УДК519.6

ГЕТЕРОГЕННЫЕ ПАРАЛЛЕЛЬНЫЕ ВЫЧИСЛЕНИЯ НА ПРИМЕРЕ РЕШЕНИЯ ПОЛНОЙ СИСТЕМЫ УРАВНЕНИЙ НАВЬЕ-СТОКСА МЕТОДОМ СЕТОК.

Маркин Е. Е., Скачков П. П.

Уральский государственный университет путей сообщения

Целью работы является разработка многопоточной программы, для полной системы уравнений Навье Стокса[1], решения которой описывают двухмерные течения сжимаемого вязкого теплопроводного идеального газа при постоянных значениях коэффициентов вязкости и теплопроводности. Способ вычислений состоит в построении решений с помощью явной разностной схемы.

Решение систем дифференциальных уравнений в частных производных методом сеток требует больших затрат машинного времени. В последние годы разрабатываются методы [3] использующие графические процессоры в аппаратно-программном способе вычислений.

В данной работе используются графические карты, для реализации массово-параллельных вычислений общего назначения. Примером данной технологии является NVidia CUDA (Compute Unified Device Architecture) — программно-аппаратная архитектура для параллельных вычислений на NVidia GPU. При этом показано, что ускорение расчетов зависит от размерности сетки и может достигать сотни раз относительно обычного распараллеливания на CPU.

При помощи указанной технологии сделаны расчеты величин удельного объема, давления и составляющих скорости. В работе проведена анимация процесса протекающего в системе и приводится видео ролик соответствующих волновых явлений.

Ключевые слова: полная система уравнений Навье-Стокса, гетерогенные вычисления, CUDA, графический процессор, поток, нить, ядро.

HETEROGENEOUS PARALLEL CALCULATIONS ON THE EXAMPLE OF SOLVING THE COMPLETE SYSTEM OF NAVIER-STOKES EQUATIONS BY THE METHOD OF FINITE DIFFERENCES

Markin E.E., Skachkov P.P.

Ural State University of Railway Transport

The aim of the paper is to develop a multithreaded program for the complete Navier Stokes equations system[1], whose solutions describe two-dimensional flows of a compressible viscous heat-conducting ideal gas at constant values of the viscosity and thermal conductivity coefficients. The method of computation consists in constructing solutions using an explicit difference scheme.

The solution of systems of partial differential equations by the grid method requires a large expenditure of computer time. In recent years, methods have been developed [3] using graphics processors in the hardware-software computational method.

In this paper, graphics cards are used to implement mass-parallel computations of general purpose. An example of this technology is NVidia CUDA (Compute Unified Device Architecture) - software and hardware architecture for parallel computing on the NVidia GPU. It is shown that the acceleration of calculations depends on the dimension of the grid and can reach hundreds of times relative to the usual parallelization on the CPU.

With the help of this technology calculations are made of the specific volume, pressure and velocity components. In the work animation of the process of the protester in the system is carried out and a video of the corresponding wave phenomena is shown.

Keywords: complete Navier-Stokes equations system, CUDA, GPU, CPU, stream, thread.

Решение полной системы уравнений Навье-Стокса методом сеток

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 двухмерном случае для безразмерных переменных эта система имеет вид [1]:

$$\begin{cases} \delta_t + u\delta_x + v\delta_y - \delta(u_x + v_y) = 0, \\ u_t + uu_x + vu_y + \dfrac{1}{\gamma}\delta p_x = \mu_0\delta\left(\dfrac{1}{4}v_{xy} + u_{xx} + \dfrac{3}{4}u_{yy}\right), \\ v_t + uv_x + vu_y + \dfrac{1}{\gamma}\delta p_y = \mu_0\delta\left(\dfrac{1}{4}u_{xy} + v_{yy} + \dfrac{3}{4}v_{xx}\right), \\ p_t + up_x + vp_y + \gamma p(u_x + v_y) = \kappa_0 p(\delta_{xx} + \delta_{yy}) + \\ +2\kappa_0(p_x\delta_x + p_y\delta_y) + \kappa_0\delta(p_{xx} + p_{yy}) + \\ \mu_0\gamma(\gamma-1)\left[(u_x^2 - u_x v_y + v_y^2) + \dfrac{3}{4}(u_y + v_x)^2\right]. \end{cases} \quad (1)$$

Здесь t – время, x, y – независимые пространственные переменные. Далее, $\delta = \dfrac{1}{\rho}$ – удельный объем газа, ρ – плотность газа, p – давление, $\mathbf{V} = (u, v)$ – вектор скорости газа с его проекциями на декартовы оси координат Ox, Oy. Постоянные коэффициенты в уравнениях μ_0, κ_0 – коэффициенты вязкости и теплопроводности, $\gamma > 1$ – показатель политропы идеального газа с уравнениями состояния, записанными в безразмерных переменных $p = \rho T$. Для системы (1) рассматривается начально-краевая задача. В области $(0 \le x \le \pi;\ 0 \le y \le \pi)$ задаются начальные условия вида

$$\begin{aligned} &\delta(t,x,y)\big|_{t=0} = \delta^0(x,y);\ u(t,x,y)\big|_{t=0} = u^0(x,y); \\ &v(t,x,y)\big|_{t=0} = v^0(x,y);\ p(t,x,y)\big|_{t=0} = p^0(x,y). \end{aligned} \quad (2)$$

А также краевые условия

$$\begin{aligned} &u(t,x,y)\big|_{x=0,x=\pi,y=0,y=\pi} = 0,\ v(t,x,y)\big|_{x=0,x=\pi,y=0,y=\pi} = 0, \\ &T_x(t,x,y)\big|_{x=0,x=\pi,y=0,y=\pi} = 0,\ T_y(t,x,y)\big|_{x=0,x=\pi,y=0,y=\pi} = 0. \end{aligned} \quad (3)$$

Первые из краевых условий (3) обеспечивает прилипание газа на границах области, а вторые – обеспечивает теплоизоляцию на границах. В соответствии с уравнением состояния $p\delta = T$, тогда $T_x = p\delta_x + \delta p_x$, $T_y = p\delta_y + \delta p_y$. Поэтому для обеспечения теплоизоляции достаточно принять

$$\delta_x\big|_{x=0,x=\pi,y=0,y=\pi} = \delta_y\big|_{x=0,x=\pi,y=0,y=\pi} = 0,$$
$$p_x\big|_{x=0,x=\pi,y=0,y=\pi} = p_y\big|_{x=0,x=\pi,y=0,y=\pi} = 0.$$

При построении решений с помощью разностных схем по пространственным переменным вводится равномерная сетка $x_i\,(0 \le x_i \le \pi,\ \ 0 \le i \le N)$, $y_j\,(0 \le y_j \le \pi,\, 0 \le j \le M)$. Для дискретизации производных выбираются следующие стандартные выражения (4).

$$\frac{\partial f}{\partial t} = \frac{f_{i,j}^{n+1} - f_{i,j}^{n}}{\Delta t},\ \frac{\partial f}{\partial x} = \frac{f_{i-1,j}^{n} - f_{i+1,j}^{n}}{2\Delta x},\ \frac{\partial f}{\partial y} = \frac{f_{i,j-1}^{n} - f_{i,j+1}^{n}}{2\Delta y},$$
$$\frac{\partial^2 f}{\partial x^2} = \frac{f_{i+1,j}^{n} - 2f_{i,j}^{n} + f_{i-1,j}^{n}}{(\Delta x)^2},\ \frac{\partial^2 f}{\partial y^2} = \frac{f_{i,j+1}^{n} - 2f_{i,j}^{n} + f_{i,j-1}^{n}}{(\Delta y)^2}, \qquad (4)$$
$$\frac{\partial^2 f}{\partial x \partial y} = \frac{f_{i+1,j+1}^{n} - f_{i-1,j+1}^{n} - f_{i+1,j-1}^{n} + f_{i-1,j-1}^{n}}{4\Delta x \Delta y}.$$

Равенство нулю производной температуры на границах области аппроксимируется уравнениями:

$$p_{0,j} = (4p_{1,j} - p_{2,j})/3,\quad p_{n,j} = (4p_{n-1,j} - p_{n-2,j})/3,$$
$$\delta_{0,j} = (4\delta_{1,j} - \delta_{2,j})/3,\quad \delta_{n,j} = (\delta_{n-1,j} - \delta_{n-2,j})/3.$$

$$p_{i,0} = (4p_{i,1} - p_{i,2})/3,\quad p_{i,m} = (4p_{i,m-1} - p_{i,m-2})/3,$$
$$\delta_{i,0} = (4\delta_{i,1} - \delta_{i,2})/3,\quad \delta_{i,m} = (4\delta_{i,m-1} - \delta_{i,m-2})/3. \qquad (5)$$

Схема решения задачи на GPU с распараллеливанием

Некоторые понятия[2].

- хост (host) = центральный процессор (CPU);
- устройство (device) = графический процессор (GPU);
- поток (stream) - логическая последовательность зависимых асинхронных операций, независимая от операций в других потоках;
- нить (thread) - элементарный параллельный процесс. Все нити группируются в иерархию - grid/block/thread
- грид(grid) - множество процессов, порождаемых запуском ядра

- блок (block) - множество нитей, в рамках блока нити могут быть синхронизованы, и могут иметь общую разделяемую (shedder) память.

Нити в блоках и блоки в гриде могут быть представлены в виде одно-, двух- или трёхмерной решетки, мы будем рассматривать одномерный грид и блок.

- варп (warp) - группа нитей, размер варпа 32 нити. Все нити одного варпа выполняются одновременно и синхронно (SIMD) на своём мультипроцессоре.
- ядро (kernel) - функция, параллельно выполняемая потоками на GPU;

Программа состоит из двух частей:

- host-кода (управляющего кода) написанного на обычном C/C++, выполняется на CPU
- device-кода (аппаратного кода) на GPU исполняются специальные функции – ядра(kernel) и функции, вызываемые внутри них.

Ядро является потоковой (stream) функцией – большое количество нитей (threads) параллельно исполняют тело ядра(kernel). Ядро вызывается со стороны CPU, при этом указывается количество блоков и количество нитей в каждом блоке, которые будут его исполнять, а так же номер потока в котором будет выполняться ядро. Если номер потока не указан, то ядро будет выполнено в нулевом потоке. Нулевой поток всегда синхронен.

Потоки бывают синхронные и асинхронные. Синхронный поток означает, что после вызова ядра, CPU будет ждать, когда GPU завершит работу и после продолжит алгоритм. Если поток асинхронен, то CPU после вызова ядра, продолжит выполнять алгоритм, не дожидаясь завершения работы GPU.

Имеем слой $M \times N$ элементов с шагом dx и dy соответственно[3,4]. На основании пяти элементов данного слоя с шагом по времени dt вычисляются элементы следующего слоя. На CPU задача решается введением двойного цикла. Идея решения задачи с распараллеливанием заключена в том, чтобы все элементы слоя были вычислены одновременно, то есть на GPU в рамках первого этапа запускается один поток с количеством нитей равном $(M-2)\times(N-2)$, которые параллельно вычисляют элементы следующего слоя с шагом dt. Процесс продемонстрирован на рисунке 1. По цветам, черные – вычисленные, белые – необходимо вычислить, желтые – вычисляются.

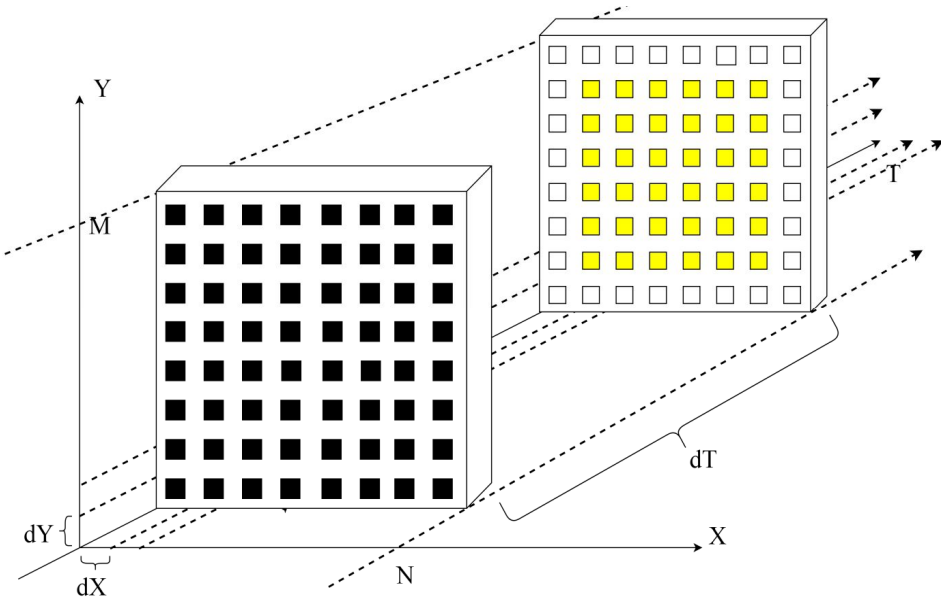

Рис.1 Первый этап - нахождение элементов в следующем слое на GPU.

2 этап заключается в вычислении граничных элементов. Граничные элементы вычисляются в два асинхронных потока (зеленым выделен первый поток, красным второй поток), первый имеет $N-2$ нитей и каждая нить вычисляет два элемента, второй поток имеет $M-2$ нитей и каждая нить также вычисляет два элемента. Для наглядности процесс показан на рисунке 2.

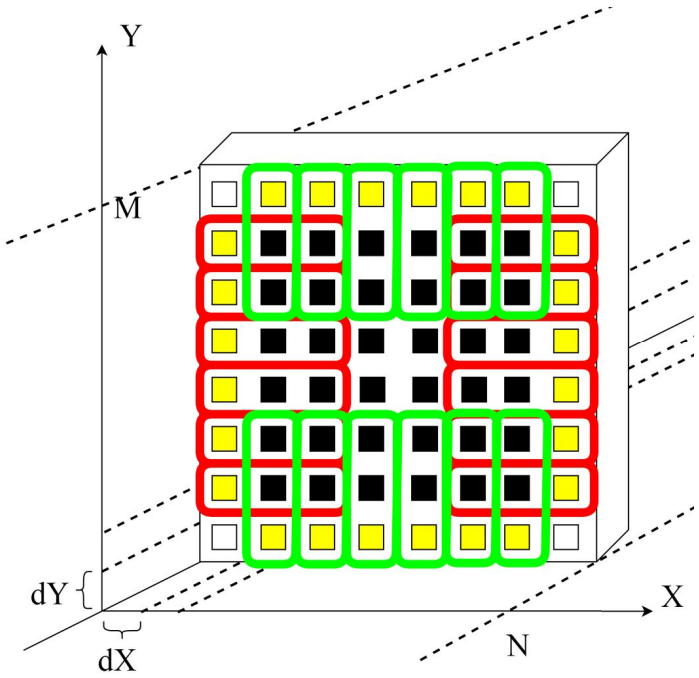

Рис.2 Второй этап – нахождение краевых элементов на GPU.

Угловые элементы вычисляются в конце второго этапа нитью с глобальным индексом “0”.

Приведем листинг вызова ядер (рис. 3)[5,6], для нахождения элементов слоя. Как видно из листинга первый вызов ядра проходит в нулевом потоке, нулевой поток синхронен, как если бы cudaDeviceSynchronize() (функция синхронизации) вставлен до и после каждой CUDA операции. Второй и третий вызов ядра происходит в асинхронных потоках, то есть они выполняются одновременно.

```
cudaStream_t stream1, stream2;//Создание потоков
cudaStreamCreate ( &stream1);
cudaStreamCreate ( &stream2);
do
{
    // Внутри массива, запуск в нулевом потоке
    CUDA_Kernel_1 <<< index.block, index.thread, 0 >>> ();
    // Элементы по оси y запуск в асинхронном потоке
    CUDA_Kernel_2 <<< idx_m.block, idx_m.thread, 0, stream1 >>> ();
    // Элементы по оси x
    CUDA_Kernel_3 <<< idx_n.block, idx_n.thread, 0, stream1 >>> ();
    cudaDeviceSynchronize();//Ждем завершение работы асинхронных потоков
    /* код программы
    T = T + dT;
} while (T <= TK);
```

Рис. 3 – Листинг организации вызова ядер.

Эффективность использования CPU и CPU+GPU

В Табл. 1 приведено программно-аппаратное обеспечение, на котором производились расчеты. В Табл. 2 полученные результаты расчетов.

Программно-аппаратная часть. Таблица 1

Аппаратное обеспечение	
Видео адаптер	Nvidia Geforce GTX 650, 1024 Мб DDR5
Центральный процессор	Intel Pentium G3260 3,3 ГГц
Оперативная память	2 ГБ 2xDDR3-1333 ГГц
Программное обеспечение	
Операционная система	Windows 7 64 bit
Драйвер	NVIDIA Driver for Windows 381.65
Среда программирования	Microsoft Visual Studio 2015 Enterprise
Версия CUDA Toolkit	CUDA Toolkit 8.0.61 for Windows

Сравнение скорости вычислений Таблица 2

N,M	GPU+CPU, мс	CPU, мс	Во сколько раз быстрей
256	0,6	19,9	33,13
512	2,1	117,6	56,00
768	5,1	397,6	77,96
1024	8,5	746,48	87,82
1280	14,1	1204,0	85,39
1536	20,4	1810,5	88,75
1792	27,8	2584,4	92,96
2048	34,1	3432,8	100,67

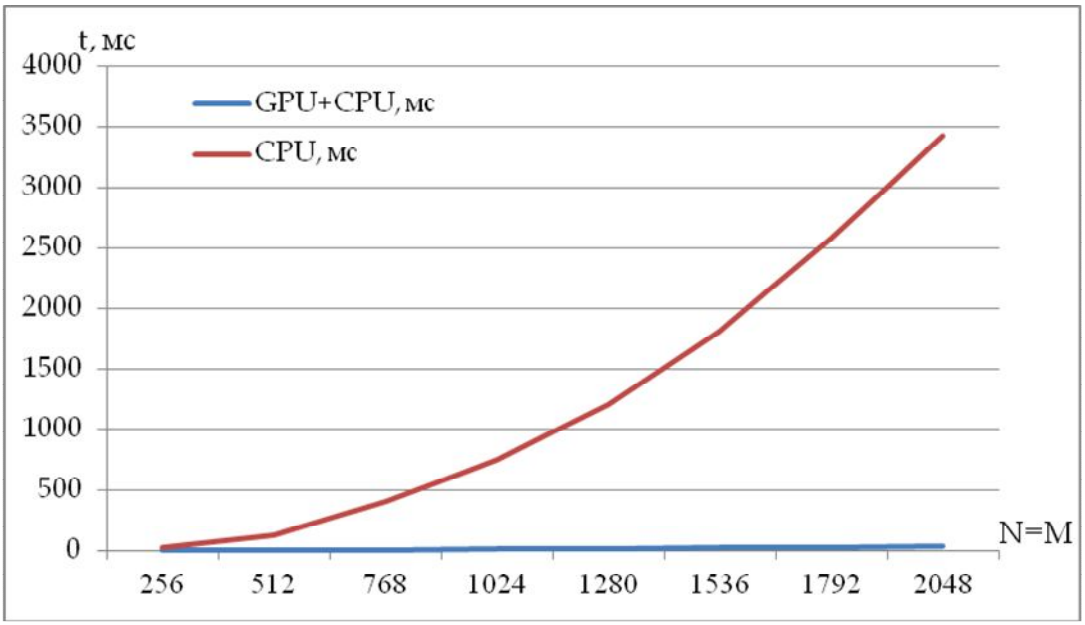

Рис.4 График затраченного времени на вычисление одного слоя.

Как видно из результатов (рис 4), при параллельной работе графического адаптера ускорение обработки данных увеличивается в десятки раз. Отношение времени обработки данных на графическом адаптере и центральном процессоре не постоянно, а зависит от размера задачи. Чем больше объем задачи, тем выгоднее использовать графический процессор.

Результаты моделирования

В соответствии с системой (1), начальными (2) и граничными (3) условиями записываются разностные уравнения. Для конкретного расчета выберем значения переменных для момента времени t=0, $\delta^0 = 1,\ u^0 = 0,\ v^0 = 0,\ p^0 = 1 + 0.2\cos(\sqrt{x^2 + y^2}),$ и значения констант $\mu_0 = 0.001,\ \kappa_0 = 1.45833\mu_0,\ \gamma = 1.4$ и шага по времени $\Delta t = 10^{-5}$.

Расчет произведен при следующих параметрах. Размер сетки - $M = N = 256$ шаг моделирования - $dt = 10^{-5}$ Коэффициенты вязкости и теплопроводности. $\mu_0 = 10^{-3}$, $\kappa_0 = 1.45633\mu_0$ Для контроля адекватности модели и точности вычислений был проведены

расчеты на больших промежутках времени до достижения состояния однородного покоя. При $\mu_0 = 10^{-3}$ этот промежуток приближенно равен $t_{покоя} \approx 1000$.

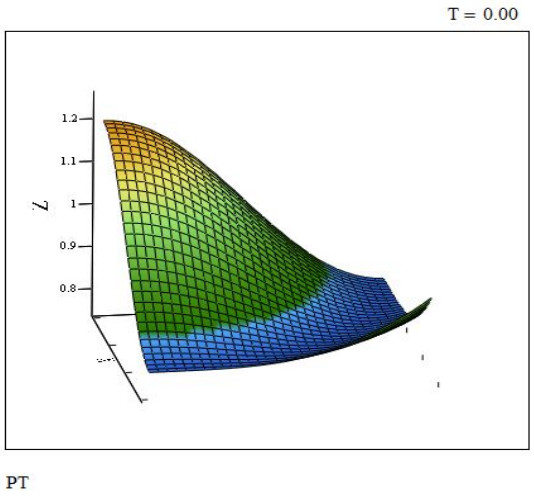

Рис. 5 Давление в момент времени t=0

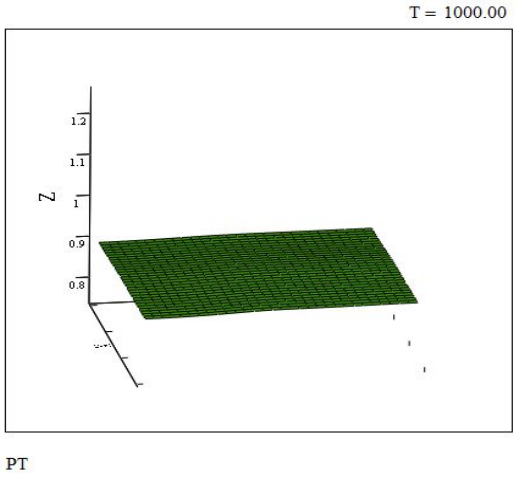

Рис. 6 – Давление в момент времени t=1000

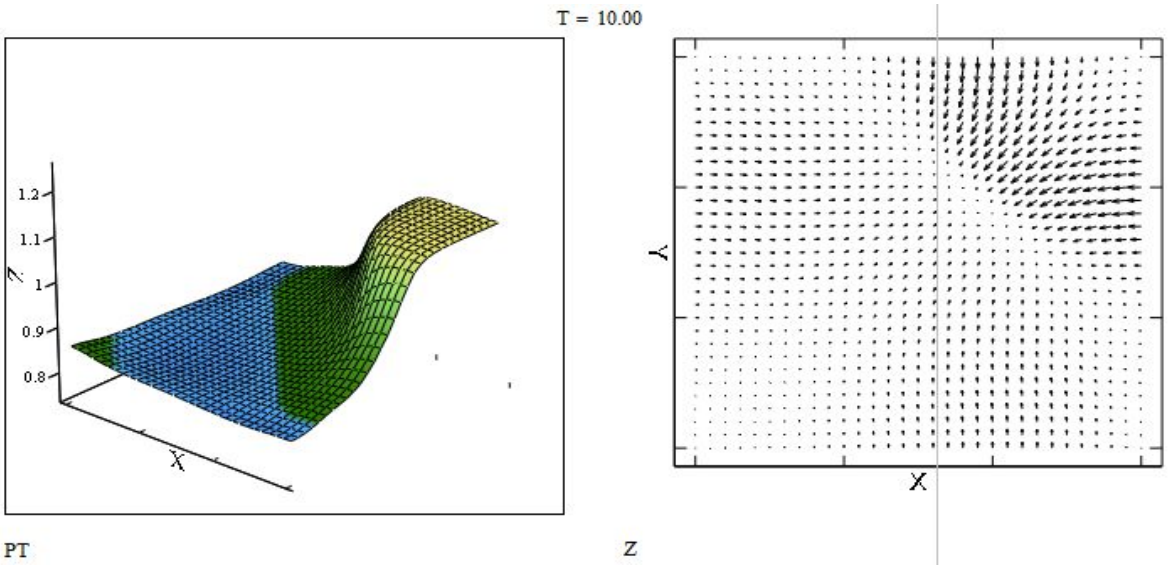

Рис. 7 Давление и распределение скоростей в момент времени t=10

Разница колебаний между минимумом и максимумом давления на покое составляет менее 0,016 (рисунок 6).

Затраченное машинное время на моделирование с использованием GPU 15 часов 30 минут. Всего было найдено 100 млн слоев, такая задача на CPU решалась бы около 3-х недель.

Видео моделирования доступно по ссылке:
https://cloud.mail.ru/public/LTLz/BQyLMM3Jf

Список литературы

1. Баутин С. П.. Замыслов В. Е. Об одном представлении приближенных решений полной системы уравнений Навье-Стокса // Проблемы прикладной математики и механики. – Екатеринбург: УрГУПС.– 2011. Вып. 95(178)/6м.– С. 5 – 16.

2. Боресков А. В. Предисл.: Садовничий В. А. Параллельные вычисления на GPU. Архитектура и программная модель CUDA: Учебное пособие. – М.: Изд. МГУ, 2012. – 336 С. ISBN: 978-5-211-06340-2
3. Иноземцева Н. Г., Перепёлкин Е. Е., Садовников Б. И. Оптимизация алгоритмов задач математической физики для графических процессоров. – М.: Изд.: Физический факультет МГУ , 2012. 240 С. ISBN: 978-5-8279-0107-5
4. Перепёлкин Е. Е., Садовников Б. И., Иноземцева Н. Г. Вычисления на графических процессорах (GPU) в задачах математической и теоретической физики. – М.:Изд: URSS, 2014, 176 С. ISBN: 978-5-9710-1085-2
5. Параллельное программирование с CUDA. Часть 1: Введение. [Электронный ресурс]// URL: https://habrahabr.ru/company/epam_systems/blog/245503/ (дата обращения 20.07.2017)
6. CUDA: Работа с памятью. [Электронный ресурс]// часть 1. URL: https://habrahabr.ru/post/55461/ ,часть 2. URL: https://habrahabr.ru/post/56514/ (дата обращения 21.07.2017).