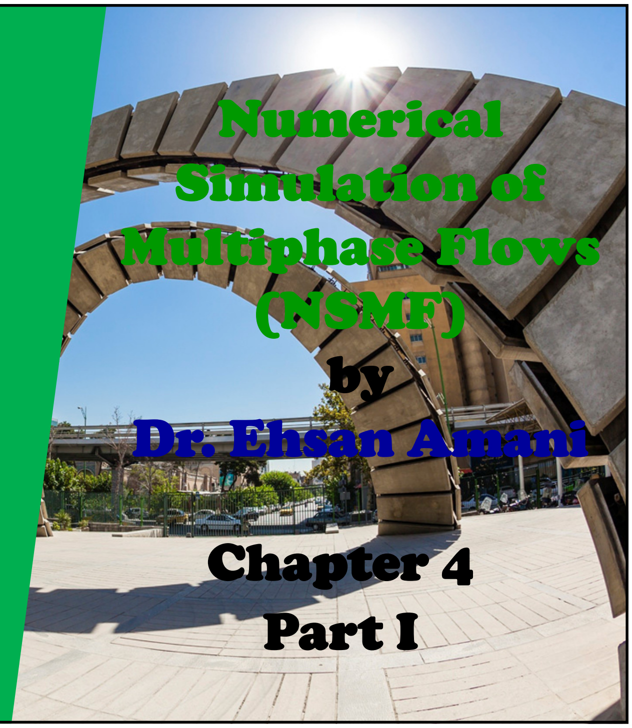


Numerical Simulation of Multiphase Flows (NSMF)

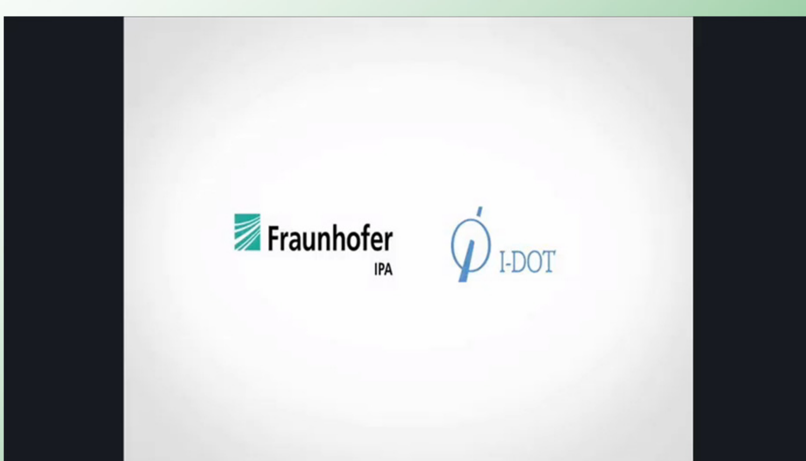
by

Dr. Ehsan Amani

Chapter 4
Part I

Case study #1: 3D printing

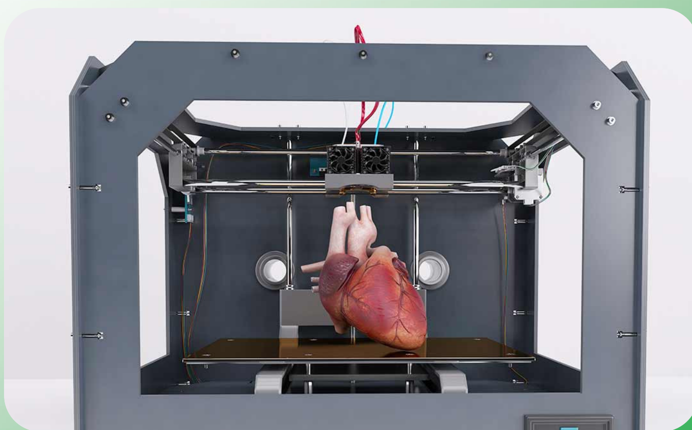
- Drop-On-Demand (DOD)



Chap 4 By E. Amani

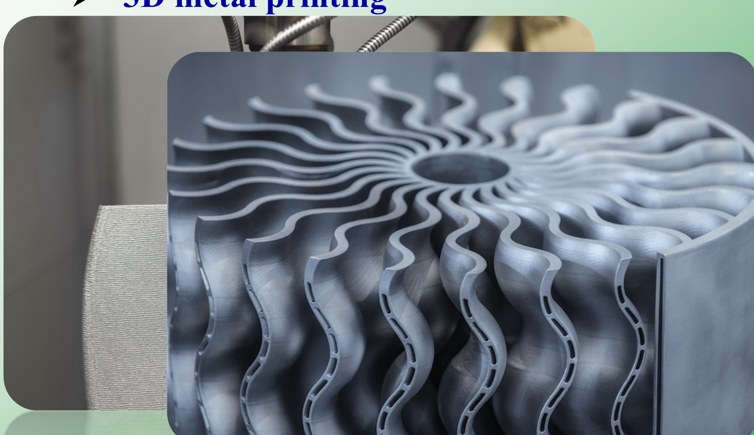
Case study #1: 3D printing

- **Applications:**
  - **Tissue engineering**



Case study #1: 3D printing

- **Applications:**
  - **Tissue engineering**
  - **3D metal printing**

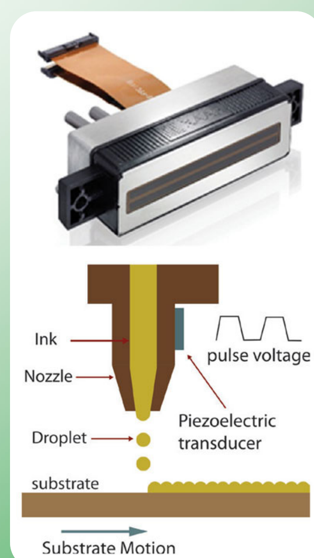


Case study #1: 3D printing

- **Applications:**
  - **Tissue engineering**
  - **3D metal printing**
  - **…**

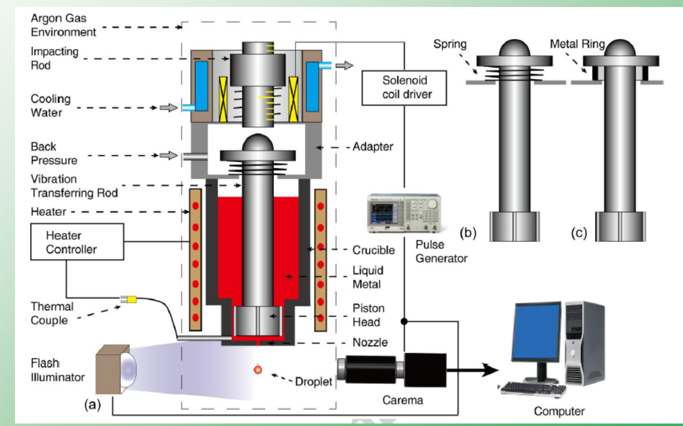
Case study #1: 3D printing

- **Actuation type:**
  - **Piezoelectric**



Case study #1: 3D printing

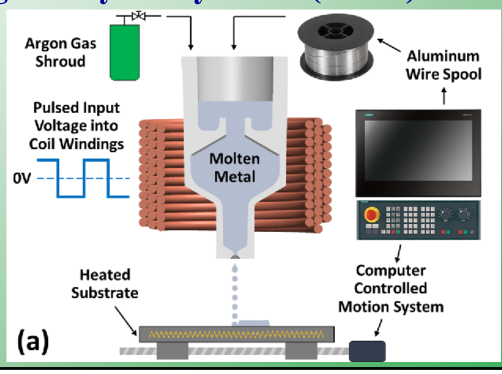
- Actuation type:
  - Piezoelectric
  - Impact-driven



Chap 4 By E. Amani

Case study #1: 3D printing

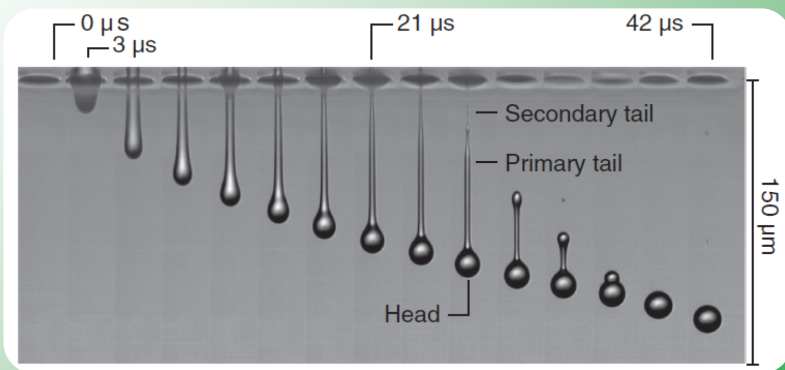
- Actuation type:
  - Piezoelectric
  - Impact-driven
  - …
  - MagnetoHydroDynamic (MHD)



Chap 4 By E. Amani

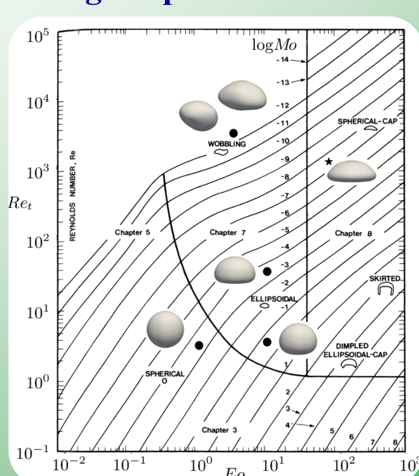
Case study #1: 3D printing

- **Sub-problems:**
  - **Droplet generation**



Case study #1: 3D printing

- **Sub-problems:**
  - **Free-falling drop**



Case study #1: 3D printing

- **Sub-problems:**
  - **Drop substrate impact and solidification**

Case study #1: 3D printing

- **Sub-problem #2:**
  - **Free-falling drop**

$$U = f(\rho_l, \rho_g, \mu_l, \mu_g, \Delta\rho g, d, \sigma, t)$$

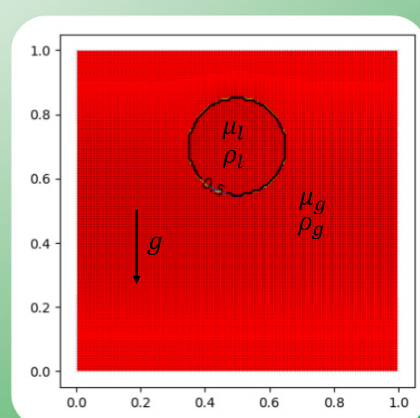
$$Fr = f(r, m, \mathrm{Ar}, \mathrm{Eo}, t^*)$$

$$r = \rho_g/\rho_l \qquad m = \mu_g/\mu_l$$

$$\mathrm{Ar} = \frac{g\rho_l \Delta\rho d^3}{\mu_l^2}$$

$$Eo = \frac{\Delta\rho g d^2}{\sigma}$$

$$t^* = \frac{t}{\sqrt{\rho_l/\Delta\rho g}}$$



Case study #1: 3D printing

- **Sub-problem #2:**
  - **Free-falling drop**

$\rho_g = 1\ kg/m^3 \quad \rho_l = 20\ kg/m^3$

$\mu_l = \mu_g = 0.01\ Pa.s$

$\sigma = 0.1$ N/m

$d = 0.3\ m$

$g = 10\ m/s^2$

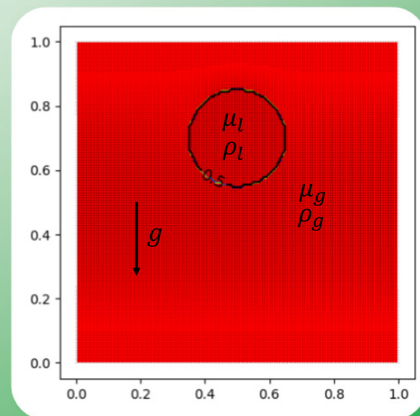
$r = 0.05 \quad m = 1$

$$\text{Ar} = \frac{g\rho_l \Delta\rho d^3}{\mu_l^2} = 1.03 \times 10^4$$

$$Eo = \frac{\Delta\rho g d^2}{\sigma} = 171$$

$$M = \frac{Eo^3}{Ar^4} = 4.44 \times 10^{-10} \longrightarrow \log M = -9.35$$

Grid: 32×32 $\Delta t = 0.00125\ s$



Chap 4 **By E. Amani**

Case study #1: 3D printing

- **Sub-problem #2:**
  - **Free-falling drop**

$\rho_g = 1\ kg/m^3 \quad \rho_l = 20\ kg/m^3$

$\mu_l = \mu_g = 0.01\ Pa.s$

$\sigma = 0.1$ N/m

$d = 0.3\ m$

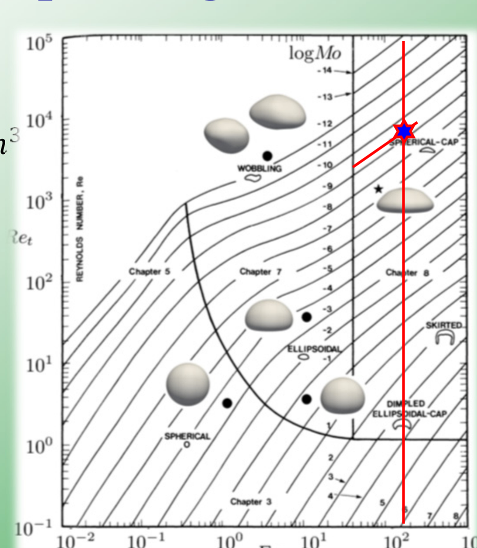
$g = 10\ m/s^2$

$r = 0.05 \quad m = 1$

$$\text{Ar} = \frac{g\rho_l \Delta\rho d^3}{\mu_l^2} = 1.03 \times 10^4$$

$$Eo = \frac{\Delta\rho g d^2}{\sigma} = 171$$

$$M = \frac{Eo^3}{Ar^4} = 4.44 \times 10^{-10} \longrightarrow \log M = -9.35$$



Chap 4 **By E. Amani**

DNS: The first step

- **Incompressible variable-density NS**

$$\frac{\partial}{\partial t}(\rho u_i) + \frac{\partial}{\partial x_j}(\rho u_i u_j) = -\frac{\partial p}{\partial x_i} + \frac{\partial}{\partial x_j}\overbrace{\left[\mu\left(\frac{\partial u_i}{\partial x_j} + \frac{\partial u_j}{\partial x_i}\right)\right]}^{D_{ij}} + \rho g_i$$

No surface tension

$$\frac{\partial u_j}{\partial x_j} = 0$$

$$\frac{\partial H}{\partial t} + \frac{\partial}{\partial x_j}(u_j H) = 0 \begin{cases} H = \chi_1 \\ S_m^{(I_k)} = 0 \end{cases}$$

← *No mass transfer*

$$\rho = H\rho_1 + (1-H)\,\rho_2$$
$$\mu = H\mu_1 + (1-H)\,\mu_2$$

← *Two-phase*

Chap 4 **By E. Amani**

DNS: The first step

- **Incompressible variable-density NS (tensorial notation)**

$$\frac{\partial}{\partial t}(\rho u_i) + \frac{\partial}{\partial x_j}(\rho u_i u_j) = -\frac{\partial p}{\partial x_i} + \frac{\partial}{\partial x_j}\overbrace{\left[\mu\left(\frac{\partial u_i}{\partial x_j} + \frac{\partial u_j}{\partial x_i}\right)\right]}^{D_{ij}} + \rho g_i$$

$$\boldsymbol{u} = u_1\boldsymbol{i} + u_2\boldsymbol{j} + u_3\boldsymbol{k} = u\boldsymbol{i} + v\boldsymbol{j} + w\boldsymbol{k}$$

$$\boldsymbol{\nabla} = \boldsymbol{i}\frac{\partial}{\partial x_1} + \boldsymbol{j}\frac{\partial}{\partial x_2} + \boldsymbol{k}\frac{\partial}{\partial x_3} = \boldsymbol{i}\frac{\partial}{\partial x} + \boldsymbol{j}\frac{\partial}{\partial y} + \boldsymbol{k}\frac{\partial}{\partial z}$$

$$\frac{\partial}{\partial t}(\rho\boldsymbol{u}) + \boldsymbol{\nabla}.(\rho\boldsymbol{u}\boldsymbol{u}) = -\boldsymbol{\nabla}p + \boldsymbol{\nabla}.\overbrace{[\mu(\boldsymbol{\nabla}\boldsymbol{u} + (\boldsymbol{\nabla}\boldsymbol{u})^T)]}^{\boldsymbol{D}} + \rho\boldsymbol{g}$$

Chap 4 **By E. Amani**

DNS: The first step

- **Incompressible variable-density NS (tensorial notation)**

$$\frac{\partial}{\partial t}(\rho\boldsymbol{u}) + \boldsymbol{\nabla}.(\rho\boldsymbol{u}\boldsymbol{u}) = -\boldsymbol{\nabla}p + \boldsymbol{\nabla}.\overbrace{[\mu(\boldsymbol{\nabla}\boldsymbol{u} + (\boldsymbol{\nabla}\boldsymbol{u})^T)]}^{\boldsymbol{D}} + \rho\boldsymbol{g}$$

$$\boldsymbol{\nabla}.\boldsymbol{u} = 0$$

$$\frac{\partial H}{\partial t} + \boldsymbol{\nabla}.(\boldsymbol{u}H) = 0$$

$$\rho = H\rho_1 + (1-H)\,\rho_2$$
$$\mu = H\mu_1 + (1-H)\,\mu_2$$

Chap 4 By E. Amani

Finite Volume (FV) discretization

- **Equations: Integral form**

$$\frac{\partial}{\partial t}\int_V \rho\boldsymbol{u}dV + \int_V \boldsymbol{\nabla}.(\rho\boldsymbol{u}\boldsymbol{u})dV = -\int_V \boldsymbol{\nabla}.(p\boldsymbol{I})dV + \int_V \boldsymbol{\nabla}.\boldsymbol{D}dV + \int_V \rho\boldsymbol{g}dV$$

Divergence theorem $\int_V \boldsymbol{\nabla}.\boldsymbol{A}dV = \oint_S \boldsymbol{A}.\boldsymbol{n}dS$

$$\frac{\partial}{\partial t}\int_V \rho\boldsymbol{u}dV + \oint_S \boldsymbol{u}(\rho\boldsymbol{u}.\boldsymbol{n})dS = -\oint_S p\boldsymbol{n}dS + \oint_S \boldsymbol{D}.\boldsymbol{n}dS + \int_V \rho\boldsymbol{g}dV$$

Chap 4 By E. Amani

Finite Volume (FV) discretization

DNS of Multiphase Flows **by G. Tryggvason**

Navier-Stokes equations in integral form

$$\frac{\partial}{\partial t}\int_V \rho \mathbf{u}\, dv + \oint_S \rho \mathbf{u}\mathbf{u}\cdot \mathbf{n}\, ds = -\oint_S p\mathbf{n}\, ds + \int_V \rho \mathbf{g}\, dv + \oint_S \mu(\nabla \mathbf{u} + (\nabla \mathbf{u})^T)\cdot \mathbf{n}\, ds + \int_V \mathbf{f}\, dv$$

Where the pressure is such that the flow is incompressible

$$\oint_S \mathbf{u}\cdot \mathbf{n}\, ds = 0$$

And the density of each fluid particle is constant

$$\frac{D\rho}{Dt} = \frac{\partial \rho}{\partial t} + \mathbf{u}\cdot \nabla \rho = 0$$

V: Control volume
S: Control surface

Notation

ρ	$\mathbf{u}$	p	Original variables
ρ_h	$\mathbf{u}_h$	p_h	Numerical approximation
$\rho_{i,j}$	$\mathbf{u}_{i,j}$	$p_{i,j}$	Numerical approximation at point (i,j)

Chap 4 **By E. Amani**

Computational grid

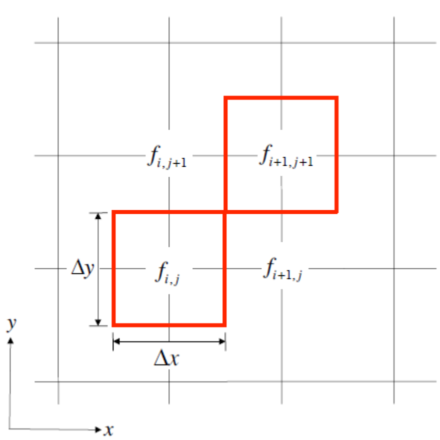
DNS of Multiphase Flows **by G. Tryggvason**

Select rectangular control volume defined by a structured grid.

(cell-centered)

Here we will assume that all the control volumes are the same size

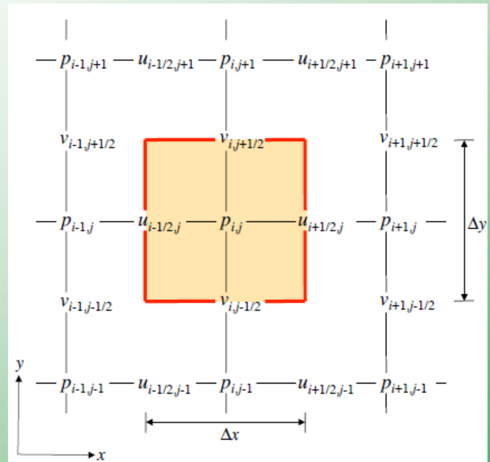
For second order approximations the average value is a good approximation for the value in the center



Chap 4 **By E. Amani**

Computational grid

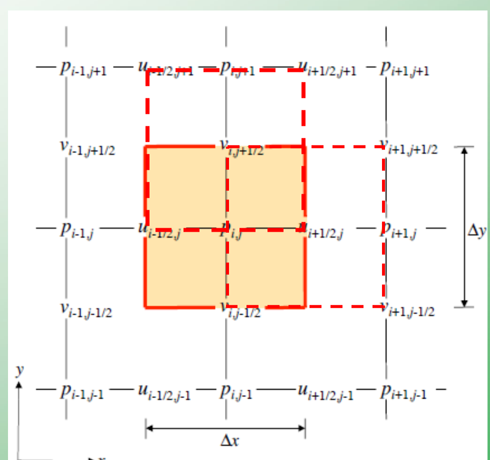
- **Odd-even decoupling: Staggered grid**



Chap 4 By E. Amani

Computational grid

- **Odd-even decoupling: Staggered grid**



Chap 4 By E. Amani

Computational grid

DNS of Multiphase Flows **by G. Tryggvason**

$$\frac{\partial}{\partial t}\int_V \rho\mathbf{u}dv + \oint_S \rho\mathbf{u}\mathbf{u}\cdot\mathbf{n}ds = -\oint_S p\mathbf{n}ds + \int_V \rho\mathbf{g}dv + \oint_S \mu(\nabla\mathbf{u} + (\nabla\mathbf{u})^T)\cdot\mathbf{n}ds + \int_V \mathbf{f}dv$$

The average value of each term, over the control volume:

$$\mathbf{M}_h = \frac{1}{V}\int_V \rho\mathbf{u}dv \qquad \rho_h\mathbf{g} = \frac{1}{V}\int_V \rho\mathbf{g}dv$$

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

$$\nabla_h p_h = \frac{1}{V}\oint_S p\mathbf{n}\, ds \qquad \mathbf{f}_h = \frac{1}{V}\int_V \mathbf{f}dv$$

V: Control volume
S: Control surface

The Navier-Stokes equations are then:

$$\frac{\partial}{\partial t}\mathbf{M}_h = -\mathbf{A}_h - \nabla_h p_h + \rho_h\mathbf{g} + \mathbf{D}_h + \mathbf{f}_h$$

Similarly, the incompressibility conditions is

$$\nabla_h\cdot\mathbf{u}_h = \frac{1}{V}\oint_S \mathbf{u}\cdot\mathbf{n}ds$$

Chap 4 **By E. Amani**

Time discretization: Semi-discrete form

DNS of Multiphase Flows **by G. Tryggvason**

Decompose the momentum into density and velocity

$\mathbf{M}_h^n = \rho_h^n\mathbf{u}_h^n$ and $\mathbf{M}_h^{n+1} = \rho_h^{n+1}\mathbf{u}_h^{n+1}$

Where

$$\mathbf{u}_h = \frac{1}{V}\int_V \mathbf{u}dv \qquad \rho_h = \frac{1}{V}\int_V \rho dv$$

This implies that the velocity and the density are slowly varying

The semi-discrete Navier-Stokes equations then become:

$$\frac{\rho_h^{n+1}\mathbf{u}_h^{n+1} - \rho_h^n\mathbf{u}_h^n}{\Delta t} = -\mathbf{A}_h^n - \nabla_h p_h + \rho_h^n\mathbf{g} + \mathbf{D}_h^n + \mathbf{f}_h^n$$

First-Order Upwind (FOU) time discretization

Supplemented by:

$$\nabla_h\cdot\mathbf{u}_h^{n+1} = 0$$

Notation
$(\ )^n$: at t
$(\ )^{n+1}$: at $t+\Delta t$

Chap 4 **By E. Amani**

Pressure-velocity coupling

DNS of Multiphase Flows **by G. Tryggvason**

To integrate in time we approximate the time derivative by a simple first order in time forward approximation:

$$\frac{\rho_h^{n+1}\mathbf{u}_h^{n+1} - \rho_h^n \mathbf{u}_h^n}{\Delta t} + \mathbf{A}_h^n = -\nabla_h p_h + \mathbf{g}_h^n + \mathbf{D}_h^n + \mathbf{f}_h^n$$

Then we split it into two steps:

Predictor:

$$\frac{\rho^{n+1}\mathbf{u}_h^* - \rho^n \mathbf{u}_h^n}{\Delta t} = -\mathbf{A}_h^n + \rho_h^n \mathbf{g} + \mathbf{D}_h^n + \mathbf{f}_h^n$$

Corrector:

$$\frac{\rho_h^{n+1}\mathbf{u}_h^{n+1} - \rho_h^{n+1}\mathbf{u}_h^*}{\Delta t} = -\nabla_h p_h$$

The pressure must ensure that

$$\nabla_h \cdot \mathbf{u}_h^{n+1} = 0$$

Projection Method

Chap 4 By E. Amani

Pressure-velocity coupling

DNS of Multiphase Flows **by G. Tryggvason**

By taking the divergence of

$$\frac{\rho_h^{n+1}\mathbf{u}_h^{n+1} - \rho_h^{n+1}\mathbf{u}_h^*}{\Delta t} = -\nabla_h p_h$$

and using

$$\nabla_h \cdot \mathbf{u}_h^{n+1} = 0$$

we obtain the pressure equation

$$\frac{\cancelto{0}{\nabla_h \cdot \mathbf{u}_h^{n+1}} - \nabla_h \cdot \mathbf{u}_h^*}{\Delta t} = -\nabla_h \cdot \left(\frac{1}{\rho_h^{n+1}}\nabla_h p_h\right)$$

$$\nabla_h \cdot \left(\frac{1}{\rho_h^{n+1}}\nabla_h p_h\right) = \frac{1}{\Delta t}\nabla_h \cdot \mathbf{u}_h^*$$

We will take the last step using the discrete versions of the corrector equation and the incompressibility conditions

Chap 4 By E. Amani

Pressure-velocity coupling

DNS of Multiphase Flows **by G. Tryggvason**

Discretization in time

1. Update the marker function to find new density and viscosity

2. Find a temporary velocity using the advection and the diffusion terms only:

$$\left\{ \mathbf{u}_h^* = \frac{1}{\rho_h^{n+1}}\left(\rho_h^n \mathbf{u}_h^n + \Delta t\left(-\mathbf{A}_h^n + \rho_h^n \mathbf{g} + \mathbf{D}_h^n + \mathbf{f}_h^n\right)\right) \right\}_{i+1/2}$$

3. Find the pressure needed to make the velocity field incompressible

$$\nabla_h \cdot \left(\frac{1}{\rho_h^{n+1}} \nabla_h p_h\right) = \frac{1}{\Delta t} \nabla_h \cdot \mathbf{u}_h^*$$

4. Correct the velocity by adding the pressure gradient:

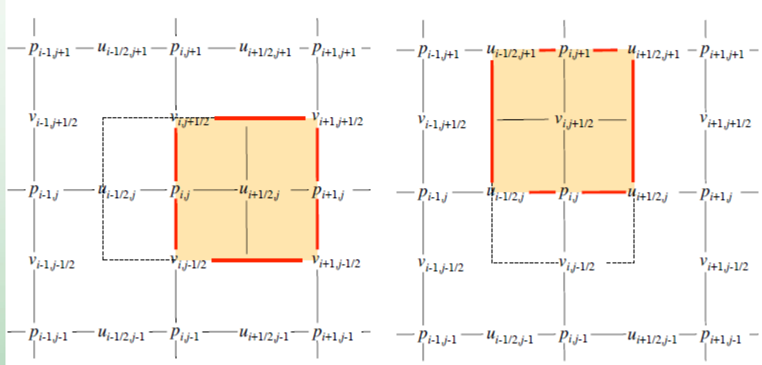
$$\mathbf{u}_h^{n+1} = \mathbf{u}_h^* - \Delta t \frac{\nabla_h p_h}{\rho_h^{n+1}}$$

Chap 4 **By E. Amani**

Pressure-velocity coupling

DNS of Multiphase Flows **by G. Tryggvason**

Define separate control volumes for each velocity component, shifted up for the vertical velocity and to the right for the horizontal velocity—Staggered Grid



u-velocity v-velocity

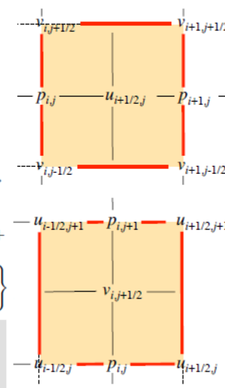
Chap 4 **By E. Amani**

Spatial discretization: Step 2

DNS of Multiphase Flows **by G. Tryggvason**

We now examine each termini in the equations in detail and derive a discrete approximation, assuming 2D flow:

$$\mathbf{u}_h^* = \frac{1}{\rho_h^{n+1}}\left(\rho_h^n \mathbf{u}_h^n + \Delta t\left(-\mathbf{A}_h^n + \rho_h^n \mathbf{g} + \mathbf{D}_h^n + \mathbf{f}_h^n\right)\right)$$



In component form:

$$u_{i+1/2,j}^* = \frac{1}{\frac{1}{2}(\rho_{i+1,j}^{n+1}+\rho_{i,j}^{n+1})}\Big\{\frac{1}{2}(\rho_{i+1,j}^n+\rho_{i,j}^n)u_{i+1/2,j}^n + \Delta t\Big(-(A_x)_{i+1/2,j}^n + \frac{1}{2}(\rho_{i+1,j}^n+\rho_{i,j}^n)(g_x)_{i+1/2,j}^n + (D_x)_{i+1/2,j}^n + (f_x)_{i+1/2,j}^n\Big)\Big\}$$

$$v_{i,j+1/2}^* = \frac{1}{\frac{1}{2}(\rho_{i,j+1}^{n+1}+\rho_{i,j}^{n+1})}\Big\{\frac{1}{2}(\rho_{i,j+1}^n+\rho_{i,j}^n)v_{i,j+1/2}^n + \Delta t\Big(-(A_y)_{i,j+1/2}^n + \frac{1}{2}(\rho_{i,j+1}^n+\rho_{i,j}^n)(g_y)_{i,j+1/2}^n + (D_y)_{i,j+1/2}^n + (f_y)_{i,j+1/2}^n\Big)\Big\}$$

Here we have used linear interpolation where quantities are not defined, such as for:

$$\rho_{i+1/2,j}^{n+1} = \frac{1}{2}(\rho_{i+1,j}^{n+1}+\rho_{i,j}^{n+1}) \quad \text{and} \quad \rho_{i,j+1/2}^{n+1} = \frac{1}{2}(\rho_{i,j+1}^{n+1}+\rho_{i,j}^{n+1})$$

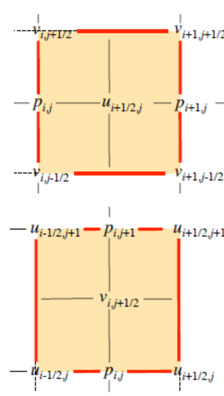
Chap 4 **By E. Amani**

Spatial discretization: Step 2

DNS of Multiphase Flows **by G. Tryggvason**

Discretization of the advection terms:

$$\mathbf{A}_h = \frac{1}{V}\oint_S \rho\mathbf{u}(\mathbf{u}\cdot\mathbf{n})ds$$



Approximate the integral by the midpoint rule

$$(A_x)_{i+1/2,j}^n = \frac{1}{V}\Big(\int_S \rho u(\mathbf{u}\cdot\mathbf{n})dS\Big)_{i+1/2,j}^n = \frac{1}{\Delta x\Delta y}\Big\{\big[(\rho uu)_{i+1,j}-(\rho uu)_{i,j}\big]\Delta y + \big[(\rho uv)_{i+1/2,j+1/2}-(\rho uv)_{i+1/2,j-1/2}\big]\Delta x\Big\}$$

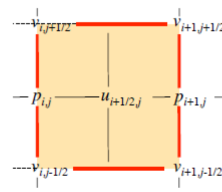
$$(A_y)_{i,j+1/2}^n = \frac{1}{V}\Big(\int_S \rho v(\mathbf{u}\cdot\mathbf{n})dS\Big)_{i,j+1/2}^n = \frac{1}{\Delta x\Delta y}\Big\{\big[(\rho uv)_{i+1/2,j+1/2}-(\rho uv)_{i-1/2,j+1/2}\big]\Delta y + \big[(\rho vv)_{i,j+1}-(\rho vv)_{i,j}\big]\Delta x\Big\}$$

Chap 4 **By E. Amani**

Spatial discretization: Step 2

DNS of Multiphase Flows **by G. Tryggvason**

Advection terms—Detailed discretization of the horizontal component using the mid point rule and averages for quantities not defined at the midpoint



$$(A_x)^n_{i+1/2,j} =$$
$$\frac{1}{\Delta x}\left[\rho_{i+1,j}\Big(\frac{u^n_{i+3/2,j}+u^n_{i+1/2,j}}{2}\Big)^2 - \rho_{i,j}\Big(\frac{u^n_{i+1/2,j}+u^n_{i-1/2,j}}{2}\Big)^2\right] +$$
$$\frac{1}{\Delta y}\left[\Big(\frac{\rho_{i,j}+\rho_{i+1,j}+\rho_{i,j+1}+\rho_{i+1,j+1}}{4}\Big)\Big(\frac{u^n_{i+1/2,j+1}+u^n_{i+1/2,j}}{2}\Big)\Big(\frac{v^n_{i+1,j+1/2}+v^n_{i,j+1/2}}{2}\Big)\right.$$
$$\left. -\Big(\frac{\rho_{i,j}+\rho_{i+1,j}+\rho_{i+1,j-1}+\rho_{i,j-1}}{4}\Big)\Big(\frac{u^n_{i+1/2,j}+u^n_{i+1/2,j-1}}{2}\Big)\Big(\frac{v^n_{i+1,j-1/2}+v^n_{i,j-1/2}}{2}\Big)\right]$$

Chap 4 **By E. Amani**

Spatial discretization: Step 2

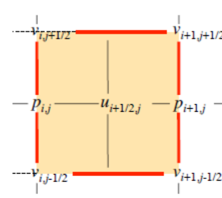
DNS of Multiphase Flows **by G. Tryggvason**

The diffusion term is:

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

In component form the rate of deformation tensor for two-dimensional flow is:

$$\mathbf{S} = \nabla\mathbf{u} + (\nabla\mathbf{u})^T = \begin{pmatrix} 2\frac{\partial u}{\partial x} & \frac{\partial u}{\partial y}+\frac{\partial v}{\partial x} \\ \frac{\partial u}{\partial y}+\frac{\partial v}{\partial x} & 2\frac{\partial v}{\partial y} \end{pmatrix}$$



For the horizontal velocity the integral is:

$$(D_x)^n_{i+1/2,j} = \frac{1}{V}\Big(\int_{\Delta y}(\mu S_{1,1}n_x)_{i+1}dy + \int_{\Delta y}\big(\mu S_{1,1}n_x\big)_i dy$$
$$+\int_{\Delta x}(\mu S_{1,2}n_y)_{j+1/2}dx + \int_{\Delta x}\big(\mu S_{1,2}n_y\big)_{j-1/2}dx\Big)$$

since

$$(n_x)_{i+1} = 1; \quad (n_x)_i = -1; \quad (n_y)_{j+1/2} = 1; \quad (n_y)_{j-1/2} = -1;$$

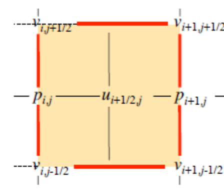
Chap 4 **By E. Amani**

Spatial discretization: Step 2

DNS of Multiphase Flows **by G. Tryggvason**

Using the midpoint rule for each component:

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



Substituting for the derivatives:

$$(D_x)^n_{i+1/2,j} = \frac{1}{\Delta x}\left\{2\mu_o\left(\frac{u^n_{i+3/2,j}-u^n_{i+1/2,j}}{\Delta x}\right) - 2\mu_o\left(\frac{u^n_{i+1/2,j}-u^n_{i-1/2,j}}{\Delta x}\right)\right\} + \frac{1}{\Delta y}\left\{\mu_o\left(\frac{u^n_{i+1/2,j+1}-u^n_{i+1/2,j}}{\Delta y} + \frac{v^n_{i+1,j+1/2}-v^n_{i,j+1/2}}{\Delta x}\right) - \mu_o\left(\frac{u^n_{i+1/2,j}-u^n_{i+1/2,j-1}}{\Delta y} + \frac{v^n_{i+1,j-1/2}-v^n_{i,j-1/2}}{\Delta x}\right)\right\}$$

Chap 4 **By E. Amani**

Spatial discretization: Step 2

DNS of Multiphase Flows **by G. Tryggvason**

Collecting the terms, the predicted velocity is:

$$u^*_{i+1/2,j} = \frac{1}{\frac{1}{2}(\rho^{n+1}_{i+1,j}+\rho^{n+1}_{i,j})}\left\{\frac{1}{2}(\rho^n_{i+1,j}+\rho^n_{i,j})u^n_{i+1/2,j} + \Delta t\left\{ -\frac{1}{\Delta x}\left[\rho_{i+1,j}\left(\frac{u^n_{i+3/2,j}+u^n_{i+1/2,j}}{2}\right)^2 - \rho_{i,j}\left(\frac{u^n_{i+1/2,j}+u^n_{i-1/2,j}}{2}\right)^2\right] - \frac{1}{\Delta y}\left[\left(\frac{\rho_{i,j}+\rho_{i+1,j}+\rho_{i,j+1}+\rho_{i+1,j+1}}{4}\right)\left(\frac{u^n_{i+1/2,j+1}+u^n_{i+1/2,j}}{2}\right)\left(\frac{v^n_{i+1,j+1/2}+v^n_{i,j+1/2}}{2}\right) - \left(\frac{\rho_{i,j}+\rho_{i+1,j}+\rho_{i+1,j-1}+\rho_{i,j-1}}{4}\right)\left(\frac{u^n_{i+1/2,j}+u^n_{i+1/2,j-1}}{2}\right)\left(\frac{v^n_{i+1,j-1/2}+v^n_{i,j-1/2}}{2}\right)\right] + \frac{1}{2}(\rho^n_{i+1,j}+\rho^n_{i,j})(g_x)^n_{i+1/2,j} + \mu_o\left(\frac{u^n_{i+3/2,j}-2u^n_{i+1/2,j}+u^n_{i-1/2,j}}{\Delta x^2} + \frac{u^n_{i+1/2,j+1}-2u^n_{i+1/2,j}+u^n_{i+1/2,j-1}}{\Delta y^2}\right) + (f_x)^n_{i+1/2,j}\right\}\right\}$$

Chap 4 **By E. Amani**

Coding: NSMFx.py

- results.py
  - Post-processing (Object-oriented programing sample)
- NSMFx.py
  - Solver (functional programing sample)
- VOFx.py
  - Library (functional programing sample)

Chap 4 By E. Amani

Coding: NSMF1.py

- Inputs:

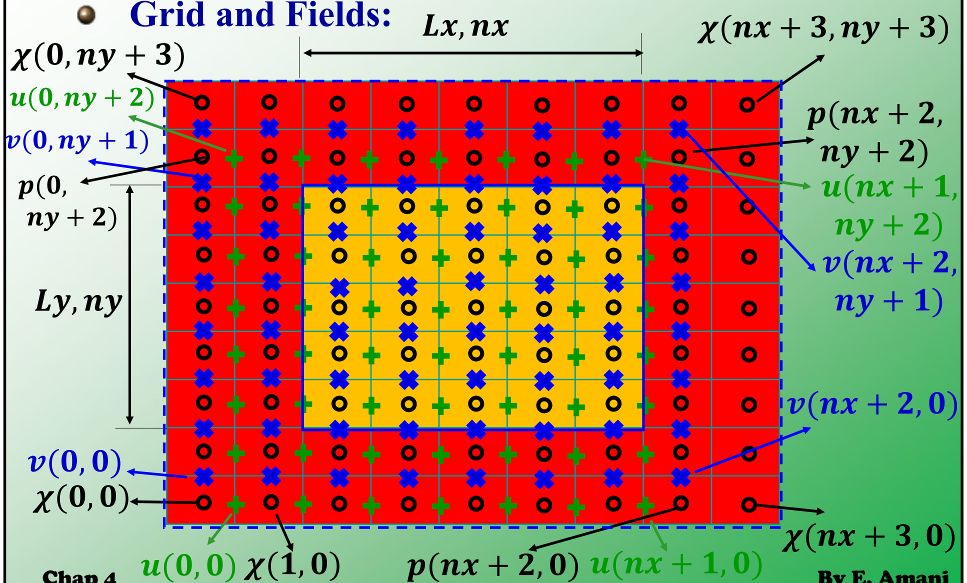
NSMF1.py × results.py ×

```
#===========================================================
# Code "NSMF1.py"
# A very simple Navier-Stokes solver for a drop falling in a
# rectangular box, using the conservative form of the equations.
# A first-order explicit projection method and centered in space
# discretizationa are used. The color function advection is
# simply discretized by a 2nd-order Central Difference (CD)
# scheme.
# The series of NSMF course codes were developed by
# "Ehsan Amani" in 2022
# by major modifications and amendments to
# the original Matlab code by "Gretar Tryggvason", which had been
# converted to Python by "Tingyi Lu" on 7/25/2018
#===========================================================

import numpy as np
import matplotlib.pyplot as plt
import time as time0
from results import results

#------------------- Physical variables --------------------
Lx=1.0;Ly=1.0;gx=0.0;gy=-10.0  # Domain size and
rho1=1.0; rho2=20.0; m0=0.01     # physical variables
unorth=0;usouth=0;veast=0;vwest=0;time=0.0
rad=0.15;xc=0.5;yc=0.7 # Initial drop size and location

#------------------- Numerical variables --------------------
nx=32;ny=32;dt=0.00125;nstep=300; maxit=200; maxError=0.001; beta=1.5
m0n=0 #0.01     # numerical dissipation
smallVal=1.e-16 #small value preventing division by zero
```

Chap 4 By E. Amani

Coding: NSMF1.py

- **Grid and Fields:**



Chap 4 By E. Amani

Coding: NSMF1.py

- **Grid and Fields:**
  - **arrays**

$p(0: nx+2, 0: ny+2)$ $\quad \chi, \rho(0: nx+3, 0: ny+3)$

$u(0: nx+1, 0: ny+2)$

$v(0: nx+2, 0: ny+1)$

NSMF1.py results.py

```python
#-------------------- Zero various arrys ----------------------
u=np.zeros((nx+2,ny+3));  v=np.zeros((nx+3,ny+2));  p=np.zeros((nx+3,ny+3))
ut=np.zeros((nx+2,ny+3)); vt=np.zeros((nx+3,ny+2)); tmp1=np.zeros((nx+3,ny+3))
uu=np.zeros((nx+2,ny+2)); vv=np.zeros((nx+2,ny+2)); tmp2=np.zeros((nx+3,ny+3))
npp=4; #npp=3
r=np.zeros((nx+npp,ny+npp)); chi=np.zeros((nx+npp,ny+npp))

dx=Lx/nx;dy=Ly/ny                           # Set the grid
x=np.linspace(-1.5*dx, (nx+0.5)*dx, nx+3)
y=np.linspace(-1.5*dy, (ny+0.5)*dy, ny+3)
xh=np.linspace(-dx, nx*dx, nx+2)
yh=np.linspace(-dy, ny*dy, ny+2)
```

Chap 4 By E. Amani

Coding: NSMF1.py

- **Initialization:**

```
NSMF1.py ×   results.py ×
#------------------- Initial Conditions ----------------------
r = np.zeros((nx+npp,nx+npp))+rho1 # Set density

for i in range(nx+3):
    for j in range(ny+3):                     # for the domain and the drop
        if((x[i]-xc)**2+(y[j]-yc)**2 <rad**2):
            r[i,j] = rho2
            chi[i,j] = 1.0
```

- **Time loop**

```
NSMF1.py ×   results.py ×
#-------------------- START TIME LOOP ----------------------
for istep in range(nstep):
    print(istep)

    #--- ADVECT marker using centered difference plus diffusion ----
    chio=chi.copy()

    if scalar_adv=='c':
        chi[2:nx+2,2:ny+2]=(chio[2:nx+2,2:ny+2]-(0.5*dt/dx)*(u[2:nx+2,2:ny+2]*(chio[3:nx+3,2:ny+2]
                  +chio[2:nx+2,2:ny+2]) - u[1:nx+1,2:ny+2]*(chio[1:nx+1,2:ny+2]+chio[2:nx+2,2:ny+2])
                  -(0.5* dt/dy)*(v[2:nx+2,2:ny+2]*(chio[2:nx+2,3:ny+3]
```

Chap 4 **By E. Amani**

Coding: NSMF1.py

- **Note: Fast computation with arrays**

```
for i in range(ni,ne):
    for j in range(mi,me):
        phi[i,j]=phi[i+1,j-1]-phi[i-1,j]
```

  - **vs.**

```
phi[ni:ne,mi:me]=phi[ni+1:ne+1,mi-1:me-1]-phi[ni-1:ne-1,mi:me]
```

Chap 4 **By E. Amani**

Coding: NSMF1.py

- Example (Step2): Velocity predictor

$$
\begin{aligned}
u^{*}_{i+1/2,j} = \frac{1}{\frac{1}{2}(\rho^{n+1}_{i+1,j} + \rho^{n+1}_{i,j})} \Bigg\{ & \frac{1}{2}(\rho^{n}_{i+1,j} + \rho^{n}_{i,j}) u^{n}_{i+1/2,j} + \Delta t \Bigg\{ \\
& - \frac{1}{\Delta x}\left[\rho_{i+1,j}\left(\frac{u^{n}_{i+3/2,j} + u^{n}_{i+1/2,j}}{2}\right)^2 - \rho_{i,j}\left(\frac{u^{n}_{i+1/2,j} + u^{n}_{i-1/2,j}}{2}\right)^2 \right] \\
& - \frac{1}{\Delta y}\Bigg[\left(\frac{\rho_{i,j} + \rho_{i+1,j} + \rho_{i,j+1} + \rho_{i+1,j+1}}{4}\right)\left(\frac{u^{n}_{i+1/2,j+1} + u^{n}_{i+1/2,j}}{2}\right)\left(\frac{v^{n}_{i+1,j+1/2} + v^{n}_{i,j+1/2}}{2}\right) \\
& - \left(\frac{\rho_{i,j} + \rho_{i+1,j} + \rho_{i+1,j-1} + \rho_{i,j-1}}{4}\right)\left(\frac{u^{n}_{i+1/2,j} + u^{n}_{i+1/2,j-1}}{2}\right)\left(\frac{v^{n}_{i+1,j-1/2} + v^{n}_{i,j-1/2}}{2}\right) \Bigg] + \frac{1}{2}(\rho^{n}_{i+1,j} + \rho^{n}_{i,j})(g_x)^{n}_{i+1/2,j} \\
& + \mu_o\left(\frac{u^{n}_{i+3/2,j} - 2u^{n}_{i+1/2,j} + u^{n}_{i-1/2,j}}{\Delta x^2} + \frac{u^{n}_{i+1/2,j+1} - 2u^{n}_{i+1/2,j} + u^{n}_{i+1/2,j-1}}{\Delta y^2} \right) + (f_x)^{n}_{i+1/2,j} \Bigg\} \Bigg\}
\end{aligned}
$$

NSMF1.py × results.py ×

```python

    #------------ Set tangential velocity at boundaries ----------
    u[:,1]=2*usouth-u[:,2];u[:,ny+2]=2*unorth-u[:,ny+1]
    v[1,:]=2*vwest-v[2,:];v[nx+2,:]=2*veast-v[nx+1,:]

    #------------ The predictor step ----------------------------
    ut[2:nx+1,2:ny+2]=((2.0/(r[3:nx+2,2:ny+2]+r[2:nx+1,2:ny+2]))*(0.5*
                        (ro[3:nx+2,2:ny+2]+ro[2:nx+1,2:ny+2])*u[2:nx+1,2:ny+2]+ dt* (
               -(0.25/dx)*(ro[3:nx+2,2:ny+2]*(u[3:nx+2,2:ny+2]+u[2:nx+1,2:ny+2])**2-
                         ro[2:nx+1,2:ny+2]*(u[2:nx+1,2:ny+2]+u[1:nx,2:ny+2])**2)
           -(0.0625/dy)*( (ro[2:nx+1,2:ny+2]+ro[3:nx+2,2:ny+2]+ro[2:nx+1,3:ny+3]+ro[3:nx+2,3:ny+3])*
                         (u[2:nx+1,3:ny+3]+u[2:nx+1,2:ny+2])*(v[3:nx+2,2:ny+2]+v[2:nx+1,2:ny+2])
                       -(ro[2:nx+1,2:ny+2]+ro[3:nx+2,2:ny+2]+ro[3:nx+2,1:ny+1]+ro[2:nx+1,1:ny+1])*(
                         +u[2:nx+1,1:ny+1])*(v[3:nx+2,1:ny+1]+v[2:nx+1,1:ny+1]))
                  +m0*((u[3:nx+2,2:ny+2]-2*u[2:nx+1,2:ny+2]+u[1:nx,2:ny+2])/dx**2+
                         (u[2:nx+1,3:ny+3]-2*u[2:nx+1,2:ny+2]+u[2:nx+1,1:ny+1])/dy**2)
                + 0.5*(ro[3:nx+2,2:ny+2]+ro[2:nx+1,2:ny+2])*gx ) ))
```

Chap 4 By E. Amani

CFD overview

- This was a short review on single-phase flow CFD
- Take a look at your previous CFD courses
- See video "chap4-CFDOverview"

Chap 4 By E. Amani