* Gas phase
  + Low density, low kinematic viscosity
  + Dynamics neglected
  + Bubble tracking necessary
    - Bubbles can be identified by flood-fill algorithm (connected components of gas phase)
    - Store in list
    - Formation model
      * Identify by connected components
      * Store initial volume in bubble list
    - Coalescence model
      * A
    - Separation model
      * B
    - Each step determine pressure by ideal gas law with constant “n”
* Liquid phase
  + Four types of cells
    - Gas (see above)
      * Dynamics neglected
    - Gas-liquid interface
      * Forms a complete Q-connected boundary separating gas and liquid
      * Special LBM dynamics
    - Liquid
      * Dynamics from LBM
    - Obstacle
      * Static
  + Cells described by state variables
    - Distribution functions F\_i (liquid and interface)
    - Bubble index (label matrix for interface and empty cells)
    - Fluid fraction (interface and obstacle)
    - Mass (interface) for fill fraction
    - Surface point (interface) for local curvature
    - Surface normal (interface) for local normal
  + Equation
    - are the discretized Boltzmann distributions for variables with velocity vector
    - are the lattice velocities associated with velocity vector
    - is the relaxation time for a distribution perturbed from equilibrium
    - is dependent on the macroscopic state variables density and velocity , not to be confused with the microscopic lattice velocities
    - Implementation is split into two parts:
      * Collision
        + Where is an intermediate distribution function
      * Streaming
        + Where the right hand side are the distribution functions from the previous time step and from the appropriate cell neighbor based on the lattice velocity
    - Implementation of phase space, i.e. choice of lattice velocities, is noted by DmQn, where m is the dimensionality (2 or 3) and n is the number of lattice velocity vectors.
      * Common choices are D2Q9, D3Q15, D3Q19, D3Q27.
      * Memory and cpu usage are directly related to choice of Q, since the entire phase space has to be modeled.
    - Macroscopic variable relationships
      * Mass density
      * Momentum density
      * Kinematic viscosity
        + is speed of sound in the fluid
  + Two-relaxation time model to eliminate some artifacts
    - Bars over variables denote symmetric quantity
    - Hats denote antisymmetric
    - Bars over index denotes negation of velocity vector
    - In the center cell, negation of index means self, so for the center cell
    - Then the equilibrium distribution functions follow in the same form as above
    - Then the collision operator changes to
  + Lattice velocity vectors are weighted based on our understand of the Boltzmann distribution, with larger lattice velocities being given smaller weight. The weight vector sums to one. The weights depend on choice of Q.
  + The equilibrium distribution functions may be found as
  + Obstacles do not participate in the dynamics. To simulate particles bouncing off the obstacles, the assignment is made when neighbors are obstacles. Note the bar on the LHS . This creates a no-slip boundary condition, which is ideal for most liquids.
  + External forces can be added during the collision step by adding terms of the form
    - where is the external body force vector.
  + Mass and fill fraction are related by .
    - is defined to be zero for gas, and one for liquid and solid.
    - Mass change can be determined depending on participating neighbor
      * For gas and obstacle neighbors, there is no mass change
      * For liquid neighbors, the exchange is
      * For interface, it is the same as liquid but weighted by mean fill fraction.
      * Mass is locally conserved by this operation, because participating neighbors exchange negated values.
    - Overall mass per cell increases through time steps by adding the current mass to sum of mass changes.
  + Gas cells do not participate in streaming, much like obstacles.
    - Distribution functions must be reconstructed there, but we can’t use bounceback directly because the boundary isn’t rigid in time.
    - Reconstruction has to meet two boundary conditions
      * Velocity of liquid and gas are equal at the interface.
      * Force exerted by gas on interface balanced by force exerted by fluid.
    - Method that meets both boundary conditions is
      * Note the bar on one of the equilibrium distribution functions, and on the RHS intermediate distribution function.
      * Furthermore, note the LHS is a neighbor, and RHS is self.
      * This method is applied INSTEAD of the usual streaming operation for all interface cells which have either a gas neighbor cell, or for which the relationship is true.
      * The function is the discretized normal vector.
      * is the gas pseudo-density, and can be computed as
        + Where is the lattice gas pressure, derived from the ideal gas law and initial volume of the relevant bubble.
        + is the lattice surface tension of the interface. It is not clear how to calculate this value from physical parameters right now.
        + is the local estimated curvature, calculated with the surface normal.
* Cell conversion
  + NOTE
    - In an ideal setup, the mass transfer should be below 1/10 cell volume
      * This means for max travel distance of 200 cells, we expect a minimum of 2000 time steps.
    - Neighboring cells follow converting cells to maintain contiguity of the interface.
    - When contradictions arise, precedence is given to the retention of a contiguous interface.
  + Solid never converts
  + Liquid can convert to interface
    - If a neighboring interface cell converts to gas
    - Fluid fraction set to 1
    - Mass is set equal to density
    - Recreate bubble indices.
      * Test neighboring interface cells
      * If only one index found, set to that inex
      * If more than one found coalescence is occurring
    - Coalesce bubbles
      * Find minimum bubble index
      * Sum initial bubble volumes and assign to minimum index
      * Sum current bubble volumes and assign to minimum index
      * Set all bubble indices involved to minimum index
      * Remove old bubble indices from tracking
    - Compute curvature
      * Calculate normals
        + Compute the negated weighted sum of fluid fraction over nearest neighbors

Weights are 1 for faces, sqrt(2) for edges, sqrt(3) for vertices

* + - * + Normalize
      * Calculate surface point
        + Identify planar segment of “true” fluid boundary that intersects the cell
        + Do this by bisecting planar offset along normal relative to cube center

Is it possible to do vectorized bisection in MATLAB?

Very likely if the function is vectorized

Have to track intervals (2 by n vec)

Function result is 1 by n vec

Compare vector with zero

Result of compare is 1 by n vec

If intervals are sorted ascending, and function is monotonically increasing, then result of compare is 0 (false) for changing the lower interval value, 1 (true) for the high interval value.

Adding 1 to the comparison result gives the index into the interval value requiring change.

Using this index in arrayfun should give us what we need pretty fast

Our function should be vectorizable…have to see if we can make that happen

* + - * + The function bisected is the volume of the cell “behind” the plane computed from the normal and plane offset.
        + The goal is to find a plane offset so that the computed fluid volume is close to the fluid volume arising from the LBM code
        + See Pohl for more details on bisection parameters
        + For fluid volume calculation…

Cube is highly symmetric, so can reduce space of consideration to 1/48th of full cube.

WLOG, if all four base x-y plane vertices are inside the fluid (fluid is always originating from x,y,z min point), then volume is the mean of the fluid-filled edge lengths

Otherwise, sum of triangular pyramidal subvolumes (up to 4)

These triangular pyramids are tetrahedra, so volume is 1/6 of the cross product of any two edge vectors.

There are up to three (have to check each axial edge for this occurring)

The fourth volume is anything from a triangular to hexagonal base pyramid, depending on number of other pyramids (more pyramid, more base points). The fourth volume is then 1/3 base times n dot p (normal times apex). Note this simplifies to 1/6 cross edges for the triangular pyramids as their base is a parallelogram, determinable as half cross product. Half times third is 1/6. The height in this case is always 1 due to the nature of edges in unit cubes.

* + - * Calculate surface curvature
        + Uses approximate mean curvature
        + Select appropriate subset of points

Only neighboring points

Only points in the interface

Constrained to distance limits based on a heuristic limit

Distance between neighbors in the interval [0.8,1.8]

Once this set has been determined, we need to check for points in nearly the same direction from the central point. To do so, loop around the points. If any pair have an angle between them of less than 30 degrees, drop the one closest to the central point. Dropping closer is a heuristic, chosen because it generally gives slightly higher accuracy.

* + - * + Triangulate

Compute simple mean normal by finding norm of equally-weighted: normal of the central point, normal of the surrounding points.

Project the surface points onto plane whose normal is the simple mean normal.

Pick any surrounding point, enumerate counter-clockwise (compared to direction of mean normal, recall normal point toward gas phase)

Triangles are formed by any two neighboring surrounding points and the central point, arranged counter-clockwise.

* + - * + Calculate

See appendix A of Pohl.

3D case requires tensor to describe curvature.

Directional curvature is a quadratic form where is the tangent vector to the surface at point , and is the curvature 2-tensor at point .

The directional curvatures are the diagonal entries.

The directional curvatures are associated with the basis of the tangent space to the surface at point , from which the tangent vector is formed parametrically.

The basis vectors are the principal directions of the surface precisely when the off-diagonal entries are both zero.

When this is true, the directional curvatures become the principal curvatures, which are of interest to this problem.

The normal to the surface is the normal to the tangent space, so the normal vector and tangent basis form an orthogonal basis when the tangent vectors are the principal directions.

Can augment the directional curvature matrix to arbitrary directions by adding the normal vector to the parameterization, with associated curvature diagonal entry of 0.

Then any arbitrary linear transformation of the augmented basis has eigenvalues equal to the associated directional curvatures when the quadratic form is applied to the tangent vector expressed in that transformed basis.

To obtain the principal curvatures, a Housholder transformation is applied to the tangent plane.

To approximate the directional curvatures for any unit length tangent vector a Taylor expansion of an arc lying on the surface in the plane of the normal and tangent vector is used, of second order.

It turns out after some linear algebra that the directional curvature is the limit

To determine

Compute normalized weighted sum of normal vectors of triangle faces determined earlier.

Weights are face areas.

Calculate approximation matrix using the sum of

is the normalized vector for central point p

is a linearization of the curvature for the point given as

is the sum of the face areas of the faces which contain as a vertex, normalized by twice the total area of all faces involved (so the sum of is 1)

The mean curvature is then the trace of the previous matrix.

* + - Use in reconstruction of missing distribution functions
  + Interface can convert to liquid or gas
    - Interface to liquid when fluid fraction greater than 1+TOL for some small TOL
    - Interface to gas when fluid fraction less than -TOL
    - TOL is taken to be 0.001, and prevents rapid flip-flopping of interface cells.
    - When mass transfers, the fluid fraction isn’t guaranteed to be in the proper [0,1] interval.
      * The “excess” mass is computed on conversion
      * The excess is then redistributed to neighboring interface cells
      * If the excess can’t be redistributed locally it is lost totally, causing global mass destruction/creation and violating mass conservation. This happens unfrequently and the loss is small. (We can track as needed)
  + Gas can convert to interface
    - If a neighboring interface cell converts to liquid
    - Have to construct distribution functions for the cell
      * For each such cell, find the neighboring interface and liquid cells
      * Compute the mean density of that set
      * Compute the mean velocity of that set
      * Compute from
      * Assign to the intermediate distribution function
      * Set fluid fraction and mass to 0 (it may gain some during redistribution)
  + Bubble segmentation
    - Not handled in the model.
    - What do we do if it happens?
    - Author claims flood-fill algorithm is too expensive, not sure I agree…
    - Why not use a flood-fill on the fluid cells only? If the values disagree with the actual number of bubbles, it should be easy to map out a new bubble.
      * Find first mismatch in flood fill and actual index
      * Create new bubble with new index in bubble aggregate
      * Separate volumes by occupied space
      * Each bubble will initially have the pressure of the initial bubble
      * Partition initial volume proportionally with current volume
    - Possible superior algorithm is to test only interface cell neighbors for differing IDs for detecting coalescence.
      * If ID of one interface has neighbor with diff ID, then merge bubbles
    - Anderl et al have a method for faster detection of bubble segmentation/splitting, but it isn’t clear what it is from the paper.
      * Part of it appears to be checking for anti-parallel normal of neighboring bubbles, which would require storing normal information
      * This also doesn’t sound very robust
      * A really robust method would be to check using flood fill (connected components) but it is claimed this is expensive.
  + Maximum lattice velocity must be much less than lattice speed of sound squared (so 1/9)
    - Adaptive time stepping possible, involves recomputing certain physical parameters (body forces, etc) with lattice units, see Thurey et al
    - Reduce time step if largest speed is greater than 5/24 in lattice units (1/3 \* 1/2 divided by 4/5).
      * 1/3 is the criteria for stability
      * 1/2 is a safety factor
      * 4/5 is the time step adjustment factor (so when we multiply later, we get to exactly 1/6 of max allowable safe speed)
    - New time step is 4/5 of current
    - Omega is adjusted by
    - Body force adjusted by
    - Density adjusted around median by multiplication, velocity by multiplication (when computed in loop)
      * where V is sum of fill fraction, M is sum of masses
    - Mass and fill value also change
    - The non-equilibrium values (of distribution function)
    - Time step can be enlarged when max velocity is less than 1/15 (1/3\*1/2\*4/5).
    - Smaller time stepping results in larger omega, and omega can’t be larger than 1.99 for stability. If this occurs, the only accurate way out is to refine the spatial mesh.
    - Time step is also affected by choice of body force, and it may need to be varied for stability. To change the body force requires changing the space step (or time step) appropriately.
    - To determine these sorts of soft limits on space step and such we will probably have to poll some common alloy properties (Fe, Al, Zn, Cu, Mg, Ti)