Описание функции `computeResidual_7`: вычисление невязки условий на межфазной границе

Введение

Функция `computeResidual_7` предназначена для вычисления невязки условий на межфазной границе в двумерной задаче течения жидкости и пара. Она основана на дискретной форме уравнения баланса импульса на интерфейсе:

$$\underbrace{(\rho u_n)_v}_{\text{массовый поток}} \cdot \underbrace{(u_{n,v} - u_{n,l})}_{\text{скачок скорости}} + \underbrace{(P_v - P_l)}_{\text{скачок давления}} - \underbrace{(\tau^v_{nn} - \tau^l_{nn})}_{\text{скачок нормальных напряжений}} - \underbrace{2\sigma H}_{\text{поверхностное натяжение}} = 0. \tag{1}$$

Сигнатура функции и входные параметры

Функция вызывается следующим образом:

```
R = computeResidual_7(fields, params, phase, H, dx, dy)
```

где:

- `fields` содержит векторы скоростей u_x, u_y и поле давления P:

```
ux = fields.u_x;
uy = fields.u_y;
P  = fields.P;
```

- `params` содержит физические параметры двух фаз:

```
rho = params.rho_liquid * (phase == 0) + params.rho_vapor *
mu  = params.mu_liquid  * (phase == 0) + params.mu_vapor
  * (phase == 1);
```

- `phase` — булево поле фаз: 0 — жидкость, 1 — пар.
- `H` — матрица кривизны интерфейса.
- `dx`, `dy` — размеры ячеек по x и y.

1. Массовый поток: $(\rho u_n)_v$

Находится в следующих строках:

```
u1n = ux1 * ni + uy1 * nj;
u2n = ux2 * ni + uy2 * nj;

tmp(gas_left) = rho1(gas_left) .* u1n(gas_left);
Jv(idx1) = Jv(idx1) + tmp;

tmp(gas_right) = rho2(gas_right) .* u2n(gas_right);
Jv(idx1) = Jv(idx1) + tmp;
```

Здесь $u_n = \vec{u} \cdot \vec{n}$, а `rho1`, `rho2` соответствуют плотностям в соседних ячейках. Значение `Jv` накапливает величину $(\rho u_n)_v$.

2. Скачок скорости: $(u_{n,v} - u_{n,l})$

Рассчитывается как разность проекций скорости на нормаль в ячейках газа и жидкости:

```
tmp(gas_left) = u1n(gas_left) - u2n(gas_left);
delta_u(idx1) = delta_u(idx1) + tmp;

tmp(gas_right) = u2n(gas_right) - u1n(gas_right);
delta_u(idx1) = delta_u(idx1) + tmp;
```

`delta_u` содержит разность нормальных компонент скорости между газовой и жидкой фазой.

3. Скачок давления: $(P_v - P_l)$

```
tmp(gas_left) = P1(gas_left) - P2(gas_left);
delta_P(idx1) = delta_P(idx1) + tmp;

tmp(gas_right) = P2(gas_right) - P1(gas_right);
delta_P(idx1) = delta_P(idx1) + tmp;
```

Разность давления между паром и жидкостью сохраняется в `delta_P`.

4. Скачок нормальных напряжений: $(\tau^v_{nn} - \tau^l_{nn})$

Сначала вычисляется вязкий тензор, затем проецируется на нормаль:

```
[dudx, dudy] = gradient(ux, dx, dy);
[dvdx, dvdy] = gradient(uy, dx, dy);
div_u = dudx + dvdy;

tau1 = mu1 .* ((4/3) * div1 * ni^2) + 2 * mu1 .* ...
viscous_proj(dudx(idx1), dudy(idx1), dvdx(idx1), dvdy(idx1), ni, nj);
tau2 = mu2 .* ((4/3) * div2 * ni^2) + 2 * mu2 .* ...
viscous_proj(dudx(idx2), dudy(idx2), dvdx(idx2), dvdy(idx2), ni, nj);

tmp(gas_left) = tau1(gas_left) - tau2(gas_left);
delta_tau(idx1) = delta_tau(idx1) + tmp;

tmp(gas_right) = tau2(gas_right) - tau1(gas_right);
delta_tau(idx1) = delta_tau(idx1) + tmp;
```

Функция `viscous_proj` реализует проекцию вязкого тензора на нормаль:

```
function T = viscous_proj(dudx, dudy, dvdx, dvdy, ni, nj)
T = ni.^2 .* dudx + nj.^2 .* dvdy + ni .* nj .* (dudy + dvdx);
end
```

Соответствует формуле:

$$\tau_{nn} = \mu \left[\left(\frac{4}{3} \boldsymbol{\nabla} \cdot \vec{u} \right) n_i^2 + 2 \left(n_x^2 \frac{\partial u_x}{\partial x} + n_y^2 \frac{\partial u_y}{\partial y} + n_x n_y \left(\frac{\partial u_x}{\partial y} + \frac{\partial u_y}{\partial x} \right) \right) \right]$$

5. Поверхностное натяжение: $2\sigma H$

Добавляется к невязке после суммирования направлений:

```
R = R - 2 * params.sigma * H;
```

6. Общая сборка невязки

Вклад от каждого направления (всего 4 — вверх, вниз, влево, вправо) добавляется в итоговую невязку:

```
R = R + (Jv .* delta_u + delta_P - delta_tau);
```

Эта строка и реализует основную формулу:

$$R = (\rho u_n)_v (u_{n,v} - u_{n,l}) + (P_v - P_l) - (\tau^v_{nn} - \tau^l_{nn})$$

Суммирование происходит по всем ячейкам, содержащим интерфейс (фазный переход между 0 и 1), с учётом направления нормали.

Заключение

- $(\rho u_n)_v$ — через `Jv`
- $(u_{n,v} - u_{n,l})$ — через `delta_u`
- $(P_v - P_l)$ — через `delta_P`
- $(\tau^v_{nn} - \tau^l_{nn})$ — через `delta_tau`
- $2\sigma H$ — добавляется в конце