Free-Surface Conditions in the Realistic Animation of Liquids

William F. Gates
Department of Computer Science
University of British Columbia
gates@cs.ubc.ca

December 17, 2001

Abstract

The realistic animation of liquids based on the dynamic simulation of free-surface flow requires appropriate conditions on the liquid-gas interface. These conditions can be painstaking to implement and are in general not unique. We present the conditions we use in our implementation of a fluid animation system and discuss our rationale behind them.

1 Introduction

Recent research in the computer animation of liquids has adopted the use of techniques from computational fluid dynamics for simulating free-surface flow [1, 2, 3]. In these methods, the domain is subdivided into cells which are classified as liquid or empty (or solid/boundary). The free-surface is defined to be in liquid cells neighboring empty cells. In these cells, the pressure must be set at cell centers to the desired gas pressure (here taken as zero) and the velocity must be set on all faces shared with empty cells so that there is zero divergence in the cell. This constraint is generally insufficient to derive unique conditions for the "free" velocities of cells containing the liquid surface. Thus, reasonable assumptions must be made, and appropriate conditions derived for all $2^6 - 1 = 63$ configurations of empty neighbors around a liquid cell. These conditions have not been published to the author's knowledge (and they are painful to get right), so we give explicit code for them here. First we briefly discuss our motivation for these free-surface conditions.

2 Configurations

2.1 One Empty Neighbor (Six Configurations)

If a liquid cell has one empty neighbor, we simply set the velocity on the face between the liquid and empty cell so that there is no divergence in the liquid cell. This is the

only uniquely determined case.

2.2 Two Empty Neighbors (15 Configurations)

If a liquid cell has two empty neighbors, these neighbor cells are either opposite each other or share an edge. If the two neighboring empty cells are opposite, then there is no unique approximation to the liquid surface normal. We assume that the fluid moves only due to body forces. If the two empty neighbors share an edge, we propagate the velocity from opposite faces and add half of the difference between the two opposite liquid faces.

2.3 Three Empty Neighbors (20 Configurations)

If a liquid cell has three empty neighbors, then the cell is either a "corner" (no empty neighbors are on opposite sides) or two of the empty neighbors are opposite. In the corner case the velocity values from opposite sides are propagated. In the other case, we set body forces on the two opposite empty neighbors, and set the remaining velocity value so that there is zero divergence in the cell.

2.4 Four Empty Neighbors (15 Configurations)

If a liquid cell has four empty neighbors, at least one pair of empty neighbors are opposite. The opposite pair (or pairs) are set to body forces. This results in either all velocity values being set or two empty neighbors of the liquid cell that share only an edge (in which case we apply the appropriate two empty neighbor conditions described previously).

2.5 Five Empty Neighbors (Six Configurations)

If a liquid cell has five empty neighbors, then two pairs of these neighbors must be opposite and are set according to body forces only. The remaining velocity is set to enforce zero divergence for the cell.

2.6 Six Empty Neighbors (One Configuration)

If all the neighbors of a liquid cell are empty, then it is assumed that the fluid motion is due entirely to body forces. The velocity on all faces is set to the old velocity plus the body force times the time step.

3 Conditions

In following the source code listing `H` is grid spacing, `DT` is the time step, `U` is the velocity in the x-direction, `V` is the velocity in the y-direction, `W` is the velocity in the z-direction, `Gx` is the body force in the x-direction, `Gy` is the body force in the y-direction, and `Gz` is the body force in the z-direction. While the grid is uniform in each

direction, the conditions are formulated here to facilitate the conversion to different grid spacings (δx, δy, δz) in each coordinate direction. The neighbor flag `NF` indicates which neighbors of the liquid cell (i, j, k) are empty cells. The neighbor flag uses a six digit binary number to represent the $-x$, $+x$, $-y$, $+y$, $-z$, and $+z$ empty neighbors respectively, e.g., `100000` indicates the liquid cell has exactly one neighbor in the $-x$ direction.

The conditions we use are as follows:

```
    //
    // one empty neighbor
    //


case NF_100000:
    U(i-1,j,k) = U(i,j,k) + H*(
        (V(i,j,k)-V(i,j-1,k))/H + (W(i,j,k)-W(i,j,k-1))/H);
    break;
case NF_010000:
    U(i,j,k) = U(i-1,j,k) - H*(
        (V(i,j,k)-V(i,j-1,k))/H + (W(i,j,k)-W(i,j,k-1))/H);
    break;
case NF_001000:
    V(i,j-1,k) = V(i,j,k) + H*(
        (U(i,j,k)-U(i-1,j,k))/H + (W(i,j,k)-W(i,j,k-1))/H);
    break;
case NF_000100:
    V(i,j,k) = V(i,j-1,k) - H*(
        (U(i,j,k)-U(i-1,j,k))/H + (W(i,j,k)-W(i,j,k-1))/H);
    break;
case NF_000010:
    W(i,j,k-1) = W(i,j,k) + H*(
        (U(i,j,k)-U(i-1,j,k))/H + (V(i,j,k)-V(i,j-1,k))/H);
    break;
case NF_000001:
    W(i,j,k) = W(i,j,k-1) - H*(
        (U(i,j,k)-U(i-1,j,k))/H + (V(i,j,k)-V(i,j-1,k))/H);
    break;


    //
    // two empty neighbors (opposite)
    //
case NF_110000:
    U(i-1,j,k) += Gx*DT;
    U(i,j,k) += Gx*DT;
```

```
    div =  (U(i,j,k)-U(i-1,j,k))/H +
        (V(i,j,k)-V(i,j-1,k))/H +
        (W(i,j,k)-W(i,j,k-1))/H;
    U(i-1,j,k) += 0.5*H*div;
    U(i,j,k) -= 0.5*H*div;
    break;
case NF_001100:
    V(i,j-1,k) += Gy*DT;
    V(i,j,k) += Gy*DT;
    div =  (U(i,j,k)-U(i-1,j,k))/H +
        (V(i,j,k)-V(i,j-1,k))/H +
        (W(i,j,k)-W(i,j,k-1))/H;
    V(i,j-1,k) += 0.5*H*div;
    V(i,j,k) -= 0.5*H*div;
    break;
case NF_000011:
    W(i,j,k-1) += Gz*DT;
    W(i,j,k) += Gz*DT;
    div =  (U(i,j,k)-U(i-1,j,k))/H +
        (V(i,j,k)-V(i,j-1,k))/H +
        (W(i,j,k)-W(i,j,k-1))/H;
    W(i,j,k-1) += 0.5*H*div;
    W(i,j,k) -= 0.5*H*div;
    break;


    //
    // two empty neighbors (adjacent)
    //

case NF_101000:
    U(i-1,j,k)=U(i,j,k)+0.5*H*((W(i,j,k)-W(i,j,k-1))/H);
    V(i,j-1,k)=V(i,j,k)+0.5*H*((W(i,j,k)-W(i,j,k-1))/H);
    break;
case NF_100100:
    U(i-1,j,k)=U(i,j,k)+0.5*H*((W(i,j,k)-W(i,j,k-1))/H);
    V(i,j,k)=V(i,j-1,k)-0.5*H*((W(i,j,k)-W(i,j,k-1))/H);
    break;
case NF_011000:
    U(i,j,k)=U(i-1,j,k)-0.5*H*((W(i,j,k)-W(i,j,k-1))/H);
    V(i,j-1,k)=V(i,j,k)+0.5*H*((W(i,j,k)-W(i,j,k-1))/H);
    break;
case NF_010100:
    U(i,j,k)=U(i-1,j,k)-0.5*H*((W(i,j,k)-W(i,j,k-1))/H);
    V(i,j,k)=V(i,j-1,k)-0.5*H*((W(i,j,k)-W(i,j,k-1))/H);
    break;
```

```
case NF_001010:
    V(i,j-1,k)=V(i,j,k)+0.5*H*((U(i,j,k)-U(i-1,j,k))/H);
    W(i,j,k-1)=W(i,j,k)+0.5*H*((U(i,j,k)-U(i-1,j,k))/H);
    break;
case NF_001001:
    V(i,j-1,k)=V(i,j,k)+0.5*H*((U(i,j,k)-U(i-1,j,k))/H);
    W(i,j,k)=W(i,j,k-1)-0.5*H*((U(i,j,k)-U(i-1,j,k))/H);
    break;
case NF_000110:
    V(i,j,k)=V(i,j-1,k)-0.5*H*((U(i,j,k)-U(i-1,j,k))/H);
    W(i,j,k-1)=W(i,j,k)+0.5*H*((U(i,j,k)-U(i-1,j,k))/H);
    break;
case NF_000101:
    V(i,j,k)=V(i,j-1,k)-0.5*H*((U(i,j,k)-U(i-1,j,k))/H);
    W(i,j,k)=W(i,j,k-1)-0.5*H*((U(i,j,k)-U(i-1,j,k))/H);
    break;

case NF_100010:
    U(i-1,j,k)=U(i,j,k)+0.5*H*((V(i,j,k)-V(i,j-1,k))/H);
    W(i,j,k-1)=W(i,j,k)+0.5*H*((V(i,j,k)-V(i,j-1,k))/H);
    break;
case NF_100001:
    U(i-1,j,k)=U(i,j,k)+0.5*H*((V(i,j,k)-V(i,j-1,k))/H);
    W(i,j,k)=W(i,j,k-1)-0.5*H*((V(i,j,k)-V(i,j-1,k))/H);
    break;
case NF_010010:
    U(i,j,k)=U(i-1,j,k)-0.5*H*((V(i,j,k)-V(i,j-1,k))/H);
    W(i,j,k-1)=W(i,j,k)+0.5*H*((V(i,j,k)-V(i,j-1,k))/H);
    break;
case NF_010001:
    U(i,j,k)=U(i-1,j,k)-0.5*H*((V(i,j,k)-V(i,j-1,k))/H);
    W(i,j,k)=W(i,j,k-1)-0.5*H*((V(i,j,k)-V(i,j-1,k))/H);
    break;



    //
    // three empty neighbors (adjacent/liquid corner)
    //

case NF_010101:
    U(i,j,k) = U(i-1,j,k);
    V(i,j,k) = V(i,j-1,k);
    W(i,j,k) = W(i,j,k-1);
    break;
```

```
case NF_100101:
    U(i-1,j,k) = U(i,j,k);
    V(i,j,k) = V(i,j-1,k);
    W(i,j,k) = W(i,j,k-1);
    break;
case NF_011001:
    U(i,j,k) = U(i-1,j,k);
    V(i,j-1,k) = V(i,j,k);
    W(i,j,k) = W(i,j,k-1);
    break;
case NF_101001:
    U(i-1,j,k) = U(i,j,k);
    V(i,j-1,k) = V(i,j,k);
    W(i,j,k) = W(i,j,k-1);
    break;
case NF_010110:
    U(i,j,k) = U(i-1,j,k);
    V(i,j,k) = V(i,j-1,k);
    W(i,j,k-1) = W(i,j,k);
    break;
case NF_100110:
    U(i-1,j,k) = U(i,j,k);
    V(i,j,k) = V(i,j-1,k);
    W(i,j,k-1) = W(i,j,k);
    break;
case NF_011010:
    U(i,j,k) = U(i-1,j,k);
    V(i,j-1,k) = V(i,j,k);
    W(i,j,k-1) = W(i,j,k);
    break;
case NF_101010:
    U(i-1,j,k) = U(i,j,k);
    V(i,j-1,k) = V(i,j,k);
    W(i,j,k-1) = W(i,j,k);
    break;


    //
    // three empty neighbors (two opposite)
    //

case NF_111000:
    U(i-1,j,k) += Gx*DT;
    U(i,j,k) += Gx*DT;
    V(i,j-1,k) = V(i,j,k) + H*(
        (U(i,j,k)-U(i-1,j,k))/H + (W(i,j,k)-W(i,j,k-1))/H);
```

```
    break;
case NF_110100:
    U(i-1,j,k) += Gx*DT;
    U(i,j,k) += Gx*DT;
    V(i,j,k) = V(i,j-1,k) - H*(
        (U(i,j,k)-U(i-1,j,k))/H + (W(i,j,k)-W(i,j,k-1))/H);
    break;
case NF_110010:
    U(i-1,j,k) += Gx*DT;
    U(i,j,k) += Gx*DT;
    W(i,j,k-1) = W(i,j,k) + H*(
        (U(i,j,k)-U(i-1,j,k))/H + (V(i,j,k)-V(i,j-1,k))/H);
    break;
case NF_110001:
    U(i-1,j,k) += Gx*DT;
    U(i,j,k) += Gx*DT;
    W(i,j,k) = W(i,j,k-1) - H*(
        (U(i,j,k)-U(i-1,j,k))/H + (V(i,j,k)-V(i,j-1,k))/H);
    break;

case NF_101100:
    V(i,j-1,k) += Gy*DT;
    V(i,j,k) += Gy*DT;
    U(i-1,j,k) = U(i,j,k) + H*(
        (V(i,j,k)-V(i,j-1,k))/H + (W(i,j,k)-W(i,j,k-1))/H);
    break;
case NF_011100:
    V(i,j-1,k) += Gy*DT;
    V(i,j,k) += Gy*DT;
    U(i,j,k) = U(i-1,j,k) - H*(
        (V(i,j,k)-V(i,j-1,k))/H + (W(i,j,k)-W(i,j,k-1))/H);
    break;
case NF_001110:
    V(i,j-1,k) += Gy*DT;
    V(i,j,k) += Gy*DT;
    W(i,j,k-1) = W(i,j,k) + H*(
        (U(i,j,k)-U(i-1,j,k))/H + (V(i,j,k)-V(i,j-1,k))/H);
    break;
case NF_001101:
    V(i,j-1,k) += Gy*DT;
    V(i,j,k) += Gy*DT;
    W(i,j,k) = W(i,j,k-1) - H*(
        (U(i,j,k)-U(i-1,j,k))/H + (V(i,j,k)-V(i,j-1,k))/H);
    break;
```

```
case NF_100011:
    W(i,j,k-1) += Gz*DT;
    W(i,j,k) += Gz*DT;
    U(i-1,j,k) = U(i,j,k) + H*(
        (V(i,j,k)-V(i,j-1,k))/H + (W(i,j,k)-W(i,j,k-1))/H);
    break;
case NF_010011:
    W(i,j,k-1) += Gz*DT;
    W(i,j,k) += Gz*DT;
    U(i,j,k) = U(i-1,j,k) - H*(
        (V(i,j,k)-V(i,j-1,k))/H + (W(i,j,k)-W(i,j,k-1))/H);
    break;
case NF_001011:
    W(i,j,k-1) += Gz*DT;
    W(i,j,k) += Gz*DT;
    V(i,j-1,k) = V(i,j,k) + H*(
        (U(i,j,k)-U(i-1,j,k))/H + (W(i,j,k)-W(i,j,k-1))/H);
    break;
case NF_000111:
    W(i,j,k-1) += Gz*DT;
    W(i,j,k) += Gz*DT;
    V(i,j,k) = V(i,j-1,k) - H*(
        (U(i,j,k)-U(i-1,j,k))/H + (W(i,j,k)-W(i,j,k-1))/H);
    break;



    //
    // four empty neighbors (two opposite pairs)
    //

case NF_111100:
    U(i-1,j,k) += Gx*DT;
    U(i,j,k) += Gx*DT;
    V(i,j-1,k) += Gy*DT;
    V(i,j,k) += Gy*DT;
    div =  (U(i,j,k)-U(i-1,j,k))/H +
        (V(i,j,k)-V(i,j-1,k))/H +
        (W(i,j,k)-W(i,j,k-1))/H;
    U(i-1,j,k) += 0.25*H*div;
    U(i,j,k) -= 0.25*H*div;
    V(i,j-1,k) += 0.25*H*div;
    V(i,j,k) -= 0.25*H*div;
    break;
case NF_110011:
    U(i-1,j,k) += Gx*DT;
```

```
    U(i,j,k) += Gx*DT;
    W(i,j,k-1) += Gz*DT;
    W(i,j,k) += Gz*DT;
    div =  (U(i,j,k)-U(i-1,j,k))/H +
        (V(i,j,k)-V(i,j-1,k))/H +
        (W(i,j,k)-W(i,j,k-1))/H;
    U(i-1,j,k) += 0.25*H*div;
    U(i,j,k) -= 0.25*H*div;
    W(i,j,k-1) += 0.25*H*div;
    W(i,j,k) -= 0.25*H*div;
    break;
case NF_001111:
    V(i,j-1,k) += Gy*DT;
    V(i,j,k) += Gy*DT;
    W(i,j,k-1) += Gz*DT;
    W(i,j,k) += Gz*DT;
    div =  (U(i,j,k)-U(i-1,j,k))/H +
        (V(i,j,k)-V(i,j-1,k))/H +
        (W(i,j,k)-W(i,j,k-1))/H;
    V(i,j-1,k) += 0.25*H*div;
    V(i,j,k) -= 0.25*H*div;
    W(i,j,k-1) += 0.25*H*div;
    W(i,j,k) -= 0.25*H*div;
    break;


    //
    // four empty neighbors (one opposite pair)
    //

case NF_110101:
    U(i-1,j,k) += Gx*DT;
    U(i,j,k) += Gx*DT;
    V(i,j,k)=V(i,j-1,k)-0.5*H*((U(i,j,k)-U(i-1,j,k))/H);
    W(i,j,k)=W(i,j,k-1)-0.5*H*((U(i,j,k)-U(i-1,j,k))/H);
    break;
case NF_110110:
    U(i-1,j,k) += Gx*DT;
    U(i,j,k) += Gx*DT;
    V(i,j,k)=V(i,j-1,k)-0.5*H*((U(i,j,k)-U(i-1,j,k))/H);
    W(i,j,k-1)=W(i,j,k)+0.5*H*((U(i,j,k)-U(i-1,j,k))/H);
    break;
case NF_111001:
    U(i-1,j,k) += Gx*DT;
    U(i,j,k) += Gx*DT;
    V(i,j-1,k)=V(i,j,k)+0.5*H*((U(i,j,k)-U(i-1,j,k))/H);
```

```
    W(i,j,k)=W(i,j,k-1)-0.5*H*((U(i,j,k)-U(i-1,j,k))/H);
    break;
case NF_111010:
    U(i-1,j,k) += Gx*DT;
    U(i,j,k) += Gx*DT;
    V(i,j-1,k)=V(i,j,k)+0.5*H*((U(i,j,k)-U(i-1,j,k))/H);
    W(i,j,k-1)=W(i,j,k)+0.5*H*((U(i,j,k)-U(i-1,j,k))/H);
    break;
case NF_011101:
    V(i,j-1,k) += Gy*DT;
    V(i,j,k) += Gy*DT;
    U(i,j,k)=U(i-1,j,k)-0.5*H*((V(i,j,k)-V(i,j-1,k))/H);
    W(i,j,k)=W(i,j,k-1)-0.5*H*((V(i,j,k)-V(i,j-1,k))/H);
    break;
case NF_011110:
    V(i,j-1,k) += Gy*DT;
    V(i,j,k) += Gy*DT;
    U(i,j,k)=U(i-1,j,k)-0.5*H*((V(i,j,k)-V(i,j-1,k))/H);
    W(i,j,k-1)=W(i,j,k)+0.5*H*((V(i,j,k)-V(i,j-1,k))/H);
    break;
case NF_101101:
    V(i,j-1,k) += Gy*DT;
    V(i,j,k) += Gy*DT;
    U(i-1,j,k)=U(i,j,k)+0.5*H*((V(i,j,k)-V(i,j-1,k))/H);
    W(i,j,k)=W(i,j,k-1)-0.5*H*((V(i,j,k)-V(i,j-1,k))/H);
    break;
case NF_101110:
    V(i,j-1,k) += Gy*DT;
    V(i,j,k) += Gy*DT;
    U(i-1,j,k)=U(i,j,k)+0.5*H*((V(i,j,k)-V(i,j-1,k))/H);
    W(i,j,k-1)=W(i,j,k)+0.5*H*((V(i,j,k)-V(i,j-1,k))/H);
    break;
case NF_010111:
    W(i,j,k-1) += Gz*DT;
    W(i,j,k) += Gz*DT;
    U(i,j,k)=U(i-1,j,k)-0.5*H*((W(i,j,k)-W(i,j,k-1))/H);
    V(i,j,k)=V(i,j-1,k)-0.5*H*((W(i,j,k)-W(i,j,k-1))/H);
    break;
case NF_011011:
    W(i,j,k-1) += Gz*DT;
    W(i,j,k) += Gz*DT;
    U(i,j,k)=U(i-1,j,k)-0.5*H*((W(i,j,k)-W(i,j,k-1))/H);
    V(i,j-1,k)=V(i,j,k)+0.5*H*((W(i,j,k)-W(i,j,k-1))/H);
    break;
case NF_100111:
    W(i,j,k-1) += Gz*DT;
```

```
    W(i,j,k) += Gz*DT;
    U(i-1,j,k)=U(i,j,k)+0.5*H*((W(i,j,k)-W(i,j,k-1))/H);
    V(i,j,k)=V(i,j-1,k)-0.5*H*((W(i,j,k)-W(i,j,k-1))/H);
    break;
case NF_101011:
    W(i,j,k-1) += Gz*DT;
    W(i,j,k) += Gz*DT;
    U(i-1,j,k)=U(i,j,k)+0.5*H*((W(i,j,k)-W(i,j,k-1))/H);
    V(i,j-1,k)=V(i,j,k)+0.5*H*((W(i,j,k)-W(i,j,k-1))/H);
    break;



    //
    // five empty neighbors
    //

case NF_011111:
    V(i,j-1,k) += Gy*DT;
    V(i,j,k) += Gy*DT;
    W(i,j,k-1) += Gz*DT;
    W(i,j,k) += Gz*DT;
    U(i,j,k) = U(i-1,j,k) - H*(
        (V(i,j,k)-V(i,j-1,k))/H + (W(i,j,k)-W(i,j,k-1))/H);
    break;
case NF_101111:
    V(i,j-1,k) += Gy*DT;
    V(i,j,k) += Gy*DT;
    W(i,j,k-1) += Gz*DT;
    W(i,j,k) += Gz*DT;
    U(i-1,j,k) = U(i,j,k) + H*(
        (V(i,j,k)-V(i,j-1,k))/H + (W(i,j,k)-W(i,j,k-1))/H);
    break;
case NF_110111:
    U(i-1,j,k) += Gx*DT;
    U(i,j,k) += Gx*DT;
    W(i,j,k-1) += Gz*DT;
    W(i,j,k) += Gz*DT;
    V(i,j,k) = V(i,j-1,k) - H*(
        (U(i,j,k)-U(i-1,j,k))/H + (W(i,j,k)-W(i,j,k-1))/H);
    break;
case NF_111011:
    U(i-1,j,k) += Gx*DT;
    U(i,j,k) += Gx*DT;
    W(i,j,k-1) += Gz*DT;
    W(i,j,k) += Gz*DT;
```

```
        V(i,j-1,k) = V(i,j,k) + H*(
            (U(i,j,k)-U(i-1,j,k))/H + (W(i,j,k)-W(i,j,k-1))/H);
        break;
    case NF_111101:
        U(i-1,j,k) += Gx*DT;
        U(i,j,k) += Gx*DT;
        V(i,j-1,k) += Gy*DT;
        V(i,j,k) += Gy*DT;
        W(i,j,k) = W(i,j,k-1) - H*(
            (U(i,j,k)-U(i-1,j,k))/H + (V(i,j,k)-V(i,j-1,k))/H);
        break;
    case NF_111110:
        U(i-1,j,k) += Gx*DT;
        U(i,j,k) += Gx*DT;
        V(i,j-1,k) += Gy*DT;
        V(i,j,k) += Gy*DT;
        W(i,j,k-1) = W(i,j,k) + H*(
            (U(i,j,k)-U(i-1,j,k))/H + (V(i,j,k)-V(i,j-1,k))/H);
        break;



        //
        // six empty neighbors
        //

    case NF_111111:
        U(i-1,j,k) += Gx*DT;
        U(i,j,k) += Gx*DT;
        V(i,j-1,k) += Gy*DT;
        V(i,j,k) += Gy*DT;
        W(i,j,k-1) += Gz*DT;
        W(i,j,k) += Gz*DT;
        break;
```

References

[1] N. Foster and D. Metaxas. Controlling fluid animation. In *Computer Graphics International 1997*, Hasselt/Diepenbeek, Belgium, June 1997. IEEE Computer Society.

[2] Nick Foster and Ronald Fedkiw. Practical animation of liquids. In *Proceedings of SIGGRAPH 2001*, Computer Graphics Proceedings, Annual Conference Series, pages 23–30. ACM Press / ACM SIGGRAPH, August 2001. ISBN 1-58113-292-1.

[3] Nick Foster and Dimitri Metaxas. Realistic animation of liquids. *Graphical Models and Image Processing*, 58(5):471–483, September 1996.