Сравнение производительности библиотек Hypre и NVIDIA AmgX на задаче моделирования дыма

М.А. Кривов[1], А.И. Новиков[1], А.А. Юданов[2]

[1] МГУ им. М.В. Ломоносова, г. Москва, Россия
[2] Puppetworks Animation Studio, г. Будапешт, Венгрия

Аннотация. В работе рассмотрена задача обтекания группы объектов потоком несжимаемой жидкости (дыма), пришедшая из компьютерной графики и сводящаяся к решению трёхмерной системы уравнений Эйлера. Авторами предложен механизм для оценки допустимой погрешности при выполнении проекционного этапа, заключающегося в численном решении уравнения Пуассона в смешанной постановке на структурированной сетке. Для выполнения данного этапа были использованы сторонние библиотеки Hypre (CPU) и NVIDIA AmgX (GPU), реализующие алгоритмы решения СЛАУ с разреженными матрицами. Проведено сравнение полученных реализаций при варьировании численных методов, типов сцен и размера сеток на системе с процессором Xeon E5-2697v3 и ускорителем Tesla K40. Показано, что, в зависимости от специфики сцены, полезная производительность сильно изменяется, в результате чего вопросы выбора предпочтительной библиотеки, аппаратной платформы и алгоритма оказываются достаточно неоднозначными. В частности, ускорение библиотеки для графического ускорителя относительно её аналога для центрального процессора находилось в диапазоне от 1.2 до 273 раз.

Ключевые слова: математическое моделирование, визуализация, дым, многосеточный метод, стабилизированный метод бисопряжённых градиентов, CPU, GPU, Hypre, NVIDIA AmgX.

DOI 10.14357/20718632180202

Введение

Необходимость решения идентичных уравнений математической физики может возникнуть в совершенно разных и на первый взгляд никак не связанных предметных областях, например, таких как аэрогидродинамика, обработка изображений или медицинская физика. Во всех случаях исходная задача после формализации математической модели сводится к конкретному уравнению в частных производных, для решения которого может быть использован один из множества хорошо известных и неоднократно апробированных численных алгоритмов, достаточно часто реализованных в виде готовых библиотек.

С другой стороны, полностью исключить из рассмотрения специфику первоначальной задачи удаётся далеко не всегда, поэтому при выборе того или иного метода также необходимо учитывать привязку к предметной области. Даже не смотря на тот факт, что после введения сетки и разностной схемы исходное уравнение в частных производных можно переписать в виде системы линейных алгебраических уравнений, один и тот же подход к решению этого СЛАУ в зависимости от исходной задачи может оказаться как предпочтительным, так и просто не применимым. Объясняется это тем, что матрица данного СЛАУ часто имеет специфическую структуру и разную степень разреженности, и при выборе семейства алгорит-

мов эти особенности могут сыграть важную роль. В частности, из-за этого в гидродинамике популярностью пользуются алгоритмы, основанные на методе сопряжённых градиентов, в то время как в компьютерной графике предпочтение обычно отдают многосеточным подходам.

В работе рассмотрена одна из актуальных проблем современной компьютерной графики, которая заключается в моделировании и последующей визуализации процесса обтекания композиции из произвольных объектов потоком жидкости или газа, в частности, дымом. Востребованность данной задачи можно объяснить тем, что при производстве фильмов и всевозможных видеоматериалов активно используются спецэффекты, часть из которых как раз основана на моделировании жидкой или газообразной среды. На данный момент вторая часть проблемы, а именно визуализация, уже успешно решается существующими программными инструментами, в то время как процесс моделирования среды в рамках первого этапа имеет ряд особенностей, в ряде случаев затрудняющих использование стандартных решений из популярных пакетов типа Houdini и FumeFX.

Действительно, вычислительные ресурсы, обычно представленные центральными процессорами (также обозначаемыми как CPU, Central Processing Unit) в подавляющем большинстве случаев ограничены. Как следствие, предпринимаются попытки максимально задействовать вспомогательные и более производительные графические ускорители (GPU, Graphics Processing Unit) и перенести на них не только предварительный рендеринг трёхмерных объектов, но и сами расчёты, а это, к сожалению, позволяют сделать далеко не все пакеты. Более того, почти все расширения для указанных инструментов ориентированы на вычислители с общей памятью, что накладывает дополнительные ограничения на размер сцены. Как показывает опыт авторов, следствием этих двух проблем является потребность студий в специализированной библиотеке, способной проводить решение некоторых задач математической физики на действительно больших сетках с использованием графических ускорителей и учётом ряда особенностей предметной области.

В рамках данной работы было проведено исследование, какую производительность можно достичь при использовании универсальных библиотек Hypre (CPU) и NVIDIA AmgX (GPU) при решении рассмотренной проблемы обтекания группы объектов дымом. Благодаря учёту особенностей задачи представляется возможным заранее определить требуемую точность, поэтому для выбранных тестовых сцен был рассмотрен такой критерий, как полезная производительность в зависимости от заданного порога погрешности. Все тесты проводились на одном вычислительном узле, оснащённом четырнадцатиядерным центральным процессором Intel Xeon E5-2697 v3 и ускорителем NVIDIA Tesla K40.

1. История вопроса

Эффекты, схожие по своей природе с дымом, применяются в компьютерной графике достаточно давно. В роли пионеров, в частности, выступили Дж. Гарднер, который в работе [1] одним из первых предложил математическую модель для моделирования и визуализации разных типов облаков, а также У. Ривз, описавший в статье [2] свой опыт использования систем частиц для создания спецэффектов в фильмах.

Дальнейшее развитие в основном шло за счёт использования лагранжевых подходов, когда дым представлялся как система частиц, каждая из которых имеет собственное значение некоего параметра и взаимодействует с другими. Большая часть подобных алгоритмов была реализована в компьютерных играх, а также в виде примеров использования аппаратных технологий рендеринга, поэтому не получила заметного отражения в публикациях. Однако есть и научные работы, в которых данный подход активно применялся — например, в статье [3] лагранжевая модель используется для повышения производительности в тех случаях, когда качество визуализации дыма не так важно, как скорость отрисовки кадров.

В рамках другой группы работ предпринимались попытки адаптировать классические сеточные методы аэрогидродинамики для решения озвученной проблемы, в том числе с использованием эйлеровой модели среды. Впервые эта идея получила воплощение в ста-

тье Н. Фостера [4], содержащей детальное описание алгоритма моделирования воды на основе решения уравнений Навье-Стокса. В дальнейшем при участии Дж. Стэма была опубликована серия работ [5, 6], в которой при визуализации дыма среда также описывалась с помощью трёхмерных уравнений Эйлера и Навье-Стокса. Этот алгоритм уже другими авторами был адаптирован для работы в режиме реального времени в двумерном [7] и трёхмерном [8] случаях, причём требуемый уровень производительности обеспечивался за счёт переноса вычислений на графический ускоритель, что на тот момент (2004 и 2007 года) было нетривиальной задачей.

Отдельным вопросом является поиск механизма, позволяющего воспроизвести завихрения дыма, вызываемые турбулентностью и теряемые при использовании достаточно грубых сеток. Помимо использования искусственного шума, прибавляемого к функции силы [6], были предложены и более совершенные модели, адаптированные специально под данную задачу [9]. Также стоит отметить работу [10], в которой для воспроизведения деталей дыма используется модель турбулентности $k-\varepsilon$.

2. Используемый подход

В данной работе за основу взят алгоритм, описанный в статье [6]. Так как нашей целью является оценка применимости к рассматриваемой задаче универсальных библиотек, то ниже приведено схематическое описание исходного метода без каких-либо изменений в его сути. При этом стоит отметить, что за прошедшие 15 лет в рамках работ других авторов были предложены уточнения для данного алгоритма, заметно улучшающие визуальные результаты. Однако этап с построением СЛАУ не претерпел каких-либо изменений, поэтому использование несколько устаревшей записи метода является вполне корректным, так как в рамках данной работы она носит иллюстративный характер.

2.1. Постановка задачи

Моделируемая среда рассматривается как невязкая и несжимаемая. Данные упрощения были сделаны с учётом не столько специфики задачи, сколько особенностей численного алгоритма. При моделировании жидкостей описываемый подход обеспечивает приемлемые в визуальном плане результаты только на достаточно мелких сетках, что делает его малоприменимым из-за ограниченности вычислительных ресурсов. В случае газов данная особенность пропадает и, таким образом, появляется возможность работы и на грубых сетках, поэтому развитие рассматриваемого семейства методов шло в основном в привязке к невязким средам. Более того, скорость протекания моделируемых процессов, в частности распространения дыма, практически всегда оказывается существенно меньше скорости звука, поэтому сжимаемостью газа также можно пренебречь. Как следствие, рассматриваемый процесс описывается следующей системой трёхмерных уравнений Эйлера:

$$\nabla \cdot \vec{U} = 0$$

$$\frac{\partial \vec{U}}{\partial t} = -(\vec{U} \cdot \nabla)\vec{U} - \nabla p + \vec{f},$$

где $\vec{U}$ и p – искомые вектор скорости и давление, а $\vec{f}$– известная внешняя сила, например, гравитационная. Так как плотность считается константной и, в данном случае, равной единице, то в рассматриваемой системе имеется четыре неизвестных и четыре уравнения, и, таким образом, для наличия решения осталось задать начальные и граничные условия, например, следующим образом:

$$0 \le x \le R, 0 \le y \le R, 0 \le z \le R$$

$$(x,y,z) \in G_{\text{ext}}\colon \frac{\partial \vec{U}}{\partial \vec{n}} = 0, \frac{p}{\partial \vec{n}} = 0$$

$$(x,y,z) \in G_{\text{int}}\colon \vec{U} \cdot \vec{n} = 0, \frac{p}{\partial \vec{n}} = 0$$

$$\vec{U}(x,y,z,t=0) = \overrightarrow{U_0}(x,y,z)$$

$$p(x,y,z,t=0) = p_0(x,y,z)$$

В данном случае расчёты будут проводиться в кубической области с внешней границей G_{ext}, на которой задано естественное условие, поэтому вещество может спокойно покидать рассматриваемый регион. Внутри имеются соответствующие объектам выколотые подобласти с общей внутренней границей G_{int}, на которой задано условие непротекания — газ не может проникать внутрь данных объектов. Начальное распределение искомых величин задаётся аниматором исходя из того, что за эффект он хочет воспроизвести, и в общем случае представляет-

ся заранее известными функциями $\overrightarrow{U_0}$ и p_0, которые должны быть согласованы с граничными условиями.

Данная задача является основой для моделирования различных эффектов, таких как дым, огонь, облака или абстрактные «магические» потоки. Чтобы обеспечить моделирование именно дыма, в решаемую систему необходимо добавить ещё две неизвестные скалярные величины — плотность ρ и температуру T дыма, который перемешивается и свободно распространяется вместе с основным газом. Именно эти величины в дальнейшем будут визуализироваться, в то время как газ считается бесцветным. Чтобы описать эту смесь из двух веществ, во введённую ранее систему добавляются два дополнительных уравнения адвекции:

$$\frac{\partial T}{\partial t} = -(\vec{U} \cdot \nabla)T$$
$$\frac{\partial \rho}{\partial t} = -(\vec{U} \cdot \nabla)\rho$$

Которые описывают процесс перемещения вещества в заданном поле скоростей и, как следствие, изменения его концентрации. После этого осталось дополнительно задать начальное распределение температуры и плотности дыма через также предоставленные аниматором функции:

$$T(x, y, z, t = 0) = T_0(x, y, z)$$
$$\rho(x, y, z, t = 0) = \rho_0(x, y, z)$$

И ввести обратную связь, чтобы тяжёлые клубы дыма опускались под действием силы тяжести, а потоки горячего воздуха восходили вверх. Для этого достаточно расписать функцию внешней силы следующим образом:

$$f = -\alpha\rho z_{up} + \beta(T - T_{amb})z_{up},$$

где z_{up} – направление, определяющее понятие «вверх», T_{amb} – температура окружающей среды, а α и β – положительные константы, определяющие интенсивность процесса распространения дыма и также задаваемые аниматором. Решив данную задачу для отрезка времени $0 \le t \le T$, мы получим данные, достаточные для наложения требуемого спецэффекта.

2.2. Численный метод

Прежде, чем переходить к описанию численного метода, стоит подчеркнуть, что задача заключается в поиске приближённого решения, в ряде случаев, весьма грубого. Это связано с тем, что результатом моделирования должно быть лишь изображение, правдоподобность которого у человека не вызывает сомнений. Поэтому в рассматриваемой статье использовался неконсервативный, зато абсолютно устойчивый подход, описание которого дано ниже.

Для решения введённой ранее системы уравнений предлагается задействовать вариацию метода расщепления, в рамках которого для перехода с i-ого на $i + 1$ временной слой должны быть выполнены три действия. Во-первых, исключив давление из уравнения сохранения момента импульса, необходимо вычислить предварительное значение вектора скорости $\vec{U}^*$:

$$\frac{\vec{U}^* - \vec{U}^i}{\Delta t} = -(\vec{U}^i \cdot \nabla)\vec{U}^i + \vec{f}^i$$

Во-вторых, требуется обеспечить несжимаемость газа, для чего вычисляется актуальное значение давления путём решения уравнения Пуассона:

$$\nabla^2 p^{i+1} = \frac{1}{\Delta t}\nabla \cdot \vec{U}^*$$

А в предварительный вектор скорости вносится соответствующая поправка:

$$\vec{U}^{i+1} = \vec{U}^* - \Delta t \nabla p^{i+1}$$

В-третьих, необходимо найти новые значения для плотности и температуры дыма:

$$\frac{T^{i+1} - T^i}{\Delta t} = -(\vec{U}^i \cdot \nabla)T^i$$
$$\frac{\rho^{i+1} - \rho^i}{\Delta t} = -(\vec{U}^i \cdot \nabla)\rho^i$$

Отдельно стоит остановиться на схеме численного решения уравнений адвекции, которые нужны для вычисления величин $\vec{U}^*$, T^{i+1} и ρ^{i+1}. Если для этого использовать аппроксимацию конечной разностью, как это сделано в выкладках выше, то не удастся обеспечить абсолютную устойчивость метода, а наличие данного свойства является крайне важным в силу ограниченности вычислительных мощностей. Поэтому в рассматриваемом алгоритме используется полулагранжевый подход, в рамках которого на этапе решения уравнений адвекции — и только на нём — жидкость представляется набором частиц, расположенных в центрах ячеек и обладающих соответствующими характеристиками.

Например, в случае температуры это означает, что для узла с номером (i,j,l) требуется вычислить величину $\vec{v} = (vx, vy, vz) = \frac{\Delta t \cdot \vec{U}^n_{i,j,l}}{\Delta h}$, где Δh — шаг по пространству. После этого значение на следующем временном слое будет определено как $T^{i+1}_{i,j,l} = T^{i}_{i-vx,j-vy,l-lz}$. Очевидно, что компоненты вектора $\vec{v}$ практически всегда будут дробными, поэтому для вычисления искомой величины температуры необходимо провести интерполяцию значений из восьми ячеек сетки, расстояние которых до узла $(i - vx, j - vy, l - lz)$ не превышает $\frac{\Delta h}{2}$. Если же индексы выходят за пределы расчётной области, то соответствующее значение определяется исходя из граничных условий.

Как легко заметить, в данном методе единственной вычислительно-ёмкой операцией является проекционный этап, на котором для каждого временного слоя требуется составить и решить уравнение Пуассона. Поэтому в настоящей работе рассматривается возможность его выполнения путём построения соответствующего СЛАУ с последующим его решением через библиотеки Hypre и NVIDIA AmgX. Результаты тестирования их производительности при данном сценарии использования будут приведены в следующих разделах.

3. Оценка допустимой погрешности

Как отмечалось ранее, в качестве результата моделирования может выступать не физически точное решение, а любое, даже достаточно грубое приближение, искусственность которого человек не в состоянии отличить визуально. Благодаря этому появляется возможность гибко настроить порог точности для итерационных методов инвертирования СЛАУ $Ax = b$, при достижении которого процесс расчётов останавливается, а найденное решение считается подходящим. Напомним, что в численных методах обычно используются следующие метрики для задания подобной погрешности на i-ой итерации:

$$\varepsilon_i^{(1)} = \| x_i - x_{i-1} \|$$
$$\varepsilon_i^{(2)} = \|Ax_i - b\|$$
$$\varepsilon_i^{(3)} = \| x_i - x \|$$

Значение $\varepsilon_i^{(1)}$ позволяет с минимальными затратами оценить сходимость итерационного процесса и завершить расчёты в том случае, когда дополнительные итерации больше не приближают нас к искомому результату. Но, к сожалению, найденное таким образом решение может отличаться от аналитического весьма существенно. Величина $\varepsilon_i^{(2)}$, также называемая невязкой, показывает, насколько близко мы приблизились к точному решению дискретной задачи, но при этом она не учитывает погрешность, вносимую использованием разностной схемы и дискретизацией геометрии объектов. В результате чего её нулевое значение также не гарантирует, что найденный вектор x_i совпадает с непрерывным решением до последнего знака после запятой. Наконец, используя метрику $\varepsilon_i^{(3)}$, можно оценить реальную погрешность путём сопоставления численного решения с эталонным. Что полезно при разработке алгоритма и отладке его программной реализации, но, очевидно, неприменимо для исходной задачи, аналитическое решение которой не только неизвестно, но может не существовать в принципе.

В работах предшественников данному вопросу, а именно выбору критерия точности расчётов, не уделялось достаточно внимания в первую очередь из-за ограниченности вычислительных ресурсов. В статье [7] при решении двумерного уравнения Пуассона использовался метод Якоби, в котором выполнялось от 20 до 50 итераций, а реальная погрешность не оценивалась. Аналогично поступили и авторы работы [8], также выбрав метод Якоби и фиксированное число итераций, но уже для трёхмерного случая. При этом они дополнительно изучили влияние погрешности на итоговый результат, показав на одном примере, что при переходе от 20 к 1000 итераций проявляются дополнительные детали дыма, однако общая структура остаётся идентичной. В статье [6], авторы которой проводили вычисления уже не в режиме реального времени, для решения СЛАУ был задействован метод сопряжённых градиентов с неполным предобуславливателем Холеского. Но, опять же, допустимый порог погрешности определялся исключительно визуально, а кри-

терием завершения служило выполнение 20 итераций выбранного метода.

Так как скорость сходимости итерационного процесса зависит не только от алгоритма, но и множества других факторов, таких как размер сетки, геометрия объектов и, хоть и в меньшей степени, от граничных условий, то идея определения требуемого числа итераций априори является не совсем корректной. Авторами данной работы была использована приведённая ниже схема, позволяющая примерно оценить требуемую точность. Пусть Δh и Δt – шаги по пространству и по времени. Так как используемый подход к решению исходной системы уравнений является безусловно устойчивым, то на них не накладывается какое-либо ограничение и их значения могут быть произвольными. Далее, допустим, что численная погрешность вносится только на n-ом шаге по времени и только в результате решения уравнения Пуассона. В таком случае она проявится в уравнении адвекции, благодаря чему мы получаем возможность ввести для неё ограничение $| \bar{U}^n_{i,j,l} \cdot \Delta t - U^n_{i,j,l} \cdot \Delta t | < \Delta h$, где $\bar{U}^n_{i,j,l}$ и $U^n_{i,j,l}$ – значения одной компоненты скорости в узле сетки (i, j, l), а подчёркивание означает величину с внесённой погрешностью. Суть данного ограничения сводится к тому, что при решении уравнения адвекции полулагранжевым методом частица газа будет попадать либо в ту же самую, либо в соседнюю ячейку. И, таким образом, вклад погрешности окажется минимальным.

Если расписать значение скорости, а производную давления заменить центральной разностью, то ограничение сводится к более полезной форме $\left|\frac{\bar{\Phi}^n_{i+1,j,l}-\bar{\Phi}^n_{i-1,j,l}}{2\Delta h} - \frac{\Phi^n_{i+1,j,l}-\Phi^n_{i-1,j,l}}{2\Delta h}\right| < \frac{\Delta h}{\Delta t^2}$, где $\Phi^n_{i,j,l}$ – значения потенциала, которыми являются перенумерованные компоненты получаемого при решении СЛАУ вектора x. Применяя неравенство треугольника и используя непрерывную норму, получаем $\left|(\bar{\Phi}^n_{i+1,j,l} - \Phi^n_{i+1,j,l}) - (\bar{\Phi}^n_{i-1,j,l} - \Phi^n_{i-1,j,l})\right| < 2\left\|\bar{\Phi}^n - \Phi^n\right\|_C$, в результате чего мы можем перейти к порогу погрешности $\varepsilon^{(3)} < \frac{\Delta h^2}{\Delta t^2}$.

Данная оценка не гарантирует, что погрешность не скажется на результатах, но позволяет понять, при каких значениях её влияние будет минимальным. Следующий этап заключается в поиске взаимосвязи между порогом $\varepsilon^{(3)}$ по норме C, критерий для которого мы вывели, и $\varepsilon^{(2)}$ по норме L^2, значения для которого возвращают библиотеки Hypre и NVIDIA AmgX. Аналитически определить данную зависимость не представляется возможным из-за необходимости учёта объектов произвольной формы, поэтому в данной работе связь двух метрик была оценена опытным путём.

На Рис. 1 приведено сопоставление этих двух величин при решении исходной смешанной задачи для уравнения Пуассона в случае кубической сцены с одной сферой радиуса 0.25 от размера области на сетке из 350x350x350 ячеек. Дополнительно к этому расчёты проводились при варьировании радиуса сферы (зна-

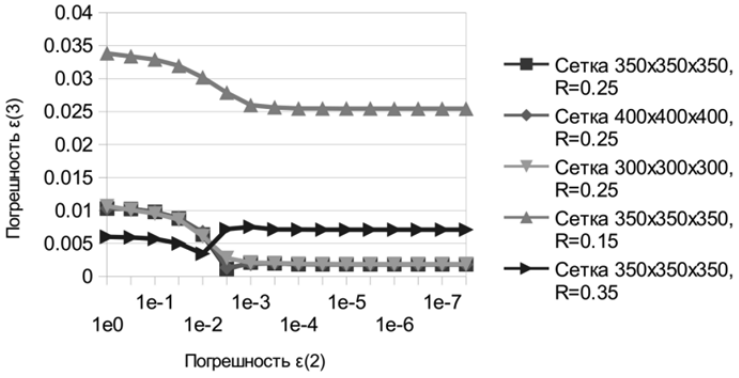

Рис. 1. Зависимость реальной погрешности $\varepsilon^{(3)}$ от невязки $\varepsilon^{(2)}$ на разных сетках

чения 0.15 и 0.35) и размера сетки (случаи 300x300x300 и 400x400x400 ячеек). Разность между максимальным и минимальным значениями потенциала была подобрана таким образом, чтобы при нулевом приближении невязка равнялась единице.

Полученная взаимосвязь верна только для рассмотренного частного случая, однако может быть использована как отправная точка при задании начального порога точности, который в дальнейшем будет корректироваться аниматором в зависимости от качества визуализации и времени расчётов. В качестве примера можно рассмотреть следующий случай — пусть требуется отрисовать сцену обтекания объекта потоком дыма, где моделируемая область задана как куб со стороной один метр, шаг по времени определяется частотой кадров и равен 1/30 секунды, а сетка имеет размер 400x400x400 ячеек. Таким образом, требуемая реальная погрешность $\varepsilon^{(3)}$ должна быть не более чем $\frac{(1/400)^2}{(1/30)^2} = 5.625 \cdot 10^{-3}$, и, пользуясь зависимостью, изображённой на рис. 1, порог для величины $\varepsilon^{(2)}$ можно грубо оценить как 10^{-2}.

Также стоит сказать несколько слов о природе величины $\varepsilon^{(3)}$, которую можно рассматривать как сумму трёх компонент — итерационной погрешности метода, ошибки от дискретизации области и ограничений машинной точности. На Рис. 1 видно, что с некоторого момента повышение точности в понимании $\varepsilon^{(2)}$ больше не приближает нас к эталонному решению, что объясняется достижением порога, при котором начинает превалировать ошибка дискретизации объектов. Если посмотреть на Рис. 2, на котором для рассматриваемого случая приведён разрез

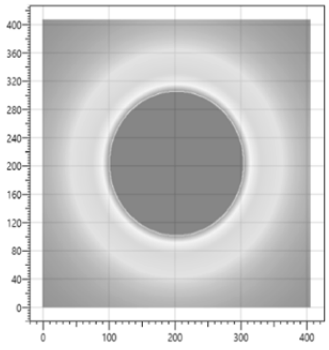

Рис. 2. Распределение реальной погрешности в срезе z=R/2, сетка 410x410x410, радиус сферы - R/4

области по измерению Z, то легко заметить, что максимальные значения погрешности находятся только на границах. Таким образом, предложенная схема оценки допустимой погрешности в некотором смысле является излишне строгой, и на практике может использоваться с понижающим коэффициентом.

4. Оценка производительности

Изначально авторами был разработан прототип расширения для пакета Houdini, в котором с помощью описанного алгоритма производилось моделирование процесса распространения дыма, и результаты работы которого в качестве иллюстрации приведены на Рис. 3.

Разработанная программа способна проводить расчёты на графических ускорителях, в частности, приведённ ый на Рис. 3 пример получен с использованием вычислителя NVIDIA GeForce 680, а расчёты, требуемые для подготовки данного 15-секундного видео, длились около 8 часов. Однако она обладает рядом су-

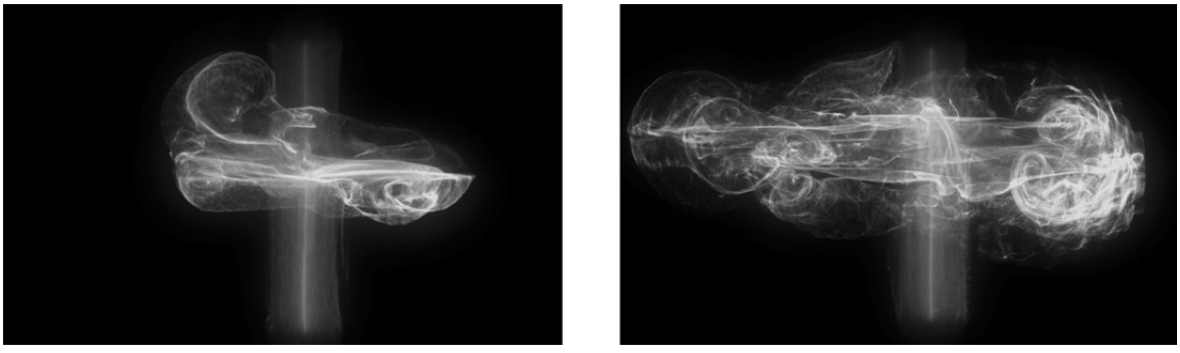

Рис. 3. Пример моделирования соударения двух потоков дыма, моменты времени t = 2.0 и t = 9.0 секунд

щественных недостатков, в первую очередь обусловленных использованием собственного и достаточно примитивного решателя СЛАУ. Поэтому, прежде чем продолжать его развитие, было решено оценить возможности и производительность существующих библиотек, чему и посвящён настоящий раздел.

4.1. План тестирования

Для тестирования использовалась система, технические характеристики которой приведены в Табл. 1.

Таким образом, по пиковой производительности графический ускоритель был быстрее центрального процессора в 14.6 и 9.8 раза при расчётах в двойной и одинарной точности соответственно. Однако в задачах данного класса узким местом зачастую становится работа с памятью, поэтому имеет смысл также сравнивать и этот показатель, выигрыш по которому составлял уже 4.2 раза.

Для проведения моделирования с использованием центрального процессора была задействована библиотека Hypre [11] в версии 2.11.1-67, которая является открытым и достаточно функциональным пакетом почти с двадцатилетней историей и в которой содержатся параллельные реализации разных алгоритмов, адаптированных для проведения масштабных расчётов на структурированных и неструктурированных сетках. Для делегирования вычислений на графический ускоритель использовалась библиотека NVIDIA AmgX [12, 13] версии 1.2.0, которая является платной, но доступной для некоммерческого использования. Хотя в названии пакета содержится прямое указание на многосеточные методы, в нём также имеются реализации и иных типов алгоритмов, а акцент сделан на качественную оптимизацию под ускорители компании NVIDIA.

Для сравнения были выбраны два алгоритма — алгебраический многосеточный метод (в литературе обычно обозначаемый как AMG) и стабилизированный метод бисопряжённых градиентов (BICGSTAB), для каждого из которых дополнительно использовался предобуславливатель ILU(0). В смежных работах предпочтение обычно отдают представителям именно этих семейств методов [14, 6], что и обусловило их выбор.

Тестовые матрицы были подготовлены с использованием семиточечной разностной схемы и сохранены в формате CSR. Были рассмотрены три сцены, которые схематично изображены на Рис. 4. В первой из них отсутствуют какие-либо объекты, в то время как во второй и третьей имелись одна крупная сфера и группа объектов соответственно. Размер сеток был подобран таким образом, чтобы полностью задействовать память графического ускорителя. Выяснилось, что в случае алгоритма BICGSTAB имеющихся 12 гигабайт достаточно для проведения расчётов на сетке 350x350x350, в то время как для метода AMG верхним порогом оказалась сетка из 150x150x150 ячеек.

Табл. 1. Характеристики используемых вычислителей

Вычислитель	Число ядер	Тактовая частота, ГГц	Пик. производ., ГФлопс (float / double)	Объём и тип памяти	Пропускная способность памяти, Гбайт/с
Xeon E5-2697v3	14	2.6	292 / 146	64 ГБ, DDR4	68
Tesla K40	2880	0.745	4291 / 1430	12 ГБ, GDDR5	288

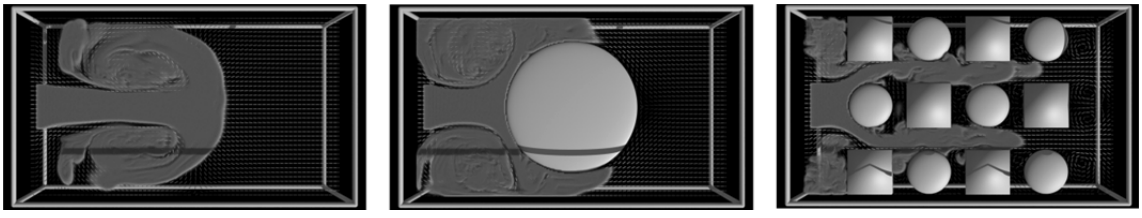

Рис. 4. Визуализация дыма и поля скоростей для рассматриваемых типов сцен

Стоит отметить, что библиотека Hypre поддерживает проведение расчётов только с двойной точностью, которая для рассматриваемой задачи является излишней. Поэтому, чтобы сравнение было объективным, для решателя NVIDIA AmgX тестирование осуществлялось в двух режимах — в *double*, что позволяет его сопоставить с пакетом Hypre, и в *float*, что больше соответствует потребностям проекта. Во всех случаях измерялось время, необходимое для достижения заданной невязки по норме L2, которая в свою очередь варьировалась от $10^{-0.5}$ до 10^{-7} с шагом $10^{-0.5}$. При этом граничные условия подбирались таким образом, чтобы невязка для начального приближения равнялась 10^{0}.

Вопросы, на которые требовалось получить ответ, ставились следующим образом — какой тип вычислителей и в каких случаях оказывается предпочтительнее в плане времени расчётов, а также какой из двух рассмотренных алгоритмов больше подходит под специфику задачи.

4.2. Центральный процессор и графический ускоритель

На графиках из разделов 4. 2 — 4.5 все результаты будут представлены в формате изменения производительности, в частности, будет показано, насколько повышается или понижается полезная скорость расчётов (отношение количества ячеек к времени расчётов) при варьировании одного из рассматриваемых параметров. При этом важно иметь общее представление относительно исходной производительности, поэтому в Табл. 2 дополнительно приведены времена, необходимые решателям для достижения заданной точности при расчётах на сцене без объектов и сетке 150x150x150.

Перед тем, как переходить к непосредственному рассмотрению результатов, стоит остановиться на таком аспекте, как их воспроизводимость. Обе рассматриваемые библиотеки не являются детерминированными, и от запуска к запуску наблюдаемые времена могут заметно отличаться. В случае NVIDIA AmgX эта особенность отражена в документации [15] и имеет вполне логичное объяснение — в целях оптимизации используется реализация алгоритмов с рандомизацией (авторы данной работы предполагают, что под этим подразумевается применение атомарных операций). Как следствие, в зависимости от запуска, для достижения сходимости требуется разное количество итераций, что и объясняет скачки производительности. Если подобное поведение нежелательно, то имеется возможность установить флаг *determinism_flag* — это приведёт к уменьшению производительности примерно на 10-20%, но обеспечит воспроизводимость результатов. В данной работе все запуски проводились именно с этим флагом.

Библиотека Hypre демонстрирует схожее поведение, однако найти для него официальное объяснение от разработчиков не удалось. В частности, процесс достижения высокой точности в ряде случаев оказывался более быстрым, чем поиск грубого решения. В Табл. 2 проявление этого утверждения можно проиллюстрировать на примере решателя BICGSTAB, взяв времена из для невязки $10^{-5.0}$ до $10^{-5.5}$. Задача поиска и устранения причин, из-за которых появляется подобная недетерминированность библиотеки Hypre, выходит за рамки данной работы, поэтому авторами была предпринята попытка только уменьшить её проявление, используя усреднение по пяти запускам.

Если говорить о самих результатах, то в первую очередь стоит отметить эффект от перехода к двойной точности. В используемой модели ускорителя количество блоков *float* и *double* соотносится как 3 к 1, однако полезная производительность снижается всего на 20-30%. Если узким местом алгоритма являются вычислительные операции, то падение должно было быть трёхкратным, если работа с памятью – двукратным. Так как исходный код библиотеки является закрытым, то можно лишь предположить, что столь небольшая разница обусловлена либо использованием смешанной точности, либо существенным вкладом в итоговое время работы вспомогательных, в том числе целочисленных, операций. Также важно подчеркнуть, что алгоритму AMG оказывается недостаточно одинарной точности для достижения небольшой невязки, что может быть проиллюстрировано резким ростом времени для строк $10^{-6.5}$ и $10^{-7.0}$ Табл. 2.

На Рис. 5 приведены аналогичные результаты, но в разрезе ускорения, достигаемого при

Табл. 2. Время расчётов для сцены без объектов на сетке 150x150x150, секунды

Невязка	Метод бисопряжённых градиентов (BICGSTAB)			Алгебраический многосеточный метод (AMG)		
	NVIDIA AmgX (один. точн.)	NVIDIA AmgX (двойн. точн.)	Hypre	NVIDIA AmgX (один. точн.)	NVIDIA AmgX (двойн. точн.)	Hypre
$10^{-1.0}$	0.1	0.15	1.63	0.15	0.2	4.24
$10^{-1.5}$	0.19	0.27	3.61	0.29	0.38	4.28
$10^{-2.0}$	0.39	0.54	7.13	0.43	0.56	8.44
$10^{-2.5}$	0.85	1.16	14.83	0.99	1.28	8.5
$10^{-3.0}$	1.63	2.2	30.94	1.96	2.52	8.51
$10^{-3.5}$	2.06	2.74	41.46	3.62	4.66	11.74
$10^{-4.0}$	2.95	3.94	44.61	5.55	7.15	12.8
$10^{-4.5}$	3.5	4.25	48.1	7.49	9.65	12.92
$10^{-5.0}$	4.22	5.22	48.25	9.56	12.31	16.99
$10^{-5.5}$	4.65	5.99	47.81	11.63	14.8	17.47
$10^{-6.0}$	5.31	6.72	51.42	13.85	17.48	21.24
$10^{-6.5}$	5.74	7.34	52.54	151.99	20.15	21.96
$10^{-7.0}$	6.75	8.3	52.55	290.13	22.64	25.77

переносе расчётов с центрального процессора на графический ускоритель. Наиболее заметные значения были зафиксированы при использовании метода BICGSTAB, в котором ускорение в процессе уточнения решения уменьшалось с 40 до 6 раз. В алгоритме AMG настолько заметного выигрыша не наблюдалось — хотя при большой невязке ускорение доходило до 30 раз, при поиске более точного решения оно достаточно быстро понижалось до величины менее 20%.

Можно сделать достаточно интересный вывод, что если рассматривать полученные результаты по отдельности, то с их помощью удастся обосновать два совершенно противоположных утверждения. Действительно, можно указать, что «в рассматриваемой задаче GPU на порядок обходит CPU» (что действительно верно для метода BICGSTAB) и «не смотря на многократное превосходство в теоретической производительности, полезная скорость расчётов на GPU соизмерима с CPU» (верно для AMG и небольших значений невязки). Таким образом, вопрос выбора предпочтительной комбинации вида «алгоритм x вычислитель» не имеет единственно верного ответа, и, как будет дополнительно показано в следующих разделах, должен рассматриваться в привязке к конкретной сцене и сетке.

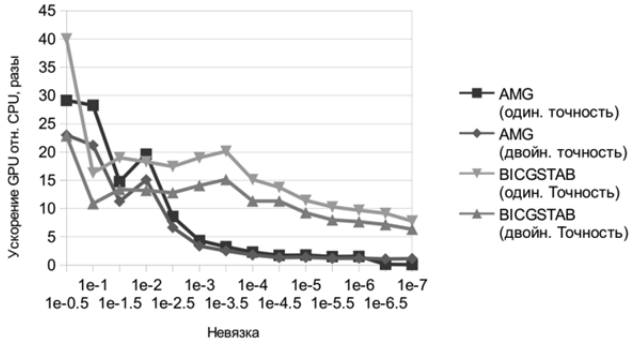

Рис. 5. Ускорение графического ускорителя относительно центрального процессора (больше — лучше)

4.3. Метод бисопряжённых градиентов и многосеточный метод

Для более объективного сравнения библиотек, отдельно стоит рассмотреть второй из поставленных ранее вопросов («какой из рассматриваемых алгоритмов оказывается предпочтительным?») с учётом типа моделируемых объектов. На Рис. 6 приведены результаты соответствующего сравнения, в рамках которого расчёты проводились на трёх типах сцен — без объектов, с одной большой сферой и набором из 125 сфер меньшего размера.

Как можно заметить, в библиотеке Hypre метод бисопряжённых градиентов вне зависимости от типа сцены обеспечивает заметный выигрыш (до 173%) только при нахождении грубого решения, в то время как при повышении точности лидировать начинает уже многосеточный метод (примерно на 70%). Однако стоит учесть, что для решаемой задачи в первую очередь важно достижение невязки в диапазоне от $10^{-0.5}$ до 10^{-2}, и, таким образом, в рамках данной библиотеки выбор алгоритма BICGSTAB оказывается более обоснованным. Данный вывод с некоторой натяжкой распространяется и на библиотеку NVIDIA AmgX. В ней наблюдается схожее поведение, когда сначала большую производительность демонстрирует метод BICGSTAB, но потом он может уступить лидерство алгоритму AMG. При этом появляется зависимость от типа сцены — к примеру, если в области нет объектов, то метод бисопряжённых градиентов оказывается абсолютным лидером.

Таким образом, можно резюмировать, что, несмотря на явную зависимость от величины невязки, при моделировании дыма как на графических ускорителях, так и на центральных процессорах предпочтение всё-таки стоит отдавать алгоритму бисопряжённых градиентов. Хотя многосеточный метод может демонстрировать лучшие результаты при достижении высокой точности, найти реальный сценарий, когда эта особенность будет востребована, авторы не смогли. В разделе 4 было показано, что для воспроизведения этого случая требуется сильно измельчать сетку при фиксированном геометрическом размере области, что имеет мало смысла из-за ограничений, накладываемых форматом кадра. Действительно, можно определить момент, когда любая ячейка отобразится не более чем в один пиксель кадра. При достижении этого порога дальнейшее измельчение сетки не приведёт к существенным видимым отличиям, так как появившиеся в результате использования более высокой точности эффекты в любом случае будут утеряны из-за усреднения при рендеринге.

4.4. Разные типы сцен

Следующее сравнение также заключалось в оценке производительности при варьировании типа сцены, но проводилось в несколько ином разрезе — учитывалось изменение полезной скорости расчётов, а не соотношение времени работы двух алгоритмов. Стоит отметить, что результаты данного теста достаточно сложно предугадать заранее. С одной стороны, появление выколотых подобластей приводит к тому, что размерность СЛАУ понижается и, как следствие, время одной итерации также становится меньшим. С другой, наложение дополнительных граничных условий ухудшает скорость сходимости, и итоговое количество итераций увеличивается.

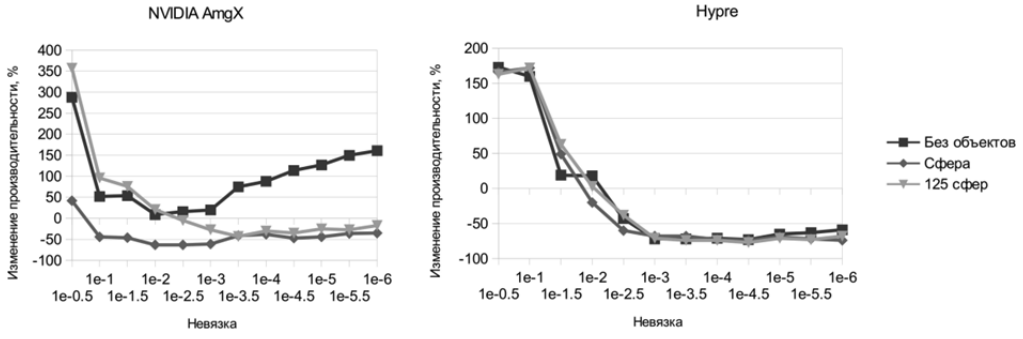

Рис. 6. Изменение производительности при переходе от алгоритма AMG к BICGSTAB (больше - лучше)

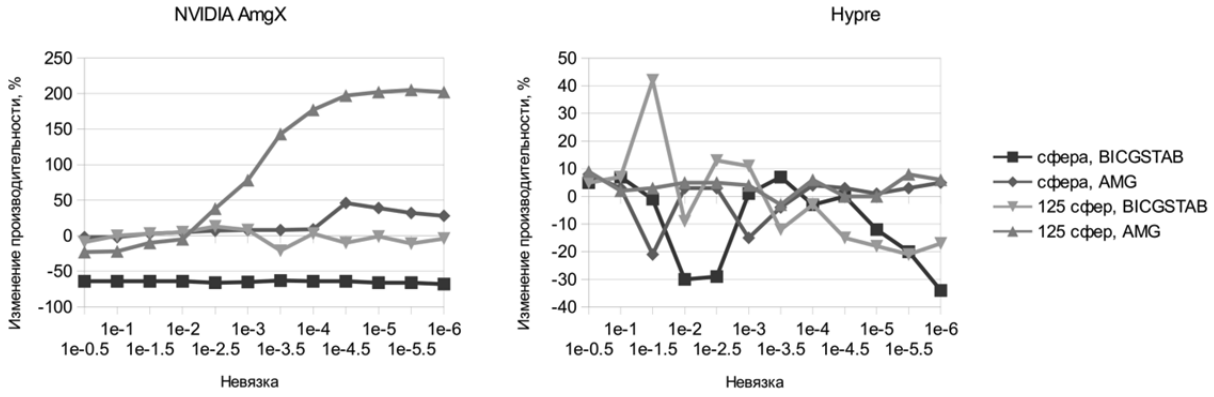

Рис. 7. Влияние типов сцен на производительность (больше — лучше)

На Рис. 7 показано, как изменялась производительность при переходе к областям с объектами. В случае библиотеки NVIDIA AmgX наблюдаемый эффект в первую очередь определялся типом алгоритма. На сложных сценах многосеточный метод работал заметно быстрее при достижении небольшой невязки, в то время как при поиске грубого решения наблюдалось снижение производительности. В противоположность ему, при добавлении в область объектов метод бисопряжённых градиентов стабильно замедлялся на величину, не зависящую от целевой невязки.

Если рассматривать результаты для библиотеки Hypre, также представленные на Рис. 7, то какие-либо закономерности обнаружить не удаётся. Логично было бы сделать вывод, что наблюдаемые скачки являются результатом накладных расходов или ошибки, допущенной авторами при проведении тестирования. Однако стоит учесть, что на базе этих же данных был построен Рис. 6, в котором ускорение метода AMG относительно BICGSTAB практически не зависит от типа сцены. Таким образом, можно заключить, что природа данного явления обусловлена спецификой реализации алгоритмов в Hypre и в первую очередь зависит от структуры матрицы.

4.5. Сетки разного размера

В последнем тесте оценивалось, как изменится скорость расчётов при переходе на более крупные сетки в случае сцены без объектов. Как отмечалось ранее, под понятием производительности понимается отношение количества ячеек к времени расчётов, требуемых для достижения соответствующей невязки. Данная величина идейно близка к метрике «байты в секунду», которая используется для задания пропускной способности и применяется при оценке программ, узким местом в которых является работа с памятью. Основное отличие, которое проявляется в рамках данного раздела, заключается в том, что рассматриваемые библиотеки, в зависимости от размера матриц, могут использовать разные внутренние форматы для их хранения, поэтому оценить реальную пропускную способность не представляется возможным. Также, в силу увеличения размерности СЛАУ, в приведённые ниже результаты был внесён некоторый вклад из-за соответствующего изменения скорости сходимости.

Так как для рассматриваемой задачи предпочтительно проведение моделирования на больших сценах, то оценивался только метод BICGSTAB в связи с его существенно меньшими требованиями к памяти. Полученные результаты приведены на Рис. 8.

Как можно заметить, в случае библиотеки NVIDIA AmgX, выполняемой на графическом ускорителе, переход на сетки большего размера приводит к уменьшению скорости расчётов, причём на относительно большую величину — до 8.3 раз при увеличении количества ячеек в 216 раз. На первый взгляд, подобное поведение аномально для графического ускорителя, так как с ростом размера матрицы повышается доступный параллелизм, что, в свою очередь, должно привести к повышению скорости расчётов. Действительно, подобная особенность является прямым следствием специфики архитектуры CUDA, позволяющей переключаться

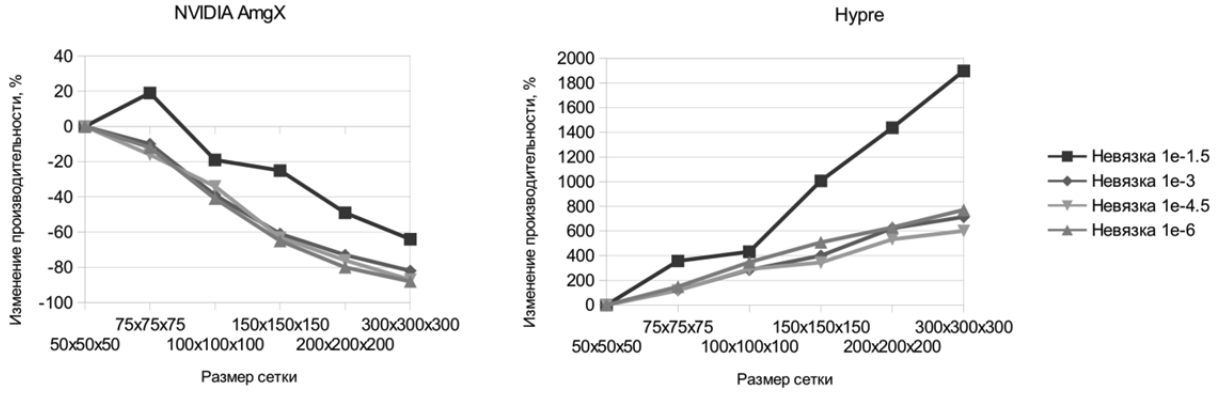

Рис. 8. Влияние размера сеток на производительность, метод BICGSTAB (больше — лучше)

между нитями с нулевой задержкой, и может наблюдаться в задачах совершенно разного типа. К сожалению, на случай решения СЛАУ с разреженными матрицами эта закономерность не распространяется. Как было показано в другой работе авторов, посвящённой разработке собственных реализаций методов BICGSTAB и D-ILU для GPU [16], весьма схожее поведение в том случае объясня лось ростом атомарных операций над глобальной памятью, которые с некоторого момента становились основным узким местом.

Библиотека Hypre, наоборот, обеспечивала большую скорость расчётов при увеличении размера сеток, причём в лучшем случае рост производительности доходил до 20 раз. Таким образом, если ускорение NVIDIA AmgX относительно Hypre на сетке 50x50x50 составляло сотни раз, то при переходе к области из 300x300x300 ячеек выигрыш находился в диапазоне от 4.9 до 12 раз. Логично предположить, что при дальнейшем росте размера сетки и достаточности памяти подобное поведение будет сохраняться, и в некоторый момент производительность CPU и GPU реализаций алгоритма окажется идентичной, после чего начнёт лидировать уже центральный процессор.

Стоит отметить, что в большинстве случаев аниматор подбирает размер сцены таким образом, чтобы максимально задействовать доступную память, поэтому можно сделать вывод, что архитектура центрального процессора на практике оказывается более предпочтительной — она не только позволяет оперировать намного большим объёмом памяти в SMP-режиме (в нашем случае, это 64 против 12 гигабайт), но и на целевых сценариях обеспечивает соизмеримую производительность. При этом перенос вычислений на графический ускоритель оказывается действительно оправданным при использовании небольших сеток, а также в случаях, когда основным приоритетом является не столько повышение скорости моделирования, сколько уменьшение стоимости аппаратной платформы.

Заключение

Проведённое в рамках работы сравнение производительности библиотеки NVIDIA AmgX (GPU) и Hypre (CPU) показало, что даже при решении столь стандартизированной и изученной задачи, как моделирование потока дыма с помощью полулагранжевого подхода [6], остаётся множество особенностей, делающих вопросы выбора конкретной аппаратной платформы и алгоритма достаточно неоднозначными и зависящими от множества факторов. По результатам проведённого авторами тестирования могут быть сформулированы следующие рекомендации:

- Если сравнивать многосеточный метод и стабилизированный метод бисопряжённых градиентов, то предпочтение стоит отдать второму, так как он показывает лучшие результаты при поиске грубого решения и использует меньше памяти.
- В общем случае предпочтительно использование одинарной точности, которой оказывается достаточно для достижения целевого порога ошибки. При переходе на двойную точность наблюдается снижение производительности в среднем на 20-30%.

- Добавление в сцену объектов может привести как к повышению производительности, так и её снижению — в рассмотренных случаях изменения доходили до нескольких раз.
- В случае выбора между CPU и GPU вопрос сводится к выделению основного приоритета. Ускоритель позволяет достичь на один или два порядка большей скорости расчётов на сетках небольшого размера, в то время как с их ростом полезная производительность уменьшается почти линейно. Центральный процессор демонстрирует лучшие результаты при работе с большими сценами и позволяет задействовать больший объём памяти.

Литература

1. G.Y. Gardner. Visual Simulation of Clouds // Computer Graphics (SIGGRAPH 85 Conference Proceedings). 1985. pp. 297–384.
2. W. Reeves. Particle Systems - A Technique for Modeling a Class of Fuzzy Objects // ACM Transactions on Graphics 2(2). April 1983. pp. 91-108.
3. W. Dong, X. Zhang, C. Zhang. Smoke Simulation Based on Particle System in Virtual Environments // 2010 International Conference on Multimedia Communications. 2010.
4. N. Foster, D. Metaxas. Realistic Animation of Liquids // Graphical Models and Image Processing 58(5). 1996. pp. 471–483.
5. J. Stam. Stable Fluids // SIGGRAPH 99 Conference Proceedings, Annual Conference Series. 1999. pp. 121–128.
6. R. Fedkiw, J. Stam, H. Jensen. Visual Simulation of Smoke // Proceedings of SIGGRAPH 2001. 2001. pp. 15–22.
7. M. Harris. Fast Fluid Dynamics Simulation on the GPU // GPU Gems: Programming Techniques, Tips and Tricks for Real-Time Graphics. Addison-Wesley. 2004. pp. 637–665.
8. K. Crane, I. Llamas, S. Tariq. Real-Time Simulation and Rendering of 3D Fluids // GPU Gems 3. Addison-Wesley. 2007. pp. 633–675.
9. S. He, H. Wong, W. Pang, U. Wong. Real-time smoke simulation with improved turbulence by spatial adaptive vorticity confinement // Computer Animation & Virtual Worlds 22(2-3). 2011. pp. 107–114.
10. T. Pfaff, N. Thuerey, J. Cohen, S. Tariq, M. Gross. Scalable Fluid Simulation using Anisotropic Turbulence Particles // Proceedings of ACM SIGGRAPH Asia (Seoul, Korea, December 15-18, 2010), ACM Transactions on Graphics 29(5). pp. 174:1-174:8.
11. R.D. Falgout, J.E. Jones, U.M. Yang. The Design and Implementation of hypre, a Library of Parallel High Performance Preconditioners // Numerical Solution of Partial Differential Equations on Parallel Computers 51(2006). pp. 267-294.
12. S. Posey, S. See, M. Wang. GPU Progress and Directions in Applied CFD // Proceedings of Eleventh International Conference on CFD in the Minerals and Process Industries. 2015.
13. M. Naumov et al.. AMGX: a Library for GPU Accelerated Algebraic Multigrid and Preconditioned Iterative Methods // SIAM Journal on Scientific Computing 37(5). October 2015. pp. 602-626.
14. Q. Dai, X. Yang. Interactive smoke simulation and rendering on the GPU // Proceedings of the 12th ACM SIGGRAPH International Conference on Virtual-Reality Continuum and Its Applications in Industry. Hong Kong. November 2013. pp. 177-182.
15. Accelerating ANSYS Fluent 15.0 Using NVIDIA GPUs (NVIDIA, 2014). Available at: https://www.nvidia.com/content/tesla/pdf/ansys-fluent-nvidiagpu-userguide.pdf (accessed May 4, 2017).
16. Кривов М.А., Гризан С.А. Опыт разработки гибридных версий решателей разреженных СЛАУ // Параллельные вычислительные технологии (ПаВТ'2012): труды международной научной конференции. Челябинск. 2012. С. 553–558.

Кривов Максим Андреевич. МГУ им. М.В. Ломоносова, г. Москва, Россия. Математик. Количество печатных работ: 28. Область научных интересов: графические ускорители, математическая физика, высокопроизводительные вычисления. E-mail: m_krivov@cs.msu.su

Новиков Александр Игоревич. МГУ им. М.В. Ломоносова, г. Москва, Россия. Магистрант. Область научных интересов: суперкомпьютеры, параллельное программирование, численные методы. E-mail: alexn@itis.cs.msu.ru

Юданов Анатолий Александрович. Puppetworks Animation Studio, г. Будапешт, Венгрия. Lead FX Artist. Количество печатных работ: 3. Область научных интересов: компьютерная графика, математическое моделирование, визуализация. E-mail: ХАРКОННЕН@gmail.com.

Performance Comparison of Hypre and NVIDIA AmgX Libraries on Fume Modeling Problem

M.A. Krivov[1], A.I. Novikov[1], A.A. Yudanov[2]

[1] Lomonosov Moscow State University, Moscow, Russia
[2] Puppetworks Animation Studio, Budapest, Hungary

This paper considers the task of interaction of incompressible fluid with multiple objects that was introduced in computer graphics and can be reduced to three dimensional Euler equations. We proposed an approach to estimating an acceptable error under projection stage that is equal to solving of Poisson equation with mixed type

boundaries on structured grid. For this stage, we used third party libraries Hypre (CPU) and NVIDIA AmgX (GPU) as an implementation of methods for SLAE solving with sparse matrices. The testing of the proposed solver was performed on system with CPU Xeon E5-2697v3 (14 cores) and GPU Tesla K40 with different numerical methods, scene types and grid sizes. It was demonstrated that the effective performance highly depends on scene specifics, therefore the questions of library, hardware and algorithm choosing are quite ambiguous. In particular, the speed-up of GPU library over its CPU-based analogue was ranged from 20% to 273 folds.
Keywords: mathematical modeling, visualization, fume, multigrid method, biconjugate gradient stabilized method, CPU, GPU, Hypre, NVIDIA AmgX

DOI 10.14357/20718632180202

References

1. G.Y. Gardner. Visual Simulation of Clouds // Computer Graphics (SIGGRAPH 85 Conference Proceedings). 1985. pp. 297–384.
2. W. Reeves. Particle Systems - A Technique for Modeling a Class of Fuzzy Objects // ACM Transactions on Graphics 2(2). April 1983. pp. 91-108.
3. W. Dong, X. Zhang, C. Zhang. Smoke Simulation Based on Particle System in Virtual Environments // 2010 International Conference on Multimedia Communications. 2010.
4. N. Foster, D. Metaxas. Realistic Animation of Liquids // Graphical Models and Image Processing 58(5). 1996. pp. 471–483.
5. J. Stam. Stable Fluids // SIGGRAPH 99 Conference Proceedings, Annual Conference Series. 1999. pp. 121–128.
6. R. Fedkiw, J. Stam, H. Jensen. Visual Simulation of Smoke // Proceedings of SIGGRAPH 2001. 2001. pp. 15–22.
7. M. Harris. Fast Fluid Dynamics Simulation on the GPU // GPU Gems: Programming Techniques, Tips and Tricks for Real-Time Graphics. Addison-Wesley. 2004. pp. 637–665.
8. K. Crane, I. Llamas, S. Tariq. Real-Time Simulation and Rendering of 3D Fluids // GPU Gems 3. Addison-Wesley. 2007. pp. 633–675.
9. S. He, H. Wong, W. Pang, U. Wong. Real-time smoke simulation with improved turbulence by spatial adaptive vorticity confinement // Computer Animation & Virtual Worlds 22(2-3). 2011. pp. 107–114.
10. T. Pfaff, N. Thuerey, J. Cohen, S. Tariq, M. Gross. Scalable Fluid Simulation using Anisotropic Turbulence Particles // Proceedings of ACM SIGGRAPH Asia (Seoul, Korea, December 15-18, 2010), ACM Transactions on Graphics 29(5). pp. 174:1-174:8.
11. R.D. Falgout, J.E. Jones, U.M. Yang. The Design and Implementation of hypre, a Library of Parallel High Performance Preconditioners // Numerical Solution of Partial Differential Equations on Parallel Computers 51(2006). pp. 267-294.
12. S. Posey, S. See, M. Wang. GPU Progress and Directions in Applied CFD // Proceedings of Eleventh International Conference on CFD in the Minerals and Process Industries. 2015
13. M. Naumov et al.. AMGX: a Library for GPU Accelerated Algebraic Multigrid and Preconditioned Iterative Methods // SIAM Journal on Scientific Computing 37(5). October 2015. pp. 602-626.
14. Q. Dai, X. Yang. Interactive smoke simulation and rendering on the GPU // Proceedings of the 12th ACM SIGGRAPH International Conference on Virtual-Reality Continuum and Its Applications in Industry. Hong Kong. November 2013. pp. 177-182.
15. Accelerating ANSYS Fluent 15.0 Using NVIDIA GPUs (NVIDIA, 2014). Available at: https://www.nvidia.com/content/tesla/pdf/ansys-fluent-nvidiagpu-userguide.pdf (accessed May 4, 2017).
16. Krivov, M.A, Grizan, S.A. Opyt razrabotki gibridnyh versij reshatelej razrezhennyh SLAU [On Development of Hybrid Versions of Sparse SLAE Solvers]. Parallel'nye vychislitel'nye tehnologii (PaVT'2012): trudy mezhdunarodnoj nauchnoj konferencii [In Proceedings of Parallel Computing Technologies (PaCT'2012)]. Chelyabinsk. 2012. pp. 553–558.

M.A. Krivov. Lomonosov Moscow State University, Moscow, Russia. Mathematician. Author of 28 papers. Research interests: GPU, mathematical physics, high performance computing. E-mail: m_krivov@cs.msu.su

A.I. Novikov. Lomonosov Moscow State University, Moscow, Russia. Student. Research interests: supercomputers, parallel programming, numerical methods. E-mail: alexn@itis.cs.msu.ru

A.A. Yudanov. Puppetworks Animation Studio, Budapest, Hungary. Lead FX Artist. Author of 3 papers. Research interests: computer graphics, mathematical modeling, visualization. E-mail: XAPKOHHEH@gmail.com