ИСПОЛЬЗОВАНИЕ ГРАФИЧЕСКИХ ПРОЦЕССОРОВ С ПОДДЕРЖКОЙ ТЕХНОЛОГИИ CUDA ДЛЯ ДВУМЕРНОГО МАТЕМАТИЧЕСКОГО МОДЕЛИРОВАНИЯ ГАЗОДИНАМИЧЕСКИХ ПРОЦЕССОВ В СОПЛЕ ЛАВАЛЯ

И.В. Алексеева, А.П. Будник, Ф.А. Петрушин, А.В. Сипачев

ГНЦ РФ Физико-энергетический институт,
г. Обнинск, 249033, Калужская область, пл. Бондаренко,1

Аннотация

Рассматривается применение технологии CUDA, разработанной компанией NVIDIA, для решения задач трансзвуковой газодинамики. Технология CUDA применялась для переноса части вычислений с центрального процессора (CPU) на графический процессор (GPU). Проведено сравнение скорости вычислений как с применением только центрального процессора, так и с применением центрального и графического процессора одновременно.

USING TECHNOLOGY CUDA FROM COMPANY NVIDIA FOR TWO-DIMENSIONAL MATHEMATICAL MODELLING OF GAS-DYNAMIC PROCESS IN LAVAL NOZZLE

I.V. Alexeeva, A.P. Budnik, F.A. Petrushin, A.V. Sipachev

State Scientific Center of Russian Federation, A.I. Leypunski Institute of Physics and Power Engineering, Russia, Obninsk, 249033

Using technology CUDA from company NVIDIA is consider for the solution of problems of a transonic gas-dynamic. Technology CUDA was applied to carry a part of computing from the central processor (CPU) on the graphic processor (GPU). Comparison of computation speed is spent with computing only the central processor, and with application of the central and graphic processor simultaneously.

1. ВВЕДЕНИЕ

В настоящее время стремительно развиваются вычислительные технологии для исследования процессов в области течения жидких и газообразных сред (ANSYS CFX, COSMOS FLOWORKS, FLUENT, FLOW 3D, FAST FLOW, а также российский пакет FLOW VISION). Такое физическое моделирование, в частности, позволяет проводить оптимизацию разработки новых конструкций путем тестирования многочисленных вариантов их работы, позволяя предсказать различные сценарии развития процесса при варьировании исходных данных.

Эти вычислительные технологии позволяют существенно увеличить скорость математического моделирования требуемого процесса.

Все эти средства, в большей или меньшей степени, обладают следующими отрицательными чертами.

Во-первых, в настоящее время на рынке отсутствуют коммерческие программные продукты, созданные для математического моделирования двумерных и трехмерных, как нестационарных, так и установившихся колебательно-неравновесных трансзвуковых течений вязкой теплопроводящей многокомпонентной смеси газов.

Во-вторых, математическое моделирование трансзвукового течения вязкого теплопроводящего газа основано на приближенных методах, даже если не учитываются в многокомпонентной смеси газов колебательно-неравновесные процессы. При этом применение различных программных продуктов дает заметно отличающиеся друг от друга результаты. Эти различия ещё более увеличиваются при использовании различных моделей турбулентности. Это, в частности, делает необходимым проводить расчеты с применением различных программных комплексов для повышения надежности расчетных результатов и грубой оценки их погрешности.

В-третьих, стоимость коммерческих программных продуктов весьма высока, поскольку при применении в научных исследованиях требует существенной доработки, как например, в данном случае - учета неравновесных кинетических процессов. Причем нет никаких гарантий, что разработать дополнение возможно и оно будет адекватно функционировать.

В-четвертых, не все из этих программных комплексов позволяют эффективно использовать параллельную обработку данных. Это касается в частности параллельных вычислений с использованием графических процессоров.

Целью данной работы была модификация существующего программного комплекса GDL-2D, с целью использования вычислительных способностей графического процессора для ускорения обработки данных. Был проведен сравнительный анализ технологий программирования с использованием графических процессоров. После анализа их особенностей и возможностей была выбрана технология CUDA (Compute Unified Device Architecture), разработанная компанией NVIDIA, как наиболее удобная с точки зрения программирования и представленных возможностей.

2. КРАТКОЕ ОПИСАНИЕ ПРОГРАММНОГО КОМПЛЕКСА GDL-2D

Программный комплекс GDL-2D позволяет проводить двумерное математическое моделирование нестационарных колебательно-неравновесных течений вязкой теплопроводящей многокомпонентной смеси газов в тракте низкотемпературного воздушного газодинамического лазера содержавшего сопло Лаваля для обоснования выбора как конструкции, так и достижимых параметров низкотемпературного воздушного газодинамического лазера. Его модифицированный вариант позволяет использовать технологию CUDA.

Для программного комплекса GDL-2D была разработана теоретическая модель, сформулирована система уравнений неравновесной газовой динамики и кинетики многокомпонентной смеси, а также разработан метод решения этих уравнений.

Всё это является теоретической основой комплекса программ GDL-2D с целью решения уравнений двумерной модели с поуровневым учетом колебательно-неравновесных кинетических процессов в среде, которая представляет собой смесь газов $CO_2 - N_2 - O_2 - H_2O$. Эта модель позволяет корректно рассматривать процессы релаксации при больших отклонениях от термодинамического равновесия, которое имеет место в активной среде при начальных температурах смеси менее 1200°K.

3. МЕТОД ЧИСЛЕННОГО РЕШЕНИЯ ЗАДАЧИ

Задача заключалась в двумерном моделировании нестационарного колебательно-неравновесного течения вязкой теплопроводящей многокомпонентной смеси газов в тракте низкотемпературного воздушного газодинамического лазера.

Метод решения данной задачи основан на расщеплении полной системы уравнений газовой динамики колебательно-неравновесного, вязкого и теплопроводного газа на три группы, а также применение для интегрирования уравнений каждой из групп различных подходов.

Первую группу составляют уравнения Навье – Стокса:

$$\frac{\partial \rho}{\partial t} + \operatorname{div}(\rho \mathbf{V}) = 0, \tag{1}$$

$$\frac{\partial \rho v_x}{\partial t} + \operatorname{div}(\rho v_x \mathbf{V}) = -\frac{\partial p}{\partial x} - \frac{2}{3}\frac{\partial}{\partial x}(\mu \operatorname{div}\mathbf{V}) + \\ + \frac{\partial}{\partial y}\left[\mu\left(\frac{\partial v_x}{\partial y} + \frac{\partial v_y}{\partial x}\right)\right] + 2\frac{\partial}{\partial x}\left(\mu \frac{\partial v_x}{\partial x}\right), \tag{2}$$

$$\frac{\partial \rho v_y}{\partial t} + \operatorname{div}(\rho v_y \mathbf{V}) = -\frac{\partial p}{\partial y} - \frac{2}{3}\frac{\partial}{\partial y}(\mu \operatorname{div}\mathbf{V}) + \\ + \frac{\partial}{\partial x}\left[\mu\left(\frac{\partial v_x}{\partial y} + \frac{\partial v_y}{\partial x}\right)\right] + 2\frac{\partial}{\partial y}\left(\mu \frac{\partial v_y}{\partial y}\right), \tag{3}$$

где x, y – декартовы координаты, $\mathbf{V} = (v_x, v_y)$ – скорость потока и ее проекции на оси x и y; ρ, p – плотность и давление газа; μ – динамический коэффициент вязкости; c_p – удельная теплоемкость при постоянном давлении; λ – коэффициент теплопроводности газа.

На графическом процессоре была реализована часть расчетов для правой (тензорной) части уравнений Навье – Стокса (подпрограмма FLUID). Также реализован ряд промежуточных вычислений для расчета различных коэффициентов и часть граничных условий.

Вторую группу составляют уравнения, описывающие энергетическое состояние неравновесного газа (уравнение сохранения энергии и уравнение, описывающее теплопроводный теплообмен стенки сопла с газом).

$$\rho c_p \frac{\partial T}{\partial t} + \rho c_p \mathbf{V}\operatorname{grad}T = \chi \frac{\partial p}{\partial t} + \\ + \chi \mathbf{V}\operatorname{grad}p + \operatorname{div}(\lambda \operatorname{grad}T) + Q_{kin}, \tag{4}$$

$$\rho_S c_{pS} \frac{\partial T_S}{\partial t} = \operatorname{div}(\lambda_S \operatorname{grad}T_S). \tag{5}$$

В уравнении сохранения энергии (4) Q_{kin} определяет тепловыделение, обусловленное неравновесностью распределения молекул газа по возбужденным состояниям, учитывающим передачу энергии между поступательными и вращательными степенями свободы и колебательными степенями свободы молекул газовой смеси, T – температура газа.

Уравнение (5) описывает теплопроводный перенос тепла в стенке сопла. Здесь ρ_s – плотность стенки; c_{ps} – удельная теплоемкость стенки при постоянном давлении; T_s – температура стенки; λ_s – коэффициент теплопроводности стенки.

На графическом процессоре был реализован расчет коэффициентов для решения уравнения сохранения энергии (подпрограмма TEMPERATURE).

Третью группу составляет система уравнений неравновесной нелокальной кинетики. Система уравнений неравновесной нелокальной кинетики включает в себя уравнение переноса и диффузии газовых компонент с учетом изменения концентрации газовых компонент за счет кинетических процессов.

$$\rho \frac{\partial \mathbf{Y}_i}{\partial t} + \rho \mathbf{V}\operatorname{grad}\mathbf{Y}_i = = \operatorname{div}(\rho D_i \operatorname{grad}\mathbf{Y}_i) + Q_{i,con}, \\ i = 1,..., n. \tag{6}$$

Уравнение (6) описывает неравновесные нелокальные кинетические процессы, протекающие в движущейся в сопле газовой смеси. Здесь D_i – коэффициент диффузии компонент газа, Q_{con} – изменение концентрации газовых компонент $\mathbf{Y_i}$ за счет кинетических процессов.

Уравнения Навье – Стокса и непрерывности, интегрируется с использованием AUSM конечно-разностных схем [8], уравнение сохранения энергии, уравнение теплопроводности для стенки сопла, а также уравнения переноса и диффузии газовых компонент (без учета неравновесных кинетических процессов) интегрировались с использованием неявного метода интегрирования пятиточечной конечно-разностной схемы [5]. Жесткая система уравнений колебательно-нерав-

новесной локальной кинетики решалась с применением метода Гира.

4. ТЕХНОЛОГИЯ CUDA

CUDA [1-4] (Compute Unified Device Architecture) – это архитектура для параллельных вычислений общего назначения, со своей оригинальной моделью параллельного программирования и набором команд для данной архитектуры, которые предназначены для параллельных вычислений на видеокартах компании NVIDIA.

4.1. Модель программирования

Модель программирования CUDA предполагает, что вычислительные потоки CUDA (thread) выполняются на отдельном физическом устройстве (device), которое работает в качестве сопроцессора центрального процессора (CPU). С центрального процессора запускается программа на языке CUDA-C, называемая ядром (kernel), для каждого из вычислительных потоков. Конфигурация потоков задается на CPU перед вызовом функции ядра. Для удобства и наилучшей производительности вычислительные потоки объединяются в блоки (block), а блоки объединяются в сетку (grid).

4.2. Иерархия памяти

CUDA потоки могут обращаться к данным, которые находятся в различных видах памяти. Каждый поток имеет видимую только ему локальную память (local memory) и разделяемую память (shared memory), видимую для всех потоков в блоке, которому он принадлежит. В свою очередь, все потоки имеют доступ к глобальной памяти (global memory). Помимо этого еще существуют константная (constant memory) и текстурная память (texture memory). Программист должен контролировать работу с памятью, аккуратно выбирать нужный тип памяти в зависимости от типа решаемой задачи и не забывать её очищать, когда данные, находящиеся в текущем разделе памяти, уже не нужны.

4.3. Гетерогенное программирование

Структура гетерогенного программирования в данном случае представляет собой последовательное выполнение вычислительных инструкций на CPU и GPU. Схематическая схема гетерогенного программирования представлена на рис. 1.

В данном случае на рис. 1 выполняется следующая последовательность действий:

1) Действия этого этапа происходят на CPU.

На данном шаге идут специальные инструкции для вызова ядра Kernel0 (указывается, что имеется двумерная сетка одномерных блоков размерностью 2 на 3, а в каждом блоке 12 вычислительных потоков), выделение памяти на GPU и перенос данных с CPU на GPU.

2) Действия этого этапа происходят на GPU.

Тут происходит запуск ядра Kernel0 для каждого из 72 вычислительных потоков. То есть для каждого из 72 потоков будет выполняться программа, написанная в ядре Kernel0.

3) Действия этого этапа происходят на CPU.

Происходит передача данных с GPU на CPU (если есть в этом необходимость), идут специальные инструкции для вызова ядра Kernel1 (указывается, что имеется двумерная сетка одномерных блоков размерностью 3 на 2, а в каждом блоке 12 вычислительных потоков) и идет передача данных с CPU на GPU (если есть в этом необходимость).

4) Действия этого этапа происходят на GPU.

Происходит запуск ядра Kernel1 для каждого из 72 вычислительных потоков.

Модель программирования CUDA допускает асинхронное выполнение определенных инструкций. К примеру, в вышеописанном случае возможен вариант с параллельным выполнением ядра Kernel1 и передачей данных с GPU и CPU[1].

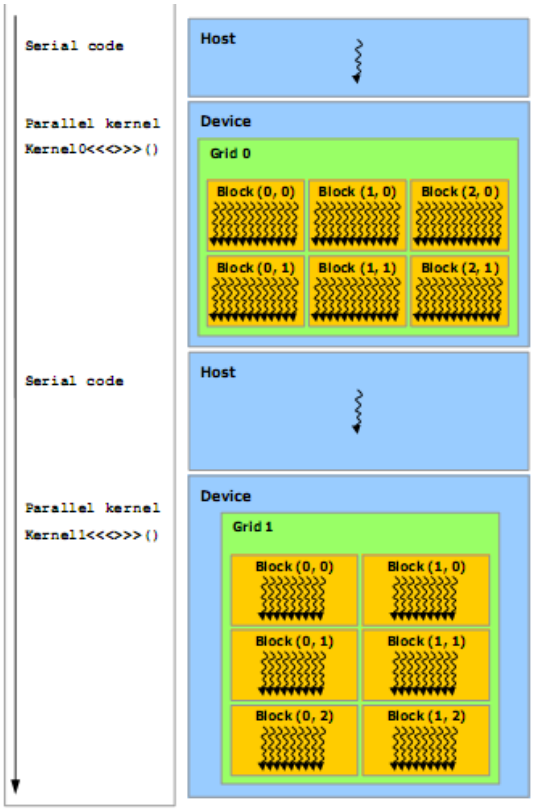

Рис. 1. Гетерогенное программирование

5. МЕТОДОЛОГИЯ РАСПАРАЛЛЕЛИВАНИЯ С ПРИМЕНЕНИЕМ ТЕХНОЛОГИИ CUDA

В начале было проведено определение узких мест в программном коде. Для этого использовались специализированные программ профайлеры (AQtime, CUDA Visual Profiler), а также так называемый метод логического анализа программного кода.

В результате анализа было определено, что много времени уходит на расчеты подпрограммы предназначенной для решения уравнений газовой динамики и подпрограммы предназначенной для решения уравнения сохранения энергии. Выяснилось, что эти подпрограммы вызываются довольно часто, и с увеличением шага по времени для моделирования количество их вызов только растет.

После этого было проведено разбиение данных узких мест на части с учетом особенностей технологии CUDA.

Были разработаны программные модули для запуска вычислений на GPU. Эти модули вызываются из главной фортрановской части программы в виде внешних статических библиотек.

При разработке модулей в основном использовалась глобальная память. При запуске расчетов в глобальную память графического процессора были скопированы массивы, которые не менялись на протяжении вычислений. Далее в зависимости от надобности, для определенных CUDA ядер подгружались нужные данные в глобальную или разделяемую память GPU, либо уже расчетные данные обратно на CPU. Причем одной из основных целей было минимизировать эти передачи и добиться эффекта объединения запросов в одну транзакцию при доступе в глобальную память GPU. В ходе программирования возникла проблема с уплотнением сетки, так как большая часть данных была определена в виде фортрановских common блоков, приходилось учитывать ограничение на максимальный размер common блоков. Из-за этого максимальный размер сетки на котором удалось провести моделирование равен 1536×128 узлов.

6. АНАЛИЗ РЕЗУЛЬТАТОВ

В ходе выполнения работы были использованы следующие программная и аппаратная среды:

- Windows 7 Professional, Intel(R) Visual Fortran 11, Visual Studio 2008 SP1, CUDA 4.0;
- Intel Core(TM)2 DUO CPU E8400 @ 3GH, 4ГБ ОЗУ, NVIDIA GeForce GTX260.

Вычисления на CPU проводились в режиме максимальной оптимизации кода (нет проверки на границы массивов, в свойствах компилятора указан режим максимальной скорости выполнения и т. д.).

Сравнения проводились на различных сетках с количеством узлов: 512×128, 1024×128, 1536×128. При увеличении размерности сетки расчеты на GPU идут быстрее, чем на CPU. Таким образом, чем массивней вычисления, тем графический процессор считает быстрее по сравнению с центральным процессором. В результате сравнения времени счета выяснилось, что на сетке размером 1536×128, большая часть расчетных модулей считается на GPU в 2÷5 раз быстрее, чем на CPU (модули для расчета уравнений Навье – Стокса), на сетке размером 1024×128 время расчетов примерно одинаково, на сетке размером 512×128 на CPU расчеты идут быстрее.

В Табл. 1 представлено сравнение расчетного времени выполнения отдельных подпрограмм для CPU и GPU, с учетом передач данных между CPU и GPU, вариант с двойной точностью и оптимизированными передачами данных, без перетасовки массивов.

В подпрограмме Forcetenzor происходит расчет правой (тензорной) части уравнений Навье – Стокса, подпрограмме COEF идет расчет коэффициентов для промежуточных прогонок (A, B, C, F и т.д.), подпрограммах Func_VX, Func_VR, Func_RO происходит расчет некоторых промежуточных массивов для расчетов Coef, подпрограммах Calcul_vx, Calcul_vr, Calcul_ro происходит часть расчетов плотности и проекций скорости потока на оси X и Y.

Время выполнения вычислений на CPU и GPU за один временной шаг программы с учетом передачи данных представленных в Табл. 2. Как и в случае со временем выполнения отдельных модулей, вычисления на GPU идут быстрее с ростом размерности сетки, но результаты не такие впечатляющие, как представленные ранее. Это обуславливается тем, что в настоящее время далеко не все вычисления перенесены на GPU

Таблица 1

Сравнение расчетного времени выполнения отдельных подпрограмм для CPU и GPU с учетом передач данных между CPU и GPU, вариант с двойной точностью и оптимизированными передачами данных без перетасовки массивов

Название модуля	Размерность сетки	Время выполнения модуля, мс	
		GPU	CPU
Forcetenzor	256×128	0.12	0.62
COEF	256×128	0.05	0.16
Func_VX	256×128	0.16	0.77
Func_VR	256×128	0.03	0.12
Func_RO	256×128	0.14	0.13
Calcul_vx	256×128	0.05	0.03
Calcul_vr	256×128	0.05	0.03
Calcul_ro	256×128	0.03	0.02
Forcetenzor	1024×128	0.45	2.81
COEF	1024×128	0.49	3.27
Func_VX	1024×128	0.13	0.67
Func_VR	1024×128	0.13	0.66
Func_RO	1024×128	0.06	0.28
Calcul_vx	1024×128	0.15	0.08
Calcul_vr	1024×128	0.15	0.11
Calcul_ro	1024×128	0.11	0.08
Forcetenzor	1536×128	0.65	4.26
COEF	1536×128	0.69	5.06
Func_VX	1536×128	0.2	0.94
Func_VR	1536×128	0.21	0.84
Func_RO	1536×128	0.1	0.47
Calcul_vx	1536×128	0.25	0.14
Calcul_vr	1536×128	0.23	0.16
Calcul_ro	1536×128	0.16	0.11

Примечания. Для сетки (256×128) размер блоков был 8 потоков по каждому направлению, а для сеток (1024×128) и (1536×128) размер блоков был 16 потоков по каждому направлению.

Таблица 2

Сравнение расчетного времени выполнения вычислений на CPU и GPU за один временной шаг программы с учетом передачи данных

Размерность сетки	Режим расчетов, мин		Полученное ускорение, %
	CPU	GPU	
512×128	14	14÷15	0
1024×128	27÷28	24÷26	4÷11
1536×128	50÷51	42÷43	14÷17

ЗАКЛЮЧЕНИЕ

Исходя из полученных результатов по времени расчетов отдельных модулей, можно отметить следующее. Технология CUDA дает ускорение расчетов в случае громоздких вычислений с минимумом ветвлений и передач данных между CPU и GPU. Так как на

данный момент времени не все массивные вычисления перенесены на GPU, можно ожидать более существенный прирост скорости вычислений. Но наряду с достоинствами технология CUDA имеет свои недостатки. Среди них надо отметить следующие:

1). Сложность разработки крупных приложений, написанных на разных языках программирования (хотя компания NVIDIA и выпустила надстройку на Visual Studio для отладки CUDA приложений – Parallel Nsight, в случае вышеописанных приложений имеется много проблем).

2). Для получения ускорения вычислений надо аккуратно использовать особенности технологии, а в случае реальных физических приложений это сделать не просто, особенно когда речь заходит о граничных условиях и атомарных операциях.

3). Проблема скорости вычислений операций с двойной точностью. Так как GPU предназначены в основном для обработки графики, а для этого повышенная точность вычислений не требуется. А значит, количество процессоров, поддерживающее вычисления с двойной точностью, гораздо меньше, чем с одинарной (Компания NVIDIA выпускает линейку GPU - Tesla, специально предназначенных для массивных вычислений общего назначения, поэтому можно ожидать, что в будущем эта проблема будет не так актуальна).

СПИСОК ОБОЗНАЧЕНИЙ

CUDA (Compute Unified Device Architecture) – технология от компании NVidia, предназначенная для разработки приложений для массивно-параллельных вычислительных устройств;

CPU (Central Processing Unit) – центральный процессор (ЦП, ЦПУ), часть компьютера, непосредственно выполняющая машинные команды, из которых состоят программы;

GPU (Graphics Processing Unit) – графический процессор (ГП, ГПУ);

Thread – поток вычислений (Поток, нить);

Kernel – модуль, написанный на языке CUDA-C, который выполняется на GPU, а вызывается либо с CPU, либо с GPU;

Block – структура для объединения потоков вычислений, для более выгодного использования технологии CUDA;

Grid – структура для объединения потоков вычислений, для более выгодного использования технологии CUDA;

x, y – декартовы координаты, $\mathbf{V} = (v_x, v_y)$ – скорость потока и ее проекции на оси x и y;

ρ, p – плотность и давление газа;

μ – динамический коэффициент вязкости;

c_p – удельная теплоемкость при постоянном давлении;

T – температура газа;

λ – коэффициент теплопроводности;

Q_{kin} – тепловыделение, обусловленное неравновесностью распределения молекул газа по возбужденным состояниям, учитывает передачу энергии между поступательными, вращательными степенями свободы и колебательными степенями свободы молекул газовой смеси;

c_{ps} – удельная теплоемкость стенки при постоянном давлении;

T_s – температура стенки;

λ_s – коэффициент теплопроводности стенки;

Y_i i –концентрация компоненты газа, n - количество компонента газа;

$Q_{i,con}$ – член, описывающий изменение концентрации i – компоненты газа в столкновительных процессах;

D_i – коэффициент диффузии i – компоненты газа.

СПИСОК ЛИТЕРАТУРЫ

1. NVIDIA CUDA C Programming Guide (Version 4.0)
2. CUDA C Best Practices Guide (Version 4.0)
3. Sanders Jason, Kandrot Edward - CUDA by Example An Introduction to General-Purpose GPU Programming, 2010. 290 с.
4. Боресков А. В., Харламов А. А. Основы работы с технологией CUDA- М.: ДМК, 2010. 230 с.
5. Самарский А.А., Попов Ю.П. Разностные методы решения задач газовой динамики. М.: Наука, 1980. 352 с.
6. Алексеева И.В., Будник А.П. Двумерная модель неравновесных кинетических процессов в многокомпонентных газовых смесях при сверхзвуковом движении // Сборник научных трудов, РАН, Отделение энергетики, машиностроения, механики и процессов управления, ИПМ им. А.Ю. Ишлинского РАН, Москва, 2010. С. 5–12
7. Алексеева И.В., Будник А.П. Двумерная модель неравновесных кинетических процессов в многокомпонентных газовых смесях при сверхзвуковом движении. // Физико-химическая кинетика в газовой динамике. 2011. Том 1.
8. Edwards, J.R., Liou, M.-S. Low-Diffusion Flux-Splitting Methods for Flow at all Speeds // AIAA Journal. 1998. Vol.36. № 9. pp.1610–1617.