ПАРАЛЛЕЛЬНАЯ РЕАЛИЗАЦИЯ ЯВНОЙ РАЗНОСТНОЙ СХЕМЫ ДЛЯ РАСЧЕТА ДВУХФАЗНЫХ ТЕЧЕНИЙ В КАНАЛАХ

Л.Л. Миньков, К.В. Озерская

Томский государственный университет

Рассматривается течение многофазной среды в плоском канале постоянного сечения, с закрытым торцом (рис. 1), с боковой поверхности которого осуществляется приход смеси газа и частиц с известной функцией распределения по размеру.

Требуется найти стационарное распределение параметров двухфазной смеси по каналу. Для решения таких задач используется метод установления. Предполагается, что решение соответствующей нестационарной задачи сходится к стационарному, если невязка решения падает на несколько порядков.

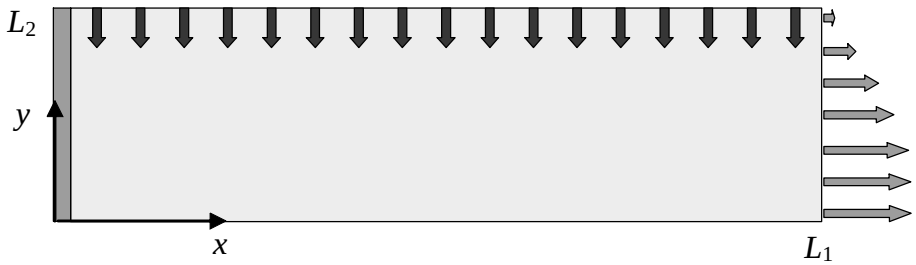

Рис. 1. Область решения

Определяющая система уравнений

Система уравнений, описывающая течение смеси газа и полидисперсных частиц в рамках многоскоростного и многотемпературного континуума [1], имеет вид

$$\oint_S \mathbf{U}dxdy + \mathbf{F}dtdy + \mathbf{G}dtdx = \iint_D \mathbf{E}dtdxdy, \qquad (1)$$

$$0 < x < L_1,\ 0 < y < L_2,\ t > 0,$$

$$\mathbf{U}=\begin{bmatrix}\rho\\ \rho U\\ \rho V\\ \rho E\\ \rho_s\\ \rho_s U_s\\ \rho_s V_s\\ \rho_s T_s\end{bmatrix};\quad \mathbf{F}=\begin{bmatrix}\rho U\\ \rho U^2+P\\ \rho UV\\ \rho UH\\ \rho_s U_s\\ \rho_s U_s^2\\ \rho_s V_s U_s\\ \rho_s T_s U_s\end{bmatrix};\quad \mathbf{G}=\begin{bmatrix}\rho V\\ \rho UV\\ \rho V^2+P\\ \rho VH\\ \rho_s V_s\\ \rho_s V_s U_s\\ \rho_s V_s^2\\ \rho_s T_s V_s\end{bmatrix};\quad \mathbf{E}=\begin{bmatrix}0\\ -\sum_{s=1}^{N}\rho_s f_{x,s}\\ -\sum_{s=1}^{N}\rho_s f_{y,s}\\ -\sum_{s=1}^{N}\rho_s\left(\vec{V}_s\vec{f}_s+q_s\right)\\ 0\\ \rho_s f_{x,s}\\ \rho_s f_{y,s}\\ \rho_s q_s/C_B\end{bmatrix},$$

где $\vec{f}_s=\varphi_{1,s}(\vec{V}-\vec{V}_s)$ и $q_s=\varphi_{2,s}(T-T_s)$ – соответственно, сила и тепловой поток, действующие на частицу со стороны газа; ρ – плотность; U, V – составляющие вектора скорости вдоль x и y осей; P – давление; E – полная энергия, $E=e+u^2/2$; e – внутренняя энергия газа, $e=P/(k-1)\rho$; H – энтальпия, $H=E+P/\rho$; $\varphi_{1,s}$ и $\varphi_{2,s}$ – обратные времена динамической и тепловой релаксации, соответственно; N – количество фракций, на которые разбита вся совокупность частиц. Индекс s относится к частицам.

Система уравнений (1) замыкается уравнением состояния $P=\rho RT$.

Начальные условия

Для газа

$$\rho=\rho_н;\quad U=0;\ V=0;\ P=p_н.$$

Для частиц

$$\rho_s=0;\quad U_s=0;\ V_s=0;\ T_s=0.$$

Граничные условия для газа

Поверхность горения $y = L_2$:

$$\rho V=(1-z)m_T P^{\nu};\ U=0;\ \frac{k}{k-1}\frac{P}{\rho}+\frac{V^2}{2}=H_T.$$

Плоскость симметрии $y = 0$:

$$\frac{\partial P}{\partial y}=\frac{\partial \rho}{\partial y}=\frac{\partial U}{\partial y}=V=0.$$

Торцевая поверхность $x = 0$:

$$U=0.$$

Поверхность истечения $x = L_1$:

$$P=P_H.$$

Граничные условия для частиц

Поверхность горения $y = L_2$:

$$\rho_s = \frac{z}{1-z}\rho f_s;\ U_s = U;\ V_s = V;\ T_s = T\,.$$

Плоскость симметрии $y = 0$:

$$\left.\frac{\partial T_s}{\partial y}\right|_{y=+0} = \left.\frac{\partial T_s}{\partial y}\right|_{y=-0};\ \left.\frac{\partial \rho_s}{\partial y}\right|_{y=+0} = \left.\frac{\partial \rho_s}{\partial y}\right|_{y=-0};\ \left.\frac{\partial U_s}{\partial y}\right|_{y=+0} = \left.\frac{\partial U_s}{\partial y}\right|_{y=-0};\ V_s = 0\,.$$

Торцевая поверхность $x = 0$:

$$U_s = 0\,.$$

Поверхность истечения $x = L_1$: условия не задаются.

Здесь m_T, ν – константы закона скорости горения топлива (боковой поверхности стенки канала); z – массовая доля частиц; H_T – энтальпия топлива; f_s – функция распределения частиц по размерам.

Разностная схема

Для решения системы уравнений (1) использовался метод конечных объемов. Для этого исходная область решения разбивалась прямоугольной разностной сеткой на ячейки, для каждой из которых система уравнений (1) после интегрирования принимала вид

$$\frac{\mathbf{U}_{i,j}^{n+1} - \mathbf{U}_{i,j}^{n}}{\tau} + \frac{\mathbf{F}_{i+1/2,j} - \mathbf{F}_{i-1/2,j}}{h_x} + \frac{\mathbf{G}_{i,j+1/2} - \mathbf{G}_{i,j-1/2}}{h_y} = \mathbf{E}^{n+1}. \qquad (2)$$

Здесь потоки $\boldsymbol{F}$ и $\boldsymbol{G}$ для газа на гранях ячеек определялись по методу Ван-Лира [2], а для частиц – по методу А.Н. Крайко [3]. Целые индексы i и j относятся к центру ячейки, а полуцелые – к граням ячеек.

Если обозначить в системе разностных уравнений (2) конвективные комплексы

$$\frac{\mathbf{F}_{i+1/2,j} - \mathbf{F}_{i-1/2,j}}{h_x} + \frac{\mathbf{G}_{i,j+1/2} - \mathbf{G}_{i,j-1/2}}{h_y}$$

для уравнений импульса и энергии, соответственно, через A_i, B_i, C_i для газа и через A_{si}, B_{si}, C_{ss} для частиц, то система (2) может быть разрешена относительно параметров на n+1 временном слое следующим образом [4]:

$$U_i^{n+1} = \frac{A + \sum_{s=1}^{N} A_{si}\dfrac{\varphi_{1s}\tau}{1+\varphi_{1s}\tau}}{\rho_i^{n+1} + \sum_{s=1}^{N} \rho_{si}^{n+1}\dfrac{\varphi_{1s}\tau}{1+\varphi_{1s}\tau}};\qquad V_i^{n+1} = \frac{B_i + \sum_{s=1}^{N} B_{si}\dfrac{\varphi_{1s}\tau}{1+\varphi_{1s}\tau}}{\rho_i^{n+1} + \sum_{s=1}^{N} \rho_{si}^{n+1}\dfrac{\varphi_{1s}\tau}{1+\varphi_{1s}\tau}};$$

$$U_{si}^{n+1} = \frac{A_{si}}{\rho_{si}^{n+1}(1+\varphi_{1s}\tau)} + \frac{\varphi_{1s}\tau}{1+\varphi_{1s}\tau}U_i^{n+1};\ V_{si}^{n+1} = \frac{B_{si}}{\rho_{si}^{n+1}(1+\varphi_{1s}\tau)} + \frac{\varphi_{1s}\tau}{1+\varphi_{1s}\tau}V_i^{n+1}; \qquad (3)$$

$$T_i^{n+1} = \frac{C_i' + \sum_{s=1}^{N} C_{si} c_B \frac{\varphi_{2s}\tau}{1+\varphi_{2s}\tau}}{\rho_i^{n+1} c_V + \sum_{s=1}^{N} \rho_{si}^{n+1} c_B \frac{\varphi_{2s}\tau}{1+\varphi_{2s}\tau}}; \qquad T_{si}^{n+1} = \frac{C_{si}}{\rho_{si}^{n+1}(1+\varphi_{2s}\tau)} + \frac{\varphi_{2s}\tau}{1+\varphi_{2s}\tau} T_i^{n+1},$$

где $C_i' = C_i - \rho_i^{n+1} \frac{U_i^{n+12} + V_i^{n+12}}{2} - \tau \rho_{si}^{n+1} \varphi_1 \vec{V}_{si}^{n+1} (\vec{V}_i^{n+1} - \vec{V}_{si}^{n+1})$.

Данная схема является устойчивой для любых размеров частиц (обратных времен релаксации φ_s).

Невязка решения на каждом шаге по времени определяется по правилу:

$$d = \frac{1}{N \times M} \sum_{i=1}^{N} \sum_{j=1}^{M} \left(\left| \rho_{ij}^{n+1} - \rho_{ij}^{n} \right| + \left| U_{ij}^{n+1} - U_{ij}^{n} \right| + \left| V_{ij}^{n+1} - V_{ij}^{n} \right| + \left| P_{ij}^{n+1} - P_{ij}^{n} \right| \right).$$

Параллельный алгоритм решения задачи

Для распараллеливания полученного последовательного алгоритма (2)–(3) используется геометрическая декомпозиция, т.е. каждый процесс, получая исходные данные, находит решение поставленной задачи в отведенной ему геометрической подобласти. Рассматриваемая область решения проецируется на процессы, организованные в соответствии с декартовой топологией.

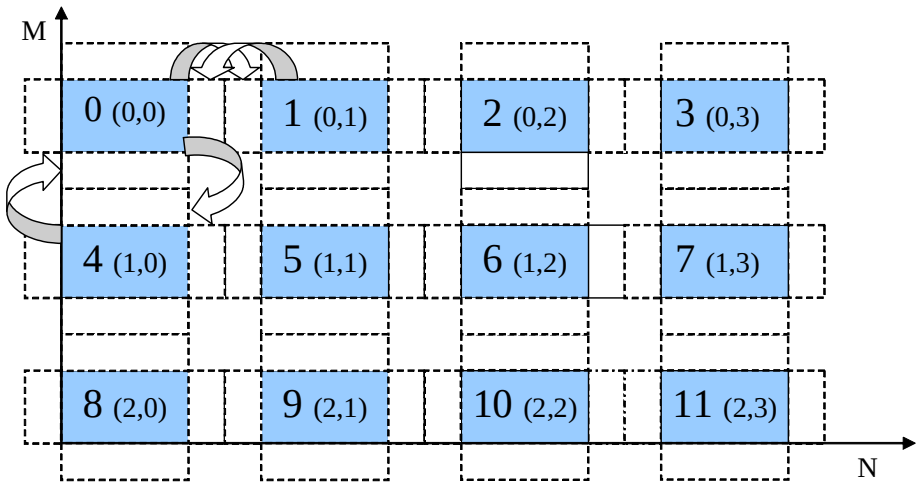

Рис. 2. Схема обмена данными между процессами в случае декартовой топологии

Для организации обмена информацией между процессами на каждом процессе вводятся четыре дополнительных граничных массива, в которые принимаются данные, посланные от соседних процессов. Эти данные необходимы для нахождения значений потоков на границах подобластей процессов. Обмен данными между процессами происходит по схеме, показанной на рис. 2.

Организация процессов в матрицу в случае декартовой топологии

Переменной SIZE присваиваем значение, равное количеству задействованных процессов, объединенных одним общим коммутатором icom, совпадающим с MPI_COMM_WORLD:

```
CALL MPI_COMM_SIZE(icom, SIZE, ierror)
```

Определяем номер текущего процесса RANK среди процессов, объединенных общим коммутатором icom:

```
CALL MPI_COMM_RANK(icom, RANK, ierror)
```

Задаем число процессов вдоль продольного направления:

```
ndims(1)=NX
```

Задаем число процессов вдоль поперечного направления:

```
ndims(2)=NY
```

Условие непериодичности границ области:

```
periods(1)=.false.,    periods(2)=.false.
```

Разрешаем производить перенумерацию процессов:

```
reorder=.true.
```

Устанавливаем новый коммутатор icom_2d, объединяющий процессы в матрицу:

```
CALL MPI_CART_CREATE(icom,2,ndims,periods,reorder,
                     icom_2d,ierr)
```

Получаем координаты id_coord текущего процесса в только что организованной матрице

```
CALL MPI_CART_COORDS(icom_2d,RANK,2,id_coord,ierr)
imap=id_coord(1)
jmap=id_coord(2)
```

Определяем номер процесса ill, находящегося слева от текущего процесса в координатах декартовой сетки

```
IF (imap.EQ.0) THEN
 ill=MPI_PROC_NULL
ELSE
 id_left(1)=imap-1
 id_left(2)=jmap
 CALL MPI_CART_RANK(icom_2d,id_left,ill,ierr)
ENDIF
```

Определяем номер процесса irr, находящегося справа от текущего процесса в координатах декартовой сетки

```
IF(imap.EQ.NX-1) THEN
 irr=MPI_PROC_NULL
ELSE
```

```
 id_right(1)=imap+1
 id_right(2)=jmap
 CALL MPI_CART_RANK(icom_2d,id_right,irr,ierr)
ENDIF
```

Определяем номер процесса ***ibb***, находящегося снизу от текущего процесса в координатах декартовой сетки

```
IF (jmap.EQ.0) THEN
 ibb=MPI_PROC_NULL
ELSE
 id_bottom(1)=imap
 id_bottom(2)=jmap-1
 CALL MPI_CART_RANK(icom_2d,id_bottom,ibb,ierr)
ENDIF
```

Определяем номер процесса ***itt***, находящегося снизу от текущего процесса в координатах декартовой сетки

```
IF(jmap.EQ.NY-1) THEN
 itt=MPI_PROC_NULL
ELSE
 id_top(1)=imap
 id_top(2)=jmap+1
 CALL MPI_CART_RANK(icom_2d,id_top,itt,ierr)
ENDIF
```

Обмен данными между процессами на одном шаге по времени

1) Заполняем буферный массив `S_right`, который будет послан «вправо» на процессор с номером `irr`.

2) Заполняем буферный массив `S_left`, который будет послан «влево» на процессор с номером `ill`.

3) Посылаем буферный массив `S_left` на процесс `ill`, который там будет записан в массив `R_left`:

```
CALL MPI_SENDRECV(S_left,Nbuf,ir8,ill,1,R_left,
        Nbuf,ir8,ill,1,icom,IREG(1),ierr1).
```

4) Посылаем буферный массив `S_right` на процесс `irr`, который там будет записан в массив `R_right`:

```
CALL MPI_SENDRECV(S_right,Nbuf,ir8,irr,1, R_right,
      Nbuf,ir8,irr,1,icom,IREG(2),ierr2).
```

5) Заполняем буферный массив `S_top`, который будет послан «вверх» на процессор с номером `itt`.

6) Заполняем буферный массив S_bottom, который будет послан «вниз» на процессор с номером ibb.

7) Посылаем буферный массив S_top на процесс itt, который там будет записан в массив R_top:

```
CALL MPI_SENDRECV(S_top,Nbuf,ir8,itt,2,R_top,
     Nbuf,ir8,itt,2,icom,IREG(3),ierr3).
```

8) Посылаем буферный массив S_bottom на процесс ibb, который там будет записан в массив R_bottom:

```
CALL MPI_SENDRECV(S_bottom,Nbuf,ir8,ibb,2,R_bottom,
     Nbuf,ir8,ibb,2,icom,IREG(4),ierr4).
```

9) Рассчитываем параметры на гранях ячеек.

10) Определяем минимальное значение шага по времени на каждом процессе, находим минимальный шаг по времени среди шагов, полученных от всех процессов, и рассылаем его на все процессы

```
CALL MPI_ALLREDUCE(TA_MIN,TA,1,ir8,MPI_MIN,
                   icom,ierror).
```

11) Рассчитываем параметры в центрах ячеек.

12) Определяем невязку на каждом процессе, находим сумму невязок со всех процессов, делим ее на число процессов size и рассылаем на все процессы

```
CALL MPI_ALLREDUCE(DD1,DD,1,ir8,MPI_SUM,icom,ierror)
DD=DD/SIZE.
```

Обсуждение результатов

Полученный параллельный алгоритм решения системы уравнений (1) был протестирован на вычислительном кластере ТГУ, состоящем из девяти двухпроцессорных узлов.

Расчеты проводились на разностной сетке размером 144×144, количество фракций принималось равным 10 и 50. Для вычисления ускорения полученного алгоритма определялось время выполнения 100 шагов по временной координате. При этом проводилось сравнение матричной и линейной декартовых топологий следующих размерностей: 2×2, 4×4 и 1×4, 1×16. В табл. 1 показано время работы программы в секундах и увеличение ускорения алгоритма, при работе программы на 8 узлах. Видно, что увеличение числа процессов в четыре раза приводит к увеличению ускорения, близкому трем в случае линейной организации процессов, и равному 3.5 – в случае матричной организации процессов. При этом скорость выполнения программы для матричной организации процессов оказалась выше, чем для линейной. Так, на 4 процессах это превышение составило 6% , а на 16 процессах – 33 %. Из результатов расчета также следует, что увеличе-

ние времени решения из-за увеличения числа фракций слабо зависит от способа организации процессов и определяется, в основном, их количеством. Пятикратное увеличение числа процессов привело к увеличению времени расчета в 5,1 раза на 4 процессах и в 5,5 раза на 16 процессах.

Таблица 1

Время выполнения программы на 8 узлах

10 фракций			50 фракций		
Число процессов	Линейка	Матрица	Число процессов	Линейка	Матрица
4	38,05	36,52	4	195,48	183,76
16	12,89	9,74	16	71,57	53,52
Увеличение ускорения	2,95	3,75	Увеличение ускорения	2,73	3,43

Поскольку загрузка узлов кластера неравномерная, ввиду коллективного его использования, то затруднительно получить адекватное время решения задачи на одном процессе. Так, на момент тестирования алгоритма с одним процессом для 50 фракций время работы программы на каждом из узлов (в секундах) было следующим: 788; 478; 268; 257; 241; 335; 241; 241.

Таблица 2

Время выполнения программы на 1 узле

10 фракций		
Число процессов	Линейка	Матрица
4	30,91	30,84
16	28,83	23,7

Для избегания влияния неравномерности загрузки узлов на результаты сравнения времени работы программы с линейной и матричной организации процессов были выполнены расчеты на одном узле. В табл. 2 приведено время решения задачи, полученное на одном из узлов с помощью 4 и 16 процессов. Видно, что способ организации процессов практически не влияет на время решения для 4 процессов, в то время как четырехкратное увеличение числа процессов ведет к экономии времени примерно на 20% в случае матричной организации.

Литература

1. Нигматулин Р.И. Основы механики гетерогенных сред. – М.: Наука, 1978. – 336 с.
2. Van Leer B. Towards the Ultimate Conservative Finite Difference Scheme III. Upstream centered Schemes for Hyperbolic Conservation Laws // J. Comp. Phys. – 1977. – Vol. 7, № 23. – P. 263–275.
3. Крайко А.Н. О поверхностях разрыва в среде, лишенной «собственного» давления // ПММ. – 1979. – Т. 43. – С. 500–511.
4. Миньков Л.Л., Шрагер Э.Р. Влияние доли несгоревших зерен дополнительного изделия на начальный участок работы ЭУ // Вычислительная газодинамика и горение твердых топлив. – Томск: Изд-во ТГПУ, 2001. – С. 160–166.
5. Корнеев В.Д. Параллельное программирование в MPI. – Новосибирск: Изд-во СО РАН, 2000. – 213 с.