

Direct Numerical Simulations of Multiphase Flows-6

Surface Tension, Unequal Viscosities, and Higher Order Time Integration

Gretar Tryggvason

1. We are almost done with the complete code. However, a few important steps remain.

DNS of Multiphase Flows — Simple Front Tracking

In this lecture we will complete our code by:

- Adding surface tension
- Allowing the viscosities in the different fluids to be different
- Make the time integration higher order

The resulting code is a fully functional one, and allows us to simulate simple multi fluid problems

2. In this lecture we will do three things. We will add a way to include surface tension, we will modify the code so that the viscosities in the different fluids can be different, and we will increase the order of the time integration.

DNS of Multiphase Flows

Surface Tension

3. First the surface tension.

DNS of Multiphase Flows

The surface tension at a point is given by

$$\mathbf{f}^\sigma = \sigma \kappa \mathbf{n}$$

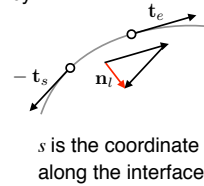
We need, however, the total force on a small segment of the front. For two-dimensional flows we can use the definition of the curvature

$$\kappa \mathbf{n} = \frac{\partial \mathbf{t}}{\partial s}$$

The force on a small element l is

$$\delta \mathbf{f}_l^\sigma = \int_{\Delta s_l} \mathbf{f}^\sigma ds = \int_{\Delta s_l} \sigma \kappa \mathbf{n} ds = \sigma \int_{\Delta s_l} \frac{\partial \mathbf{t}}{\partial s} ds = \sigma (\mathbf{t}_e - \mathbf{t}_s)$$

Thus, we do not need to find the curvature, just the tangent vectors, which is generally much simpler



4-1. The key thing about surface tension is that it is a tension. The force is parallel to the interface and if the interface is flat, the pull on one end of a small interface segment is balanced by the pull on the other side. If, however, the interface is curved, there is a net force in the normal direction that is proportional to the curvature. For our purpose it is important that we need the force on the part of an interface that lies within a control volume---not the force per unit area. While we may be tempted to write down the force per unit area and then multiply it by the area of the interface segment, it is better to go back to the definition of the surface tension. We can get there from the standard definition of the interface force by using that the curvature times the normal is, by definition, equal to the derivative of the tangent vector with respect to the interface length s . The integral over the curvature times the normal over the interface segment Δs is, if σ is constant, exactly equal to the difference in the pull on either edge times the surface tension coefficient, or the surface tension coefficient times the difference between the tangents at the edges.

DNS of Multiphase Flows

The surface tension at a point is given by

$$\mathbf{f}^\sigma = \sigma \kappa \mathbf{n}$$

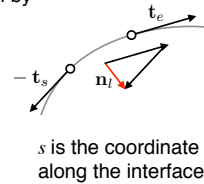
We need, however, the total force on a small segment of the front. For two-dimensional flows we can use the definition of the curvature

$$\kappa \mathbf{n} = \frac{\partial \mathbf{t}}{\partial s}$$

The force on a small element l is

$$\delta \mathbf{f}_l^\sigma = \int_{\Delta s_l} \mathbf{f}^\sigma ds = \int_{\Delta s_l} \sigma \kappa \mathbf{n} ds = \sigma \int_{\Delta s_l} \frac{\partial \mathbf{t}}{\partial s} ds = \sigma (\mathbf{t}_e - \mathbf{t}_s)$$

Thus, we do not need to find the curvature, just the tangent vectors, which is generally much simpler



4-2. Thus, we only need to find the tangents to the interface and not the curvature, which simplifies the computations considerably. Furthermore, we end up with a more robust algorithm that is conservative in the sense that the force on the end of one segment is exactly equal and opposite to the force on the segment attached to it. This is important since for closed surfaces the net surface force should be zero and when we sum over all the interface elements, the forces on adjacent elements cancel.

DNS of Multiphase Flows

Here we take the interface segment around each interface point to consist of half the distance to the points on either side

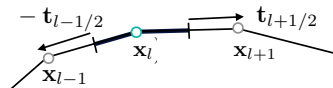
The tangents at the end of the segment are found using a centered approximation

$$\mathbf{t}_{l+1/2} = (\mathbf{x}_{l+1} - \mathbf{x}_l) / \Delta s$$

where
$$\Delta s = \sqrt{(x_{l+1} - x_l)^2 + (y_{l+1} - y_l)^2}$$

The surface force acting on the interface segment around point l is:

$$\delta \mathbf{f}_l^\sigma = \sigma (\mathbf{t}_{l+1/2} - \mathbf{t}_{l-1/2})$$



5. The numerical implementation is straightforward. We compute the surface force on an interface segment that consists of half of the elements attached to an interface point. The tangent force that pulls on the segment to the right of point l , halfway between point l and $l+1$, is given by the difference in the coordinates of these points divided by Δs , the distance between them. Δs is computed as the square root of the difference in the x coordinates squared plus the difference in the y coordinates squared. Thus, the force on the interface segment containing point l , and consisting of half the element before and after the point is given by $\Delta \mathbf{f}$ equal to σ times the tangent vector at $l+1/2$ minus the tangent vector at $l-1/2$.

DNS of Multiphase Flows

The surface tension is distributed to the fixed grid. On the front the force is per unit area and on the grid the force is per unit volume. The total force is conserved, so that:

$$\int_{\Delta s_i} \mathbf{f}_s^\sigma ds = \int_{\delta V} \mathbf{f}_v^\sigma dv$$

The total force on an interface segment is

$$\mathbf{F}_i^\sigma = \int_{\Delta s_i} \mathbf{f}_s^\sigma ds \approx \sum_l \delta \mathbf{f}_i^\sigma$$

where the integral is over the part of the interface contributing to a given grid point and the sum is over the interface segment that do.

The total force at a given grid point is

$$\mathbf{F}_{i,j}^\sigma = \int_{\delta V} \mathbf{f}_v^\sigma dv \approx \mathbf{f}_{i,j}^\sigma \Delta x \Delta y$$

6. The main consideration when transferring the force from the interface to the fixed fluid grid is that the total force must be conserved. Thus, the surface integral over the force per unit area (or length in two-dimensions) must be equal to the volume integral over the force per unit volume (or area in two dimension) on the fixed grid. The total force on the interface segment that will be transferred to a given grid point is the integral over that segment. Each segment generally consists of several interface elements and we must sum over those elements. For each element we found the force, denoted by $\delta \mathbf{f}_l$, on the last slide by subtracting the tangents. On the fixed grid we need the force per unit area in two-dimensions, and the total force is the integral over the control volume surrounding each grid point, or $\mathbf{f}_{i,j}$ times Δx and Δy .

DNS of Multiphase Flows

Using that $\mathbf{F}_{i,j}^\sigma = \mathbf{F}_i^\sigma$ the force at each grid point, transferred from the front, is:

$$\mathbf{f}_{i,j}^\sigma = \sum_l \frac{\delta \mathbf{f}_l^\sigma w_{i,j}^l}{\Delta x \Delta y} \quad \begin{array}{l} \text{Sum over elements} \\ \text{"close" to grid point } i,j \end{array}$$

The weights $w_{i,j}^l$ are generally taken to be the same as those used to interpolate the velocities to the front

On a staggered grid, the x -component of the surface force is distributed to the u -nodes and the y -component to the v -nodes.

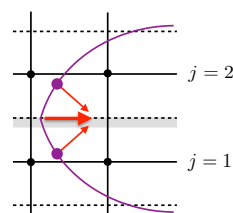
The surface forces on the fixed grid are then added to the discrete Navier-Stokes equations

7. The total force on a surface segment that is transferred to a given grid point must be equal to the total force on the grid. The force transferred from each element is already the integral over the element, so we just need to sum those over all elements that are close to each grid points, and to recover the force per unit volume on the fixed grid we divide by Δx times Δy . In general each surface element can contribute to more than one point on the fixed grid and we divide the surface force between the different grid points based on weights w , where the superscript l denotes the element and the subscript i,j denotes the fixed grid points. Obviously, the weights for each element must add up to one, so that the total force is accounted for. When we use a staggered grid, where the different velocity components each have their own control volume, the different component of the surface force are distributed to different locations on the fixed grid. Once we have the surface force per unit volume on the grid, those can be added to the discrete Navier-Stokes equations.

DNS of Multiphase Flows

At solid walls we prescribe symmetry boundary conditions so the net force at the wall would be tangent to the wall

For weights based on bilinear interpolation, the force is distributed to the four nearest grid points so no adjustment is needed for the normal velocity. For the tangential component we need to add the contribution from the "ghost" part



For the bottom boundary

$$(f_x)_{i,j=2} = (f_x)_{i,j=2} + (f_x)_{i,j=1}$$

Similar adjustments are done for the other boundaries

8. For the fluid flow we can assume that a flat boundary can be represented as a symmetry boundary. Although this is mostly used for inviscid flows, we can also use it for no-slip boundaries if we add the constraint of a zero tangent velocity. We assume that the same is true if we add surface tension so the surface tension for the boundary nodes can be set by assuming a symmetric, or a ghost, interface on the other side of the boundary. This results in a zero normal force by symmetry and for bilinear interpolation, or area weighting, no adjustment is needed for the normal velocity since we are not solving for the velocity at the boundary. The tangent force, at a point half a grid spacing away from the boundary, has to be modified by adding the contribution from the ghost interface on the other side.

DNS of Multiphase Flows

Code to find surface tension and distribute it to the fixed grid

```
%----- Find surface tension -----
fx=zeros(nx+2,ny+2);fy=zeros(nx+2,ny+2); % Set fx & fy to zero
for i=1:Nf+1
    ds=sqrt((x(i)-x(i-1))^2+(y(i)-y(i-1))^2);
    tx(i)=(x(i)-x(i-1))/ds; % Tangent vectors
    ty(i)=(y(i)-y(i-1))/ds;
end
tx(Nf+2)=tx(2);ty(Nf+2)=ty(2);

for i=2:Nf+1 % Distribute to the fixed grid
    ntx=sigma*(tx(i)-tx(i-1));nty=sigma*(ty(i)-ty(i-1));

    ip=floor(x(i)/dx)+1; jp=floor(y(i)/dy)+1;
    ax=(x(i)-x(ip)+1; ay=(y(i)-y(jp)+1;
    fx(ip,jp) =fx(ip,jp)+(1.0-ax)*(1.0-ay)*ntx/dx/dy;
    tx(ip+1,jp) =tx(ip+1,jp)+ax*(1.0-ay)*ntx/dx/dy;
    fy(ip,jp+1) =fy(ip,jp+1)+(1.0-ax)*nty/dx/dy;
    ty(ip+1,jp+1)=ty(ip+1,jp+1)+ax*nty/dx/dy;

    ip=floor(x(i)/dx)+1; jp=floor(y(i)/dy)+1;
    ax=(x(i)-x(ip)+1; ay=(y(i)-y(jp)+1;
    fy(ip,jp) =fy(ip,jp)+(1.0-ax)*(1.0-ay)*nty/dx/dy;
    ty(ip+1,jp) =ty(ip+1,jp)+ax*(1.0-ay)*nty/dx/dy;
    fy(ip,jp+1) =fy(ip,jp+1)+(1.0-ax)*nty/dx/dy;
    ty(ip+1,jp+1)=ty(ip+1,jp+1)+ax*nty/dx/dy;
end

fx(1:nx+2,2)=fx(1:nx+2,2)+fx(1:nx+2,1); % Correct boundary
fx(1:nx+2,nx+1)=fx(1:nx+2,nx+1)+fx(1:nx+2,nx+2); % values for the
fy(2,1:ny+2)=fy(2,1:ny+2)+fy(1,1:ny+2); % surface force
fy(nx+1,1:ny+2)=fy(nx+1,1:ny+2)+fy(nx+2,1:ny+2); % on the grid
```

9. The addition to our code is relatively simple and consists mostly of lines needed to transfer the force from the front to the grid. Since the x component goes to the points where the u-velocity is stored and y component goes to the points where the v-velocity is stored we need separate lines of code for the transfer of each components. Here, we find the tangent vector in the first for loop and distribute the force to the grid in the second loop. The tangent force for boundary nodes is then corrected at the end.

DNS of Multiphase Flows

Unequal Viscosities

10. In general the different fluids will have different viscosities. We started by assuming that they were the same, to make the development simpler, but here we extend the code to fluids with different viscosities. We do, however, assume that the viscosity in each fluid remains constant, as the densities do.

DNS of Multiphase Flows

For unequal viscosities we need to use the full deformation tensor. The viscous term is discretized by integrating over the boundaries of the velocity control volume

$$\mathbf{D}_h = \frac{1}{V} \oint_S \mu (\nabla \mathbf{u} + (\nabla \mathbf{u})^T) \cdot \mathbf{n} ds$$

$$= \frac{1}{V} \oint_S \mu \left[\begin{array}{cc} 2 \frac{\partial u}{\partial x} & \frac{\partial u}{\partial y} + \frac{\partial v}{\partial x} \\ \frac{\partial u}{\partial y} + \frac{\partial v}{\partial x} & 2 \frac{\partial v}{\partial y} \end{array} \right] \cdot \mathbf{n} ds$$

Using that on vertical boundaries the normal is

Horizontal boundaries $\mathbf{n} = (0, 1)$ and $\mathbf{n} = (0, -1)$

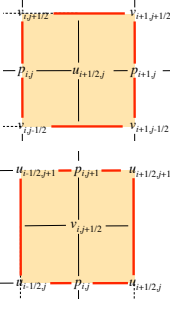
Vertical boundaries $\mathbf{n} = (1, 0)$ and $\mathbf{n} = (-1, 0)$

11. The main change is that we need to include the full deformation tensor, so the viscous stresses are now given by the surface integral of the viscosity times the full rate of deformation tensor dotted into the normal. The deformation tensor is the symmetric part of the velocity gradient tensor, obtained by adding the tensor to its transpose, and for two dimensional flow we have three independent components since the off-diagonal terms are the same. Since we are working with values per unit volume, we need to divide by the volume of the control volume. For rectangular control volumes the normal on each side has only one non-zero component.

DNS of Multiphase Flows

Integrating around the control volume gives

$$(D_x)_{i+1/2,j}^n = \frac{1}{\Delta x \Delta y} \left\{ 2 \left(\mu \frac{\partial u}{\partial x} \right)_{i+1,j} - 2 \left(\mu \frac{\partial u}{\partial x} \right)_{i,j} \right\} \Delta y + \left(\mu \left(\frac{\partial u}{\partial y} + \frac{\partial v}{\partial x} \right)_{i+1/2,j+1/2} - \mu \left(\frac{\partial u}{\partial y} + \frac{\partial v}{\partial x} \right)_{i+1/2,j-1/2} \right) \Delta x$$

$$(D_y)_{i,j+1/2}^n = \frac{1}{\Delta x \Delta y} \left\{ \left(\mu \left(\frac{\partial v}{\partial x} + \frac{\partial u}{\partial y} \right)_{i+1/2,j+1/2} - \mu \left(\frac{\partial v}{\partial x} + \frac{\partial u}{\partial y} \right)_{i-1/2,j+1/2} \right) \Delta y + \left(2 \left(\mu \frac{\partial v}{\partial y} \right)_{i,j+1} - 2 \left(\mu \frac{\partial v}{\partial y} \right)_{i,j} \right) \Delta x \right\}$$


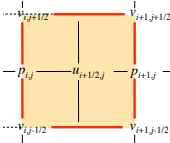
12. The integral is approximated by the midpoint rule for each side and since the normal on each side has only one non-zero component, only one part of the deformation tensor appears on each side. For the x-component of the viscous stresses, the first part results from integrating over the vertical boundaries, giving the difference in the viscosity times the rate of change of the u-velocity in the x-direction on the left and the right boundary, times delta y, and the second part gives the difference in the viscosity times the sum of the y derivative of u and the x derivative of v on the top and the bottom, times delta x. Similarly, the y-component of the viscous stresses consists of the difference in the viscosity times the sum of the x derivative of v and the y derivative of u on the left and right, times delta y, plus the difference in the viscosity times the rate of change of the v-velocity in the y-direction on the top and the bottom boundary, times delta x.

DNS of Multiphase Flows

Diffusion term: discrete term for the u-velocity

$$(D_x)_{i+1/2,j}^n = \frac{1}{\Delta x} \left\{ 2 \mu_{i+1,j}^n \left(\frac{u_{i+3/2,j}^n - u_{i+1/2,j}^n}{\Delta x} \right) - 2 \mu_{i,j}^n \left(\frac{u_{i+1/2,j}^n - u_{i-1/2,j}^n}{\Delta x} \right) \right\} + \frac{1}{\Delta y} \left\{ \mu_{i+1/2,j+1/2}^n \left(\frac{u_{i+1/2,j+1}^n - u_{i+1/2,j}^n}{\Delta y} + \frac{v_{i+1,j+1/2}^n - v_{i,j+1/2}^n}{\Delta x} \right) - \mu_{i+1/2,j-1/2}^n \left(\frac{u_{i+1/2,j}^n - u_{i+1/2,j-1}^n}{\Delta y} + \frac{v_{i+1,j-1/2}^n - v_{i,j-1/2}^n}{\Delta x} \right) \right\}$$

$$\mu_{i+1/2,j+1/2}^n = \frac{1}{4} (\mu_{i+1,j}^n + \mu_{i+1,j+1}^n + \mu_{i,j+1}^n + \mu_{i,j}^n)$$

$$\mu_{i+1/2,j-1/2}^n = \frac{1}{4} (\mu_{i+1,j-1}^n + \mu_{i+1,j}^n + \mu_{i,j}^n + \mu_{i,j-1}^n)$$


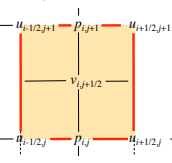
13. Approximating the velocity gradients by centered differences we find that the staggered grid provides the various velocity components exactly where we need them. For the u-component we need the x derivative of the u velocity on the left and the right boundary, and on the top and the bottom we need the y derivative of u and the x derivative of v. The viscosity, however, is given at the center of the pressure control volume, which works fine for the left and the right boundary, but for the top and the bottom we need to interpolate from the four surrounding pressure control volumes.

DNS of Multiphase Flows

Diffusion term: discrete term for the v-velocity

$$(D_y)_{i,j+1/2}^n = \frac{1}{\Delta x} \left\{ \mu_{i+1/2,j+1/2}^n \left(\frac{u_{i+1/2,j+1}^n - u_{i+1/2,j}^n}{\Delta y} + \frac{v_{i+1,j+1/2}^n - v_{i,j+1/2}^n}{\Delta x} \right) - \mu_{i+1/2,j-1/2}^n \left(\frac{u_{i+1/2,j}^n - u_{i+1/2,j-1}^n}{\Delta y} + \frac{v_{i+1,j-1/2}^n - v_{i,j-1/2}^n}{\Delta x} \right) \right\} + \frac{1}{\Delta y} \left\{ 2 \mu_{i,j+1}^n \left(\frac{v_{i,j+3/2}^n - v_{i,j+1/2}^n}{\Delta y} \right) - 2 \mu_{i,j}^n \left(\frac{v_{i,j+1/2}^n - v_{i,j-1/2}^n}{\Delta y} \right) \right\}$$

$$\mu_{i+1/2,j+1/2}^n = \frac{1}{4} (\mu_{i+1,j}^n + \mu_{i+1,j+1}^n + \mu_{i,j+1}^n + \mu_{i,j}^n)$$

$$\mu_{i+1/2,j-1/2}^n = \frac{1}{4} (\mu_{i+1,j-1}^n + \mu_{i+1,j}^n + \mu_{i,j-1}^n + \mu_{i,j}^n)$$


14. Similarly, the v-component is given by approximating the velocity gradients by centered differences and again we find that the velocity components are given where we need them, but that we need to interpolate the viscosity on the left and the right boundary.

DNS of Multiphase Flows

New Code for the diffusion terms, allowing unequal viscosities

```
for i=2:nx,for j=2:ny+1 % Temporary u-velocity-viscosity
    ut(i,j)=ut(i,j)+(2.0/(r(i,j+1)+r(i,j)))*dt*(...
        +(1./dx)*2.*(m(i+1,j)*(1./dx)*(u(i+1,j)-u(i,j)) - ...
            m(i,j) *(1./dx)*(u(i,j)-u(i-1,j))) ) ...
        +(1./dy)*( 0.25*(m(i,j)+m(i+1,j)+m(i+1,j+1)+m(i,j+1))* ...
            ((1./dy)*(u(i,j+1)-u(i,j)) + (1./dx)*(v(i+1,j)-v(i,j))) - ...
            0.25*(m(i,j)+m(i+1,j)+m(i+1,j-1)+m(i,j-1))* ...
            ((1./dy)*(u(i,j)-u(i,j-1)) + (1./dx)*(v(i+1,j-1)-v(i,j-1))) ) ) ;
end,end

for i=2:nx+1,for j=2:ny % Temporary v-velocity-viscosity
    vt(i,j)=vt(i,j)+(2.0/(r(i,j+1)+r(i,j)))*dt*(...
        +(1./dx)*( 0.25*(m(i,j)+m(i+1,j)+m(i+1,j+1)+m(i,j+1))* ...
            ((1./dy)*(u(i,j+1)-u(i,j)) + (1./dx)*(v(i+1,j)-v(i,j))) - ...
            0.25*(m(i,j)+m(i,j+1)+m(i-1,j+1)+m(i-1,j))* ...
            ((1./dy)*(u(i-1,j+1)-u(i-1,j)) + (1./dx)*(v(i,j)-v(i-1,j))) )...
            +(1./dy)*2.*(m(i,j+1)*(1./dy)*(v(i,j+1)-v(i,j)) - ...
            m(i,j) *(1./dy)*(v(i,j)-v(i,j-1))) ) ) ;
end,end
```

15. The modified code for the viscous terms is a little longer than for the simplified version but the structure is the same. We do each component separately since the number of grid points is different for the u and the v velocities and here we interpolate the viscosity directly in the expression for the viscous term. Since some of those are the same, we could have pre-computed them, at the cost of adding storage.

DNS of Multiphase Flows

After the density is updated, we need to set the viscosity as a function of the marker function

```
for i=1:nx+2,for j=1:ny+2 % Update the viscosity
    m(i,j)=m1+(m2-m1)*chi(i,j);
end,end
```

For a simple code as the one developed here we could use the density as a marker function. However, using a separate marker function and then set both density and viscosity as a function of the marker function is more general and allows us, for example, to examine flows of two fluid with the same density but different viscosities

16. Since the viscosity is different in the different fluids, it needs to be reset as the interface moves, just as the density. Although we really only need to visit points close to the interface, since those are the only ones that change, here we loop over all the grid points, to keep the code simple.

DNS of Multiphase Flows

Higher Order in Time

17. We started by using a one-sided approximation for the time derivative where the quantity that we are evolving at the next time level is given by the current value plus the right hand side—or the rate of change of the variable with time—multiplied by dt. This results in a method, often referred as Euler's method, that is only first order in time. This is usually not sufficiently accurate for serious computations.

DNS of Multiphase Flows

A widely used third order Runge-Kutta (RK-3) method. To integrate

$$\frac{du}{dt} = L(u)$$

in time, RK-3 involves taking three steps

$$u^1 = u^n + \Delta t L(u^n)$$

$$u^2 = \frac{3}{4}u^n + \frac{1}{4}u^1 + \frac{1}{4}\Delta t L(u^1)$$

$$u^{n+1} = \frac{1}{3}u^n + \frac{2}{3}u^2 + \frac{2}{3}\Delta t L(u^2)$$

We start by rewriting the steps as:

$$u^1 = u^n + \Delta t L(u^n)$$

$$u^2 = \frac{3}{4}u^n + \frac{1}{4}[u^n + \Delta t L(u^n)]$$

$$u^{n+1} = \frac{1}{3}u^n + \frac{2}{3}[u^2 + \Delta t L(u^2)]$$

18. The simplest way to make the time integration higher order is to use a two-step predictor corrector method, sometimes called Heun's method. It is easily shown that we can write the method as two first order Euler step, followed by an averaging of the initial and final value. However, we can do a little better with almost no effort and here we will implement a third order Runge-Kutta method in a very similar way. To simplify the notation we use the shorthand du over dt equal to L of u , for the Navier-Stokes equations. The third order Runge-Kutta method used here is usually written as three steps. First we do a simple Euler step where the right hand side is multiplied by dt and added to the old value, then we compute the right hand side using the new velocity and add it to a weighted sum of the original and new velocity and then we do the same thing again. As written here we have to store the intermediate velocities at each step. We can, however, rewrite it where that is not necessary. First we rewrite the steps as shown in the bottom part in the slide, where each new velocity is written as a weighted sum of an Euler step and the original velocity.

DNS of Multiphase Flows

Then we write the steps as three first-order (Euler) step, followed by a weighted average with u^n . Thus:

$$u^1 = u^n + \Delta t L(u^n) \quad \longrightarrow \quad \tilde{u} = u^n + \Delta t L(u^n) \quad \text{Euler step}$$

$$u^1 = \tilde{u}$$

$$u^2 = \frac{3}{4}u^n + \frac{1}{4}[u^n + \Delta t L(u^n)] \quad \longrightarrow \quad \tilde{u} = u^n + \Delta t L(u^1) \quad \text{Euler step}$$

$$u^2 = \frac{3}{4}u^n + \frac{1}{4}\tilde{u}$$

$$u^{n+1} = \frac{1}{3}u^n + \frac{2}{3}[u^2 + \Delta t L(u^2)] \quad \longrightarrow \quad \tilde{u} = u^2 + \Delta t L(u^2) \quad \text{Euler step}$$

$$u^{n+1} = \frac{1}{3}u^n + \frac{2}{3}\tilde{u}$$

The Euler steps are all identical

Note that u^1 and u^2 do not need to be stored

19. Using this rewrite it is easily seen that we can split each of the steps on the last slide into an Euler step followed by a weighted average with the original velocity at the n -th step. Using this rewrite, the conversion of our code from a first order explicit code to a third order explicit code is very simple. We simply take three first order steps, each one of which is followed by a weighted average of the new results and the value at the beginning of the step.

DNS of Multiphase Flows

Changes in the code for RK-3 order time integration

```
un=zeros(nx+1,ny+2); vn=zeros(nx+2,ny+1); % Used for
m=zeros(nx+2,ny+2); mn=zeros(nx+2,ny+2); % higher order
xfn=zeros(1,Nt+2); yfn=zeros(1,Nt+2); % in time

OTHER INITIALIZATION

%----- START TIME LOOP -----
for is=1:nstep,is
    un=u; vn=v; m=m; mn=mn; xfn=xf; yfn=yf; % Higher order
    for substep=1:3 % in time

MAIN TIME LOOP

        if substep==2, % Higher order (RK-3) in time
            u=0.75*un+0.25*u; v=0.75*vn+0.25*v; r=0.75*rn+0.25*r;
            m=0.75*mn+0.25*m; xfn=0.75*xfn+0.25*xf; yfn=0.75*yfn+0.25*yf;
            else substep==3
                u=(1/3)*un+(2/3)*u; v=(1/3)*vn+(2/3)*v; r=(1/3)*rn+(2/3)*r;
                m=(1/3)*mn+(2/3)*m; xfn=(1/3)*xfn+(2/3)*xf; yfn=(1/3)*yfn+(2/3)*yf;
            end

        end % End of sub-iteration for RK-3 time integration

%----- Add points to the front -----

REST OF TIME LOOP

end
```

20. The changes to the code are very minimal. In the main time integration loop we start by storing the current velocities, then we add a loop, which goes three times through the first order step that we had in the original code. At the end of each pass we average the new and the old value and the only complication is that the averaging is different in each pass. Thus, we need an if-statement to determine which averaging we should do. I note that it is important that we do not change the front during the substeps and this is the reason that we have put the restructuring of the front to the end of the time step, rather than right after we have moved it.

DNS of Multiphase Flows

Full code

Full code

21. We now have a fully functional code that is second order accurate in space, third order accurate in time, and includes surface tension and different viscosities in the different fluids.

[illegible]

**Full Front
Tracking Code
3rd order in time**

22. I don't really expect you to be able to read the code on this slide, but include it mainly to be able to insist that it still fits on one slide. Or to show that its size is still very modest, or about 280 lines in Matlab, including a few lines for diagnostics and post processing of the data, that I will discuss in the next lecture.

DNS of Multiphase Flows

The codes discussed here and in earlier lectures are available from both the course webpage and from dropbox directory:

<https://www.dropbox.com/s/38eunx1lkdg08hs/CodeC1-advChi.m?dl=0>

<https://www.dropbox.com/s/kdrhbt6qxm1qdnk/CodeC2-frm.m?dl=0>

<https://www.dropbox.com/s/ar7xg5uhpoy71bd/CodeC3-frm-st-BK3.m?dl=0>

The codes discussed here and in earlier lectures are available from both the course webpage and from dropbox directory:

<https://www.dropbox.com/s/38eunx1kdg08hs/CodeC1-advChi.m?dl=0>

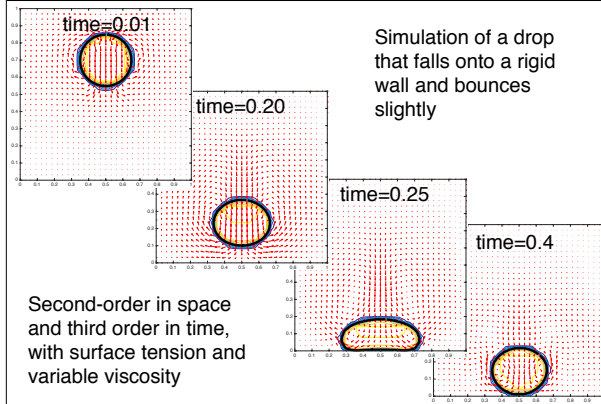
<https://www.dropbox.com/s/kdrhbt6qxm1qdnk/CodeC2-frt.m?dl=0>

<https://www.dropbox.com/s/ar7xg5uhpoy71bd/CodeC3-frt-st-RK3.m?dl=0>

23. The codes that we have developed can be downloaded both from the website with these lectures and also from a dropbox site by clicking the links listed here.

Results

24. We use the same initial condition as before, but set the surface tension to a finite value designed so that the drop remains nearly circular as it falls and hits the wall.



25. Unlike the zero surface tension drop that we simulated earlier, which continued to deform as it fell, here the surface tension keeps the drop nearly circular as it falls and it is only as it hits the bottom wall that it deforms to a significant degree. Then it bounces slightly off the wall before falling back and settling into a nearly circular shape.

This is a complete code, capable of solving real problems in an accurate way. It can, however, be improved in several ways. Some of the more obvious ones include:

- More robust advection scheme (ENO, QUICK, etc) to allow larger time steps for high Reynolds number flows

- More efficient pressure solver (Multigrid, Krylow methods such as GMRES, BICGSTAB, for example). In our current 3D code we use HYPRE.

- Non-uniform grids to allow rudimentary grid adaption

- More effective construction of the marker function

For production runs we would usually re-implement the code using a compile language such as fortran or c

26. The code that we have produced is a complete code that is capable of producing reasonably accurate results that would have been publishable in scientific journals when it was acceptable to present results for two-dimensional flows only. Indeed, some journals still accept such results. We can, however, improve the code in a number of ways, and after we have examined the accuracy of the results in the next lecture, we will extend it slightly by allowing unevenly spaced grid lines and incorporating a more robust scheme for the advection terms.