РЕШЕНИЕ ПОЛНОЙ СИСТЕМЫ УРАВНЕНИЙ НАВЬЕ СТОКСА С РАСПАРАЛЛЕЛИВАНИЕМ ПРОЦЕССА ВЫЧИСЛЕНИЙ

Маркин Е.Е., Скачков П.П.

Аннотация
научно-исследовательской работы
«Решение полной системы уравнений Навье Стокса с распараллеливанием процесса вычислений»

Целью работы являетя разработка многопоточной программы, для полной системы уравнений Навье Стокса, решения которой описывают течения сжимаемого вязкого теплопроводного идеального газа при постоянных значениях коэффициентов вязкости и теплопроводности. Способ вычислений состоит в построении решений с помощью явной разностной схемы.

При решении представленной задачи был использован пакет программ – DevC++ 5.11 и мультимедийная кроссплатформенная библиотека SFML.

В работе для дифференциальных уравнений, в одномерном случае, описываются уравнения в конечных разностях. Рассказывается об идеи распараллеливания процесса. Изучаются способы организации потоков и их взаимодействий. Рассчитывается эффективность использования многопоточной программы.

Итогом работы является программный продукт, написанный на языке программирования С++ реализующий расчет величин давления, скорости течения и удельного объема.

Приводятся примеры графиков изучаемых величин как результат расчета по построенной программе включая вычисленные спектры Фурье коэффициентов.

В работе проведена анимация процесса протекающего в системе и получен видео ролик волновых явлений.

Abstract research work
"Solution of the Navier Stokes complete system of equations with parallelization of the computation process"
The aim of the work is the development of a multithread program for the complete Navier Stokes equations system, the solutions of which describe the flow of a compressible viscous heat-conducting ideal gas at constant values of the viscosity and thermal conductivity coefficients. The method of computation is in the construction of solutions using an explicit difference scheme.

When solving the presented problem, the software package was used - DevC ++ 5.11 and the multi-media cross-platform library SFML.

In the paper for differential equations, in the one-dimensional case, equations in finite differences are described. It is told about the idea of parallelizing the process. Methods for the organization of flows and their interactions are being studied. The efficiency of using a multithreaded program is calculated.

The result of the work is a software product written in the C ++ programming language that calculates the values of pressure, flow velocity and specific volume.

Examples of the graphs of the studied quantities are given as a result of calculation of the constructed program including the calculated Fourier spectra of the coefficients.

In the work animation of the process proceeding in the system is carried out and a video of the wave phenomena is obtained.

Рассматривается полная система уравнений Навье Стокса, решения которой описывают течения сжимаемого вязкого теплопроводного идеального газа. В системе выполнен переход от переменной плотности и температуры к удельному объему и давлению. Это позволяет решать систему уравнений с частными производными в нормальной форме относительно производных по времени.

В одномерном случае в безразмерных переменных эта система выглядит так [1,3]:

$$\begin{cases} \delta_t = \delta u_x - u\delta_x, \\ u_t = -uu_x - \dfrac{1}{\gamma}\delta p_x + \mu_0 \delta u_{xx}, \\ p_t = -up_x - \gamma p u_x + \kappa_0 (\delta p)_{xx} + \mu_0 \gamma(\gamma - 1) u_x^2, \end{cases} \qquad (1)$$

где $t-$ время, x $-$пространственная переменная, $u-$ скорость течения газа $\delta = 1/\rho$ – удельный объем и p – давление. Кроме того μ_0, κ_0, γ коэффициенты вязкости теплопроводности и показатель политропы соответственно.

Для системы (1) рассматривается начально-краевая задача. Именно, на отрезке $[0,\pi]$ заданы начальные и краевые условия вида (2)

$$\begin{cases} \delta\big|_{t=0} = \delta^0(x), \\ u\big|_{t=0} = u^0(x), \\ p\big|_{t=0} = p^0(x), \\ u\big|_{x=0,\ x=\pi} = 0, \quad T_x\big|_{x=0,\ x=\pi} = 0, \quad t \ge 0, \quad 0 \le x \le \pi. \end{cases}$$

Приближенные решения в одномерном случае находятся построением численных решений с помощью явной разностной схемы.

При построении решений с помощью разностных схем по пространственной переменной вводится равномерная сетка x_i $(0 \le x_i \le \pi, \quad 0 \le i \le N)$. Для дискретизации производных выбираются следующие стандартные выражения:

$$u_t \approx \frac{u_i^{n+1} - u_i^n}{\tau}, \qquad u_x \approx \frac{u_{i+1}^n - u_{i-1}^n}{2h}, \qquad u_{xx} \approx \frac{u_{i+1}^n - 2u_i^n + u_{i-1}^n}{h^2}.$$

Разностные уравнения для системы (1) в этом случае имеют вид:

$$\begin{cases} \dfrac{\delta_i^{n+1} - \delta_i^n}{\tau} = \delta_i^n \dfrac{u_{i+1}^n - u_{i-1}^n}{2h} - u_i^n \dfrac{\delta_{i+1}^n - \delta_{i-1}^n}{2h}, \\ \dfrac{u_i^{n+1} - u_i^n}{\tau} = -u_i^n \dfrac{u_{i+1}^n - u_{i-1}^n}{2h} - \dfrac{1}{\gamma}\delta_i^n \dfrac{p_{i+1}^n - p_{i-1}^n}{2h} + \mu_0 \delta_i^n \dfrac{u_{i+1}^n - 2u_i^n + u_{i-1}^n}{h^2}, \\ \dfrac{p_i^{n+1} - p_i^n}{\tau} = -u_i^n \dfrac{p_{i+1}^n - p_{i-1}^n}{2h} - \gamma p_i^n \dfrac{u_{i+1}^n - u_{i-1}^n}{2h} + \kappa_0 \dfrac{\delta_{i+1}^n p_{i+1}^n - 2\delta_i^n p_i^n + \delta_{i-1}^n p_{i-1}^n}{h^2} + \\ + \mu_0 \gamma(\gamma - 1)\left(\dfrac{u_{i+1}^n - u_{i-1}^n}{2h}\right)^2 \end{cases} \tag{3}$$

с начальными (4) и граничными (5) условиями:

$$(4) \quad \begin{cases} \delta\big|_{n=0} = \delta_i^0 = \delta^0(x_i), \\ u\big|_{n=0} = u_i^0 = u^0(x_i), \\ p\big|_{n=0} = p_i^0 = p^0(x_i), \end{cases}$$

$$u_i^n\Big|_{i=0,i=N} = 0, \quad T_x = (\delta p)_x\Big|_{i=0,i=N} = 0, \quad n \geq 0. \tag{5}$$

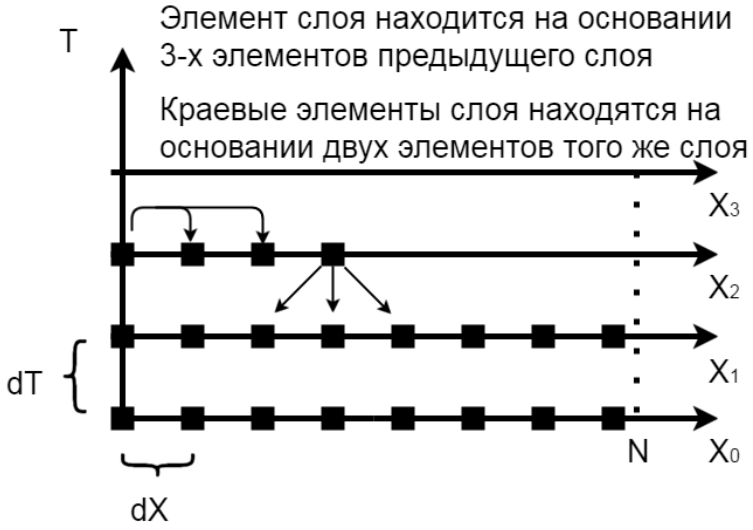
Равенство нулю производной температуры на концах отрезка $[0, \pi]$ аппроксимируется уравнениями:

$$\begin{aligned} \delta_0^{n+1} &= (4\delta_1^{n+1} - \delta_2^{n+1})/3, \qquad \delta_N^{n+1} = (4\delta_{N-1}^{n+1} - \delta_{N-2}^{n+1})/3, \\ p_0^{n+1} &= (4p_1^{n+1} - p_2^{n+1})/3, \quad p_N^{n+1} = (4p_{N-1}^{n+1} - p_{N-2}^{n+1})/3. \end{aligned} \tag{6}$$

Из системы (7) и граничных условий (9,10) получается разностная схема для определения значений неизвестных δ_i^{n+1}, u_i^{n+1}, p_i^{n+1} на следующем шаге по времени [2].

Построение программы и распараллеливание процесса вычислений

Алгоритм решения сеточных уравнений показан на рисунке 1.
Имеется первый слой - начальные условия. И затем на основании его вычисляем следующий слой. При этом используется явная схема пересчета, так как разностные уравнения нелинейные и неявную схему реализовать не имеет смысла из-за необходимости в обратном ходе решать нелинейную систему (метод прогонки невозможен). Задача будет решенной, если значения искомых переменных определены на всех «временных слоях».



Идея распараллеливания показана на рисунке 2. Как видно из условий задачи, распараллелить процесс мы можем только на участке нахождения элементов слоя. При этом потоки должны быть согласованны, то есть потоки должны приступать к решению следующего слоя одновременно. Первый поток также является главным, он решает, в

зависимости от состояния всех остальных потоков переходить ли на следующий слой, и производит проверку нужно ли записывать слой в файл.

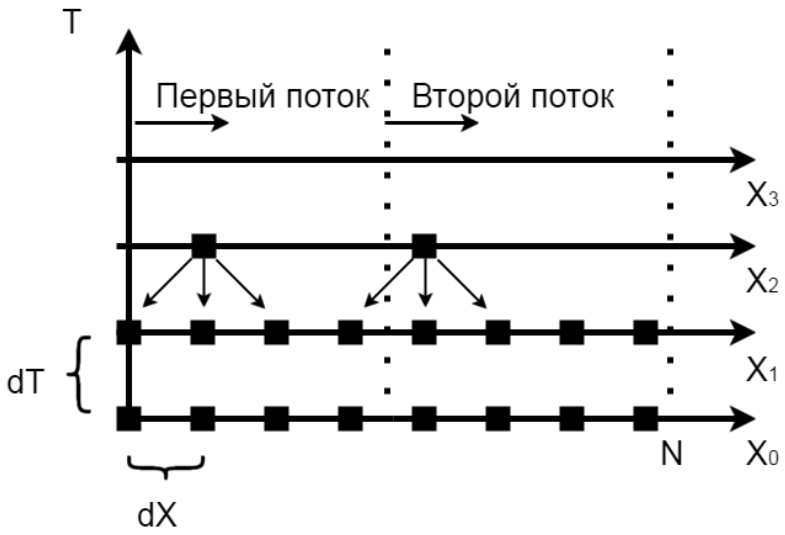


Рис. 2. Алгоритм нахождение элементов слоя двумя потоками.

Количество потоков, задействованных для решения задачи, должен определить сам пользователь, потоки создаются динамически.

Для решения задачи использовалась среда программирования Dev-C++.

Программа состоит из пяти подпрограмм.

void init(); - для задания начальных условий.

void kpud(byte ntt); - подпрограмма, находящая значения всех функций во всех узлах сетки.

void ThreadProc1(); - функция первого, главного потока.

void ThreadProc(byte A); - функция второго и последующего потока.

int main(); - главная подпрограмма реализующая запуск потоков и вывод информации в консоль.

Функции kpud(byte ntt); и ThreadProc(byte A); требуют аргумент – номер потока.

Создание потоков

Существует два способа создания потоков, с помощью кроссплатформенной мультимедийной библиотекой SFML и стандартным набором классов для работы с потоками доступными начиная с версии C++2011.

Параллельно запустить функцию позволяет библиотека SFML (англ. Simple and Fast Multimedia Library — простая и быстрая мультимедийная библиотека) - Свободная кроссплатформенная мультимедийная библиотека. Написана на C++, но доступна также для C, D, Java, Python, Ruby, OCaml, .Net и Go. Представляет собой объектно-ориентированный аналог SDL

Подключаем библиотеку SFML следующим образом.

1. Определяем версию установленного компилятора C++. Это важно, так как разные версии компиляторов GCC не совместимы между собой:
   В нашем случае была доступна версия на 1 апреля 2017 года DEV-C++ версии 4.8.1 TDM (SJLJ).
2. скачиваем 32-х разрядную и 64-разрядную версии библиотеки SFML GCC 4.8.1 TDM (SJLJ). (Ссылки на библиотеки доступны в конце статьи).
   SFML-2.2-windows-gcc-4.8.1-tdm-32-bit.zip
   SFML-2.2-windows-gcc-4.8.1-tdm-64-bit.zip
3. Распаковываем оба архива в отдельные папки на жестком диске (лучше, чтобы путь не содержал кириллических символов). Например , 32-х разрядную версию распакуем в папку C:\Dev-Cpp\SFML-2.2-32, а 64-х разрядную - в папку C:\Dev-Cpp\SFML-2.2-64.
4. Переходим к настройке среды разработки DEV C++. Открываем «Сервис» -> «Параметры компилятора» -> вкладка «Компилятор»;
5. Для каждого набора настроек компилятора добавляем новые параметры в поле «Добавить эти команды к командной строке компоновщика».
   Код
   -lsfml-graphics -lsfml-window -lsfml-audio -lsfml-system

6) Переходим к вкладке «Каталоги» (рисунок 4) и так же, для каждого набора настроек компилятора прописываем пути для:
   «Библиотек»:
   для 32-х разрядной версии - D:\Soft\Dev-Cpp\SFML-2.2-32\lib
   для 64-х разрядной версии - D:\Soft\Dev-Cpp\SFML-2.2-64\lib
   «Включаемых файлов C»:
   для 32-х разрядной версии - D:\Soft\Dev-Cpp\SFML-2.2-32\include
   для 64-х разрядной версии - D:\Soft\Dev-Cpp\SFML-2.2-64\include
   «Включаемых файлов C++»:
   для 32-х разрядной версии - D:\Soft\Dev-Cpp\SFML-2.2-32\include
   для 64-х разрядной версии - D:\Soft\Dev-Cpp\SFML-2.2-64\include

На этом настройка IDE DEV C++ закончена.

После проведенных действий, подключаем библиотеку, прописывая в начале кода программы.

#include <SFML/System.hpp>

В функции main() инициализируем объект потока, и запускаем его.

sf::Thread thread = new sf::Thread(&ThreadProc1);*

```
thread->launch();
```

Далее в цикле запускаем функцию ThreadProc(byte A); и сообщаем ей номер потока.

```
for (byte i=2;i<=THR;i++)
{       thread = new sf::Thread(&ThreadProc,i);
        thread->launch();  }
```

Таким образом, мы можем динамически создавать потоки, указывая константе THR количество задействованных потоков.

Так как требуется синхронизация потоков, то в главной функции пишем цикл ожидания завершения работы остальных потоков. Если хоть один поток еще не рассчитал слой, то переменная типа bool станет true и цикл будет повторен.

```
do
{               next=false;
        for (thh=2;thh<=THR;thh++)
        {  if (n_t[thh]!=n_t[1]) next = true; }}
                while(next);
```

Если все потоки завершены, осуществляется переход на новый слой.

Функция второго и последующего потока.

В данной функции описан цикл проверки слоя, на котором находится первый поток, если первый поток переходит на следующий слой, то текущий поток вызывает функцию обработки своего участка слоя.

```
void ThreadProc(byte A)
{
        do
        {
                if (n_t[A]!=n_t[1]){//обрабатываем новую строку
                        kpud(A);//Если строка новая то вызываем функцию
                        n_t[A]=n_t[1];
                }
        }
        while(!thr_stop);
}
```

Ссылка на полный исходный код программы доступен в конце статьи.

Подсчет эффективности работы программы

Для подсчета времени работы программы воспользуемся процессорным временем. Процессорное время увеличивается, когда процесс работает и потребляет циклы CPU. Во

время операций ввода-вывода, блокировок потоков и других операций, которые приостанавливают работу процессора, процессорное время не увеличивается, пока процесс снова не начнет использовать CPU.

Для такого расчета служит функция getCPUTime();

Вызываем getCPUTime( ) до и после запуска алгоритма, и получаем разницу в секундах.

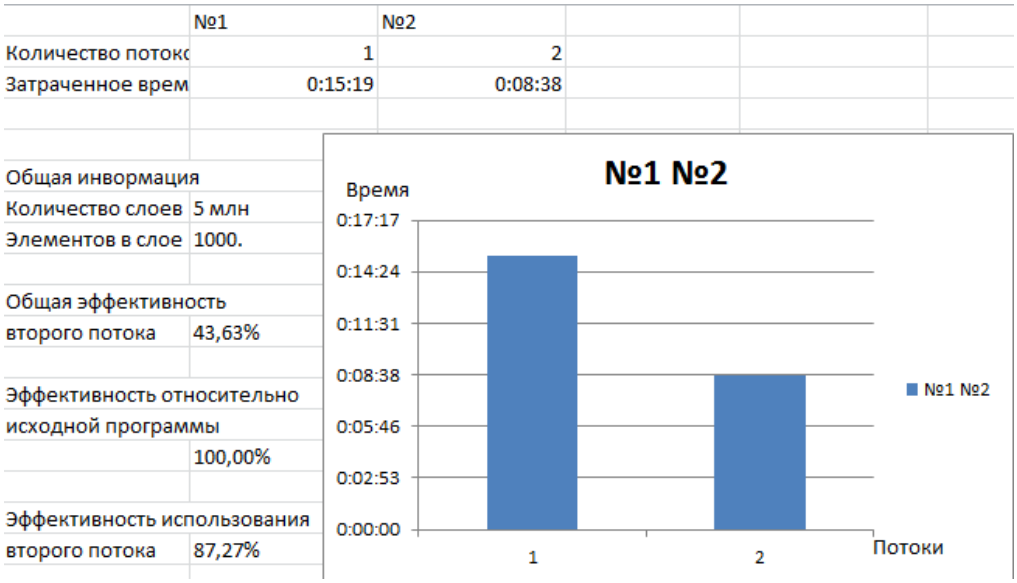


Рис. 3. Подсчет эффективности работы программы

Время одного полного цикла первого потока рассчитывается по первой формуле (7)

Время полного цикла второго и последующего потока рассчитывается по второй формуле (7)

$$T_{глав} = t_{алг} + 19...24 \cdot t_{команд}$$
$$T_{потоков} = t_{алг} + 4 \cdot t_{команд} \qquad (7)$$

Если учесть что t_алг очень большое, а t_ком очень маленькое (элементарные команды: сложение, сравнение…), то можно предположить что первый поток фактически работает всегда. А у второго время простоя очень незначительно. Это также видно из конечного результата (Рисунок 5), время сократилось на 43,63%. Если учесть, что первый поток активен всегда, то расчеты показывают, что второй загружен на 87,27 %.

Результаты численных расчетов решения начально-краевой задачи

При значениях вязкости $\mu_0 \geq 0.001, \quad \kappa_0 = 1.453\mu_0$ и малых значениях h и τ ($h \approx 0.001, \quad \tau \approx 0.0001$) на достаточно больших промежутках времени ($t \approx 1/\mu_0$) графики давления при начальных условиях

$$p^0(x) = 1 + 0.1\cos(3x),\ \ u^0(x) = 0,\ \ \delta^0(x) = 1,$$

то характерный вид графика давления имеет вид

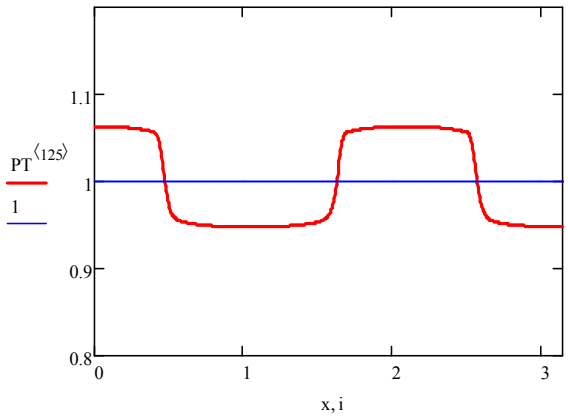


Рис. 4. График давления, вычисленные разностным методом, при $t = 12$

Видео ролик волнового процесса можно посмотреть, обратившись к облачному файлу по второй ссылке.

https://cloud.mail.ru/public/NEip/J4DFydQgy

https://cloud.mail.ru/public/EUu8/hD9B372Eo

Литература

1. Титов С.С. Пространственно-периодические решения полной системы Навье Стокса. // Доклады РАН. − 1999. − Т. 365, №6. − С. 761−763.
2. Замыслов В.Е., Скачков П.П. Сравнение двух приближенных методов решения одной начально-краевой задачи газовой динамики с учетом вязкости и теплопроводности // Вестник УрГУПС.– 2012.– № 4(16).–С.29–38.
3. Баутин С.П., Замыслов В.Е. Представление приближенных решений полной системы уравнений Навье-Стокса в одномерном случае. // Вычислительные технологии. М.: − 2012. − Т. 17, №3. − С. 3−12. ISSN 1560-7534.
4. Mathcad 14 для студентов, инженеров и конструкторов.: БХВ-Петербург, 2007-368 с.
5. Романов Е.Л. - Практикум по программированию на С++ - 2004. ВУЗ: СумГУ.