Описание функции `computeResidual_8`: баланс касательной компоненты на межфазной границе

Введение

Функция `computeResidual_8` реализует дискретную форму касательного условия на межфазной границе в двумерной задаче течения. В отличие от нормального баланса импульса, описанного в `computeResidual_7`, здесь рассматривается касательная составляющая уравнения Навье–Стокса, проецированная на интерфейс:

$$(\rho u_n)_v (u_{t,v} - u_{t,l}) - (\tau^v_{n\tau} - \tau^l_{n\tau}) = 0, \tag{1}$$

где:

- $u_n = \vec{u} \cdot \vec{n}$ — нормальная компонента скорости,
- $u_t = \vec{u} \cdot \vec{\tau}$ — касательная компонента скорости,
- $\tau_{n\tau}$ — проекция вязкого тензора на направление $\vec{n} \cdot \vec{\tau}$,
- индекс v — пар, l — жидкость.

Цель — вычислить невязку этого выражения в каждой ячейке, содержащей интерфейс, и сохранить её в массиве R.

Сигнатура функции и входные параметры

```
R = computeResidual_8(fields, params, phase, dx, dy)
```

- `fields` содержит компоненты скорости:

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

- `params` содержит физические параметры:

```
rho_liq = params.rho_liquid;
rho_vap = params.rho_vapor;
mu_liq  = params.mu_liquid;
mu_vap  = params.mu_vapor;
```

- `phase` — фазовое поле: 0 — жидкость, 1 — пар.
- `dx`, `dy` — размеры ячеек по координатам.

1. Построение нормалей и касательных

Для интерфейсных ячеек определяется нормальный вектор $\vec{n} = (n_x, n_y)$ и касательный вектор $\vec{\tau} = (-n_y, n_x)$:

```
ni = nx(idx1);   % n_x
nj = ny(idx1);   % n_y

ti = -nj;        % tau_x
tj = ni;         % tau_y
```

2. Нормальная компонента скорости и массовый поток

Скорости в двух ячейках по обе стороны от интерфейса проецируются на нормаль:

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

Для вычисления $(\rho u_n)_v$ используется плотность в соответствующих ячейках:

```
rho1 = rho_liq * (phase1 == 0) + rho_vap * (phase1 == 1);
rho2 = rho_liq * (phase2 == 0) + rho_vap * (phase2 == 1);

Jv = zeros(size(idx1));
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;
```

Величина `Jv` содержит $(\rho u_n)_v$.

3. Касательная компонента скорости

Аналогично вычисляется u_t — проекция скорости на касательное направление:

```
u1t = ux1 .* ti + uy1 .* tj;
u2t = ux2 .* ti + uy2 .* tj;

delta_u = zeros(size(idx1));

tmp(gas_left) = u1t(gas_left) - u2t(gas_left);
delta_u(idx1) = delta_u(idx1) + tmp;

tmp(gas_right) = u2t(gas_right) - u1t(gas_right);
delta_u(idx1) = delta_u(idx1) + tmp;
```

`delta_u` содержит $(u_{t,v} - u_{t,l})$.

4. Вязкие напряжения: $\tau_{n\tau}$

Сначала вычисляются градиенты компонент скорости:

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

Далее — проекция тензора напряжений на направление $\vec{n} \cdot \vec{\tau}$:

```
mu1 = mu_liq * (phase1 == 0) + mu_vap * (phase1 == 1);
mu2 = mu_liq * (phase2 == 0) + mu_vap * (phase2 == 1);

tau1 = mu1 .* ((dudx1 .* ni + dudy1 .* nj) .* ti + ...
(dvdx1 .* ni + dvdy1 .* nj) .* tj);
tau2 = mu2 .* ((dudx2 .* ni + dudy2 .* nj) .* ti + ...
(dvdx2 .* ni + dvdy2 .* nj) .* tj);
```

Затем — вычисление скачка между фазами:

```
delta_tau = zeros(size(idx1));

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;
```

5. Сборка полной невязки

Итоговое выражение формируется как:

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

```
R = Jv .* delta_u - delta_tau;
```

Невязка `R` вычисляется в ячейках, где присутствует фазный переход ($0 \to 1$ или $1 \to 0$), с учётом направления нормали.

Заключение

Функция `computeResidual_8` реализует касательный баланс импульса на межфазной границе. Структура её работы:

- $(\rho u_n)_v$ — через переменную `Jv`,
- $(u_{t,v} - u_{t,l})$ — через `delta_u`,
- $(\tau^v_{n\tau} - \tau^l_{n\tau})$ — через `delta_tau`,
- Общая сборка — через `R = Jv .* delta_u - delta_tau`.

Эта функция используется совместно с `computeResidual_7` для обеспечения полного баланса импульса на межфазной границе — как по нормали, так и по касательной.