numer. methods fluids*, **5**, 25–42 (1985).
2. Y. Kronzon, I. Partom and M. Wolfstein, in C. Taylor and B. A. Schrefler (eds), *Numerical Methods in Laminar and Turbulent Flow*, Pineridge Press, 1981, pp. 955–964.
3. J. E. Welch, F. H. Harlow, J. P. Shanon and B. J. Daly, 'The MAC method, a computing technique for solving viscous, incompressible, transient fluid flow problems involving free surface', *LASL Rept. LA-3425*, 1966.
4. C. W. Hirt and B. D. Nichols, 'Volume of fluid (VOF) methods for the dynamic of free boundaries', *J. Comp. Phys.*, **39**, 201–205 (1981).
5. C. W. Hirt *et al.*, 'SOLA-A numerical solution algorithm for transient fluid flows', *LASL Rept. LA-5852*, 1975.
6. B. D. Nichols and C. W. Hirt, 'Improved free surface boundary conditions for numerical incompressible flow calculations', *J. Comp. Physics*, **3**, 434–448 (1971).
7. B. D. Nichols and C. W. Hirt, 'Methods for calculation multi-dimensional, transient, free surface flow past bodies', *Proceedings First Int. Conf. Num. Ship Hydrodynamics*, Gaithersberg Md., October 1971.