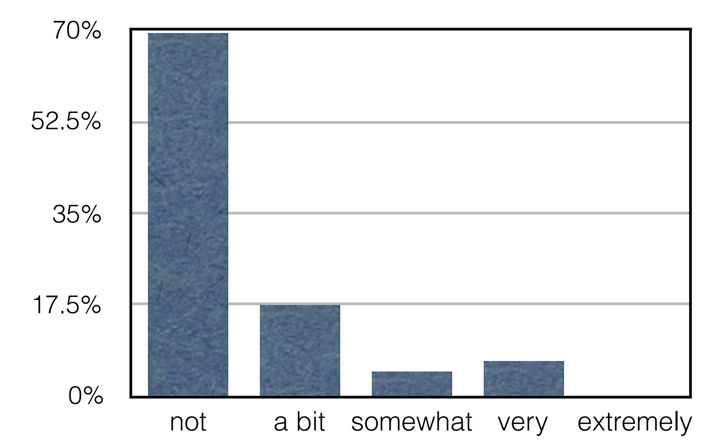
# Muddiest Points from Class 04/03

- *"In crank-nicholson/adams-bashforth methods, will we have to modify the explict convective term at the boundaries (for every time step) like we did for crank-nicholson ADI?"*
- *"I am still confused about how to deal with the Adams-Bashforth part if we want to code ADI. You said it can be calculated before each time step but the first term is in time level n as well."*
  - The explicit convective terms (AB) remain frozen throughout the CN/ADI solve.
  - The explicit convective terms (AB) only involve known time levels n and n-1 and thus can be calculated at the beginning of the CN/ADI solve.
- *"Will solutions to these problems ask for the data obtained at the cell centers and cell faces?"*
  - Depends on the problem you are solving. You may have to interpolate to a specific point in space you are interested in.
- *"What is the advantage of using staggered meshes? It seems like they would not give any more accuracy vs normal meshes."*
  - We'll cover a major advantage when we finally solve Navier-Stokes. No advantage for Burger's equation really.
- *"In the case the staggered mesh, we are using data from a 2-times finer mesh, here we have 2MN-M-N nodes corresponding to a MxN cell centered mesh. So would it be reasonable to code the mesh nodes as a 2Mx2N matrix with nodes where there are no nodes are kept blank or is there another way?"*
  - DO NOT think about staggered meshes in this way. It will lead to a world of pain.
  - u and v are defined on "different meshes". More later today.
- *"Is the method for coding these cell face meshes similar to how we code the alpha flux term in HW9?"*
  - Yes
- *"Is it possible to use the implicit in space method but still make use of the conservative from for a non-linear PDE like in Class22-Slide8?"*
  - Yes, but similar time-lagging is required to linearize the equations



## Burgers Equation on Staggered Mesh

need velocity boundary condition @ velocity locations!

Example: tangentially moving bottom wall

- velocity in y-direction:

$$v_{i,\frac{1}{2}} = 0$$

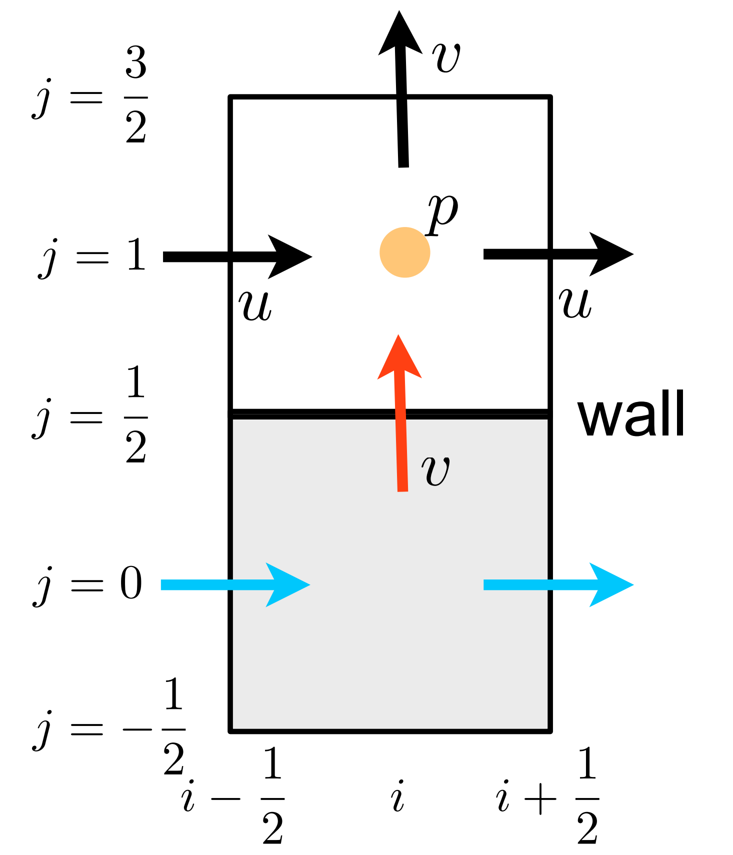
- what about x-direction velocity?
  - we don't have u velocities defined at the wall!
  - idea: let's define ghost cell velocities

$$u_{i+\frac{1}{2},0} = ?$$

  - use ghost cell velocity to determine wall velocity

$$u_{wall} = \frac{1}{2}\left(u_{i+\frac{1}{2},1} + u_{i+\frac{1}{2},0}\right)$$

$$\Rightarrow \quad u_{i+\frac{1}{2},0} = 2u_{wall} - u_{i+\frac{1}{2},1}$$



## Burgers Equation on Staggered Mesh

need velocity boundary condition @ velocity locations!

Example: inlet in the bottom boundary

- velocity in y-direction (normal to inlet):

$$v_{i,\frac{1}{2}} = f(x,t)$$

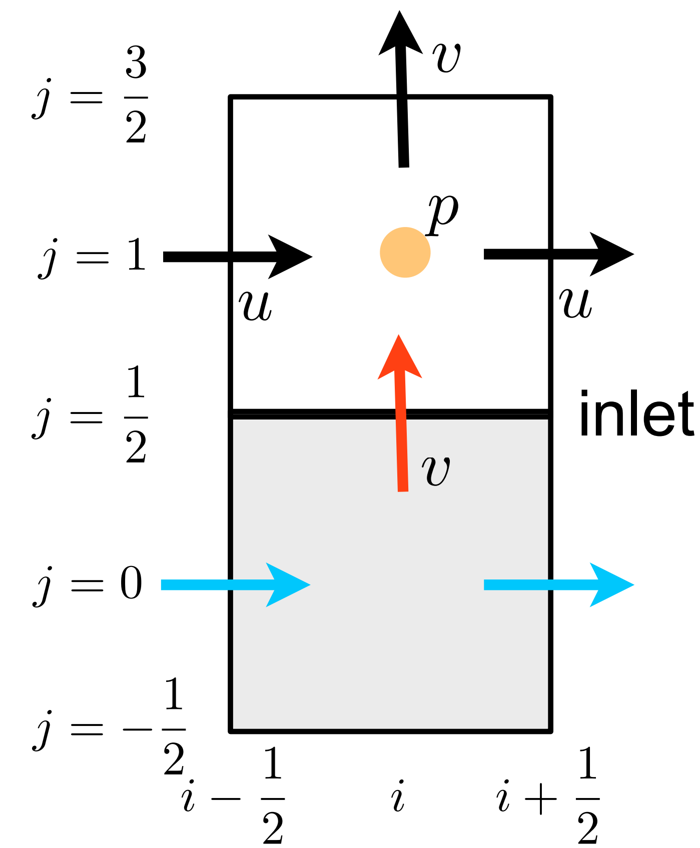
- what about x-direction velocity (tangential to inlet)?
  - again we don't have u velocities defined at the boundary!
  - again let's define ghost cell velocities

$$u_{i+\frac{1}{2},0} = \,?$$

  - use ghost cell velocity to determine inlet velocity

$$u_{inlet} = \frac{1}{2}\left(u_{i+\frac{1}{2},1} + u_{i+\frac{1}{2},0}\right)$$

$$\Rightarrow \quad u_{i+\frac{1}{2},0} = 2\,u_{inlet} - u_{i+\frac{1}{2},1}$$




  - for flow only normal to the inlet, the tangential velocity is zero
  - to have inflow at an angle to the boundary, use both normal and tangential components

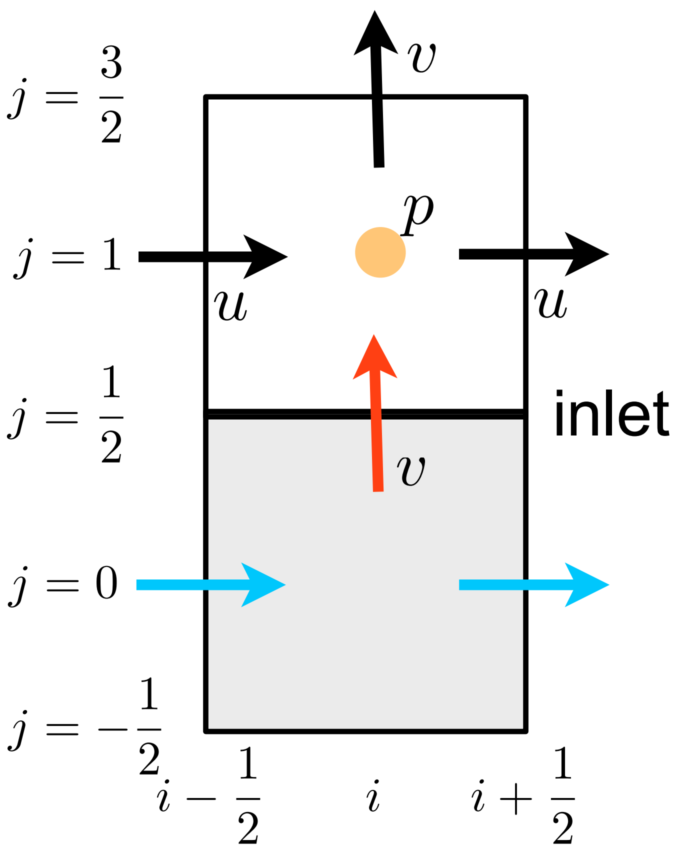
## Burgers Equation on Staggered Mesh

need velocity boundary condition @ velocity locations!

Example: outlet in the bottom boundary

- velocity should be set from information inside of the domain to respect direction of information transport (characteristics)
- idea: extrapolate velocity to the boundary

$$v_{i,\frac{1}{2}} = v_{i,\frac{3}{2}} \qquad \text{or} \qquad v_{i,\frac{1}{2}} = 2v_{i,\frac{3}{2}} - v_{i,\frac{5}{2}}$$




- what about x-direction velocity (tangential to outlet)?
  - again we don't have u velocities defined at the boundary!
  - can use zero gradient Neumann condition

$$u_{i+\frac{1}{2},0} = u_{i+\frac{1}{2},1}$$

  - can also use zero value Dirichlet condition: makes the flow normal to the outlet

$$u_{i+\frac{1}{2},0} = -u_{i+\frac{1}{2},1}$$

  - no general rule which one is better: depends on the case, e.g., channels: Dirichlet

## Burgers Equation on Staggered Mesh

some comments on coding:

- vector/array indices must be whole numbers, i.e. integers ⇒ cannot use i+1/2
- you must decide what `u(i,j)` refers to: $u_{i+\frac{1}{2},j}$ or $u_{i-\frac{1}{2},j}$
- similar considerations apply to `v(i,j)`
- my suggestion:

`u(i,j)` = $u_{i+\frac{1}{2},j}$

`v(i,j)` = $v_{i,j+\frac{1}{2}}$

## Comments on Programming

the following are some good practices, but do not replace a good programming class

- use functions/subroutines to structure your program

```
call initialConditions(u,v,phi)
do while (t < tend)
   dt = calcTimeStep(u,v)
   call solveBurgers2D(u,v,…)
   t = t+dt
   istep = istep + 1
   call doPostProcessing(u,v,…)
end
```

- use variables for mesh sizes: M and N
- preallocate all vectors and arrays

```
M = 40; N = 20;
u = zeros(M+1,N+2);
v = zeros(M+2,N+1);
```

```
M = 40; N = 20;
allocate(u(0:M  ,0:N+1))
allocate(v(0:M+1,0:N  ))
```

## Comments on Programming

the following are some good practices, but do not replace a good programming class

- in Matlab, arrays are matrices, where the first index is row# and the second column#
- can use Matlab functions `transpose()` (and/or `flipud()`) for plotting
- run times can be long, so to give peace of mind that your code is still running properly, output some diagnostics after a fixed number of time steps, for example, min/max velocities, etc.

```
if (mod(istep,100) == 0) then
   print some diagnostics
end if
```

- use Matlab's build in **<u>debugger</u>** to locate bugs!!!!
- test individual pieces of your code = subroutines/functions separately

## Comments on Programming

$$\frac{\partial u}{\partial t} + \frac{\partial (uu)}{\partial x} + \frac{\partial (uv)}{\partial y} = \nu \left( \frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} \right) \qquad \frac{\partial v}{\partial t} + \frac{\partial (uv)}{\partial x} + \frac{\partial (vv)}{\partial y} = \nu \left( \frac{\partial^2 v}{\partial x^2} + \frac{\partial^2 v}{\partial y^2} \right)$$

- u and v have different array sizes for staggered meshes => solve them in separate loops
- for FTCS what's the stable time step in 2D?

hyperbolic part: use product rule / chain rule to bring hyperbolic parts into this form:

$$\frac{\partial u}{\partial t} + a\frac{\partial u}{\partial x} + b\frac{\partial u}{\partial y} = \dots \qquad \frac{\partial v}{\partial t} + c\frac{\partial v}{\partial x} + d\frac{\partial v}{\partial y} = \dots$$

$$\Delta t_u = CFL \cdot \frac{\min(\Delta x, \Delta y)}{\max(|a|) + \max(|b|)} \qquad \Delta t_v = CFL \cdot \frac{\min(\Delta x, \Delta y)}{\max(|c|) + \max(|d|)}$$

careful if denominator = 0

$$\Delta t_{hyperbolic} = min(\Delta t_u, \Delta t_v)$$

$$\mathrm{Re}_{u,c} = \frac{(\max(|a|) + \max(|b|)) \max(\Delta x, \Delta y)}{\nu} \qquad \mathrm{Re}_{v,c} = \frac{(\max(|c|) + \max(|d|)) \max(\Delta x, \Delta y)}{\nu}$$

parabolic part: $\Delta t_{parabolic} = CFL \cdot \dfrac{\Delta x^2 \Delta y^2}{2\nu \left(\Delta x^2 + \Delta y^2\right)}$

overall: $\Delta t = \min\left(\Delta t_{hyperbolic}, \Delta t_{parabolic}\right)$