Описание функции `computeResidual_9`

Назначение функции

Функция `computeResidual_9` рассчитывает невязку баланса энергии на ячейках, расположенных у границы фаз «жидкость-пар». В невязку входят:

- конвективный поток полной энергии;
- вязкий диссипативный вклад;
- скачок теплового потока (получается через `computeHeatFluxJump`).

Формально:

$$R = \left[\rho\vec{u}\cdot\vec{n}\left(h+\frac{1}{2}|\vec{u}|^2\right) - \vec{u}\cdot\boldsymbol{\tau}\cdot\vec{n}\right]^{\text{vap}} - \left[\rho\vec{u}\cdot\vec{n}\left(h+\frac{1}{2}|\vec{u}|^2\right) - \vec{u}\cdot\boldsymbol{\tau}\cdot\vec{n}\right]^{\text{liq}} + \left(q_{\text{vap}}^{(n)} - q_{\text{liq}}^{(n)}\right)$$

1. Получение параметров и полей

На основе фазового поля задаются свойства: плотность, вязкость и энтальпия.

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

2. Градиенты скоростей и температурный поток

Сначала рассчитываются градиенты полей скорости и теплового потока через вспомогательную функцию:

```
[dudx, dudy] = gradient(ux, dx, dy);
[dvdx, dvdy] = gradient(uy, dx, dy);
q_jump = computeHeatFluxJump(fields, params, phase, dx, dy);
```

3. Обход интерфейсных ячеек

Проход по 4 соседним направлениям: влево, вверх, вправо, вниз. Для каждой пары ячеек определяются:

- индексы соседей;
- принадлежность фазе (газ/жидкость);
- нормальный вектор к интерфейсу.

```
shifts  = {[-1, 0], [0, -1], [1, 0], [0, 1]};
normals = {[0, -1], [-1, 0], [0, 1], [1, 0]};
...
is_interface = p1 ~= p2;
gas_left  = is_interface & is_gas1 & ~is_gas2;
gas_right = is_interface & ~is_gas1 & is_gas2;
```

4. Вычисление потоков энергии

Для каждой пары ячеек вычисляется:

- скалярное произведение скорости и нормали;
- полная энергия: $h + \frac{1}{2}|\vec{u}|^2$;
- проекция вязкого тензора: $\vec{u} \cdot \boldsymbol{\tau} \cdot \vec{n}$.

```
u1n = ux1 * ni + uy1 * nj;
tau1 = mu1 .* viscous_proj(...);
utau1 = ux1 .* ni + uy1 .* nj;
```

5. Формирование локальной невязки

Если газ находится слева, то значение невязки берётся из ячейки `idx1`. Если справа — из `idx2`. Разность энергий и потоков складывается в вектор `Rloc`.

```
Rloc(gas_left) = ...
rho1 .* u1n .* (h1 - h2 + 0.5 * (u1sq - u2sq)) ...
- utau1 .* tau1 + utau2 .* tau2;
```

6. Агрегация невязки и добавление теплового потока

Результирующее значение записывается в массив R. В конце добавляется скачок теплового потока:

```
R(idx1) = R(idx1) + Rloc;
R = R + q_jump;
```

7. Вспомогательная функция: `viscous_proj`

Реализует проекцию симметричной части тензора скорости на нормаль:

$$\tau_{nk} u_k = n_i \tau_{ik} u_k = n_x^2 \frac{\partial u_x}{\partial x} + n_y^2 \frac{\partial u_y}{\partial y} + n_x n_y \left(\frac{\partial u_x}{\partial y} + \frac{\partial u_y}{\partial x} \right)$$

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