УДК 551.21.3;535.341;519.85

МОДЕЛИРОВАНИЕ ПЕРЕНОСА ИЗЛУЧЕНИЯ МЕТОДОМ МОНТЕ-КАРЛО С ИСПОЛЬЗОВАНИЕМ CUDA

Д.П. Глазунов, И.Ю. Гендрина (Томск)

Введение

Исследование процесса переноса излучения в рассеивающих средах является одной из ключевых задач в области вычислительной физики и оптики. Одним из наиболее универсальных и широко применяемых подходов к её решению является метод Монте-Карло, позволяющий проводить численную имитацию сложных стохастических процессов, включая взаимодействие излучения с веществом. Настоящая работа посвящена изучению и реализации алгоритмов моделирования переноса частиц с использованием метода Монте-Карло, с акцентом на вопросы параллельных вычислений и их эффективности.

Целью работы является реализация алгоритма прямого моделирования переноса излучения в рассеивающих средах, а также оценка эффективности параллельных вычислений и визуализации результатов.

1. Материалы и методы

1.1. Метод Монте-Карло

Метод Монте-Карло – это обобщённое название численных методов, использующих случайные величины и вероятностное моделирование для приближённого решения математических задач. Впервые эти подходы начали применяться в середине XX века при расчётах ядерных процессов, а затем получили широкое распространение в таких областях, как физика, биология, экономика, а также компьютерное моделирование сложных систем.

Основная идея метода состоит в генерации большого числа реализаций случайных событий, описывающих поведение моделируемой системы. Полученные данные затем обрабатываются статистически, что позволяет приблизиться к искомому результату с заданной точностью. Метод особенно эффективен для высокоразмерных задач и в тех случаях, когда аналитические методы неприменимы [1].

В данной работе метод Монте-Карло используется для моделирования переноса частиц в ограниченной среде, обладающей рассеивающими и поглощающими свойствами.

1.2. Перенос частиц

Под переносом частиц понимается совокупность физических процессов, происходящих при взаимодействии частиц с веществом [2]. К таким процессам, как правило, относят:

1. **поглощение** – полное исчезновение частицы в результате взаимодействия с веществом, при котором её энергия передаётся среде;
2. **рассеяние** – изменение направления движения частицы при столкновении, часто сопровождаемое частичной потерей энергии;
3. **вылет за границу** – ситуация, при которой частица покидает определённую область и не учитывается в дальнейшем моделировании.

Важной характеристикой является свободный пробег – расстояние, которое частица проходит между двумя последовательными взаимодействиями. В большинстве случаев это случайная величина, подчиняющаяся экспоненциальному закону распределения.

Для корректного моделирования переноса частиц методом Монте-Карло необходимо тщательно реализовать выбор расстояния до следующего столкновения, выбор типа взаимодействия, и расчёт направления движения после рассеяния.

При реализации параллельного моделирования важно учитывать необходимость обмена частицами между потоками или процессами, особенно когда частицы выходят за пределы участка среды, закреплённого за конкретным вычислительным процессом.

1.3. Индикатриса рассеяния

Индикатриса рассеяния – это функция, описывающая угловое распределение рассеянного излучения или потока частиц. В контексте физики газов, аэрозольных и многокомпонентных сред, а также в задачах радиационного переноса, индикатриса рассеяния (или просто индикатриса) характеризует вероятность рассеяния частицы (или фотона) на заданный угол относительно направления её первоначального движения.

Формально, индикатриса определяется как нормированная функция $\Phi(\cos(\theta), \varphi)$ (плотность распределения), где θ – угол между направлением падающего и рассеянного излучения, φ – азимутальный угол, определяющий направление рассеянного излучения в плоскости, перпендикулярной первоначальному направлению. Данная функция удовлетворяет условию нормировки:

$$\int_0^{2\pi} \int_0^{\pi} \Phi(\cos(\theta), \varphi) \sin(\theta)\, d\theta d\varphi = 1,$$

или, при использовании симметрии относительно азимута, в виде:

$$2\pi \int_0^{\pi} \Phi(\cos(\theta)) \sin(\theta) d\theta = 1.$$

В задачах переноса излучения индикатриса рассеяния часто представляется в виде разложения по полиномам Лежандра, поскольку эти полиномы образуют ортогональную систему на отрезке $[-1,1]$, где аргументом является $\mu = \cos(\theta)$ – косинус угла между направлениями движения до и после акта рассеяния.

Для моделирования в соответствии с индикатрисой рассеяния $\mu = \cos(\theta)$, необходимо использовать теорему об обратной функции. Пусть $\Phi(\mu)$ – нормированная индикатриса на отрезке $\mu \in [-1,1]$, тогда функция распределения:

$$F(\mu) = \int_{-1}^{\mu} \Phi(\mu') d\mu'.$$

Физически индикатриса рассеяния $\Phi(\mu)$ является неотрицательной функцией, определённой на интервале $[-1,1]$, обычно предполагается, что она либо непрерывна, либо задана таблично с достаточно частыми узлами, позволяющими её адекватно аппроксимировать. При этом соответствующая функция распределения $F(\mu)$, определяемая как интеграл от $\Phi(\mu)$, будет непрерывной и строго монотонно возрастающей. Согласно теореме об обратной функции, существует обратная функция F^{-1}, отображающая отрезок $[0,1]$ в $[-1,1]$ и удовлетворяющая равенству:

$$F\left(F^{-1}(\alpha)\right) = \alpha \quad \forall \alpha \in [0,1].$$

Это позволяет находить направление рассеяния, используя равномерную случайную величину $\alpha \in [0,1]$.

$$F(\mu) = \alpha \quad \Rightarrow \quad \mu = F^{-1}(\alpha)$$

В данной работе использованы следующие индикатрисы рассеяния, для каждой из которых приведены функция $\Phi(\mu)$ и выражение для моделирования косинуса угла рассеяния μ:

1. Изотропная индикатриса [2]:

$$\Phi(\mu) = \frac{1}{2}, \quad \mu = 2\alpha - 1.$$

2. Рэлея [3]:

$$\Phi(\mu) = \frac{3}{8}(1+\mu^2), \quad \mu = \begin{cases} \frac{8\alpha}{3} - 1, & \alpha \le \frac{3}{4}, \\ \sqrt[3]{8\alpha - 7}, & \alpha > \frac{3}{4}. \end{cases}$$

3. Хеньи–Гринстейна [4]:

$$\Phi(\mu) = \frac{1-g^2}{2(1+g^2-2g\mu)^{\frac{3}{2}}}, \quad \mu = \frac{1}{2g}\left(1+g^2-\left(\frac{1-g^2}{1-g+2g\alpha}\right)^2\right).$$

Здесь $g \in [-1,1]$ – показатель асимметрии.

4. Ламберта [5]:

$$\Phi(\mu) = 2\mu, \quad \mu = \sqrt{\alpha}.$$

5. Ми [6]:

$$\Phi(\mu) = \frac{me^{m(1-\mu)}}{e^{2m}-1}, \quad \mu = 1 - \frac{1}{m}\ln\left(e^{2m} - \alpha(e^{2m}-1)\right).$$

Здесь $m \in [0, \infty)$ — «разброс» распределения.

1.4. Ускорение параллельной программы

Ускорение (speedup) — это метрика, показывающая, во сколько раз параллельное выполнение программы быстрее последовательного [7]. Оно определяется как:

$$S = \frac{T_1}{T_k}. \tag{1}$$

где T_1 – время выполнения задачи на одном процессе;

T_k – время выполнения задачи на k процессах.

В идеальном случае ускорение пропорционально количеству процессов:

$$S \sim k. \tag{2}$$

Однако в реальных условиях ускорение обычно меньше k из-за накладных расходов на синхронизацию, передачи данных и неравномерного распределения нагрузки.

1.5. Эффективность параллельного алгоритма

Эффективность (efficiency) показывает, насколько хорошо используются вычислительные ресурсы, и определяется как [7]:

$$E = \frac{S}{k}. \tag{3}$$

При идеальной работе всех процессов:

$$E \sim 1.$$

1.6. Интерпретация ускорения и эффективности в архитектуре CUDA

В классической модели параллельных вычислений ускорение определяется по формуле (1), а эффективность - по формуле (3).

В CUDA под k можно понимать число **нитей (threads)** или/и **блоков (blocks)** [8], в зависимости от уровня анализа.

- **блоки (blocks)** можно рассматривать как аналоги процессов в классической модели, так как они распределяются между мультипроцессорами (SM) и выполняются независимо. Поэтому при увеличении числа блоков ускорение чаще всего масштабируется ближе к линейному закону (2);
- **нити (threads) внутри блока** делят между собой ресурсы одного мультипроцессора (регистр, shared memory, ALU), поэтому рост ускорения с

увеличением числа нитей ограничен. Здесь наблюдается эффект насыщения: после определённого количества нитей время выполнения почти не уменьшается.

Таким образом, в CUDA интерпретация ускорения и эффективности принимает вид:

$$S_{CUDA} = \frac{T_{1,1}}{T_{b,t}},$$

где b — число блоков, t — число нитей в блоке.

Тогда эффективность:

$$E_{CUDA} = \frac{S_{CUDA}}{b \cdot t}.$$

На практике значение E_{CUDA} обычно значительно меньше 1, особенно при большом числе нитей, что связано с накладными расходами на доступ к памяти и ограничениями пропускной способности GPU.

1.7. Алгоритм моделирования переноса излучения

Моделирование переноса частиц (или фотонов) в рассеивающей и/или поглощающей среде осуществляется поэтапно на основе стохастических процедур, реализующих вероятностную природу взаимодействия излучения с веществом [3].

В данной работе рассматривается стохастическая модель переноса излучения, основанная на использовании метода Монте-Карло. Модель предполагает, что частицы (например, фотоны) распространяются в среде, которая может быть либо однородной, либо состоять из конечного числа горизонтально однородных слоев. В каждом из таких слоёв характеристики считаются постоянными, однако они могут отличаться от слоя к слою, формируя неоднородную структуру в целом.

Движение частицы в среде описывается как чередование свободного пробега и актов взаимодействия со средой. Длина пробега до следующего взаимодействия подчиняется экспоненциальному закону, что отражает статистический характер процессов рассеяния и поглощения. После прохождения случайного расстояния частица может либо поглотиться, либо рассеяться, в зависимости от локальных свойств среды. Вероятность поглощения определяется соотношением между соответствующими коэффициентами поглощения и полного ослабления, тогда как вероятность рассеяния – дополнением до единицы.

Если происходит рассеяние, направление дальнейшего движения частицы изменяется согласно выбранной индикатрисе. В модели предусмотрено несколько различных индикатрис, каждая из которых применяется в зависимости от физических предпосылок. Угол между исходным и новым направлением движения моделируется через косинус угла рассеяния, а случайность плоскости отклонения обеспечивается выбором азимутального угла как равномерно распределенного в интервале $[0,2\pi]$.

Виртуальные границы слоёв играют важную роль в определении текущего состояния частицы: по мере продвижения вдоль направления распространения отслеживается, в каком именно слое она находится, и, следовательно, какие характеристики взаимодействия применимы на текущем этапе. Если частица выходит за пределы среды, процесс завершён, и такая траектория считается покинувшей систему. Аналогично, в случае поглощения частицы её эволюция прекращается.

Таким образом, построенная модель позволяет воспроизводить статистику взаимодействий, распределение рассеянных направлений и длину траекторий частиц в различных конфигурациях среды. Благодаря модульному описанию, она применима как для анализа однородных систем с фиксированными свойствами, так и для исследований сложных слоистых структур с резкими изменениями параметров на границах, при этом в статистику дополнительно сохраняются такие параметры, как количество рассеяний, число поглощений и число вылетов.

2. Результаты

В общей сложности было реализовано две версии симуляционной программы: первая – без записи, а вторая – с записью в файл, оптимизированной с учётом параллельных вычислений.

2.1. Параллельная версия без записи

Для оценки эффективности параллельного алгоритма была разработана первая версия программы, в которой результаты моделирования не записывались в файл. Такой подход позволил минимизировать задержки, вызванные операциями ввода-вывода и сосредоточиться на чистом анализе производительности CUDA-ядра. При этом в выводимой статистике учитывались ключевые параметры, такие как число рассеяний, количество поглощений и число вылетов частиц.

2.1.1. Результаты программы

Перед запуском моделирования формируется таблица параметров слоёв. В табл. 1 представлена конфигурация, использованная для серии тестов.

Таблица 1. Параметры слоёв неоднородной среды, используемые при моделировании

Граница(Z)	Вероятность поглощения	Коэффициент ослабления	Вид индикатрисы	Параметр индикатрисы
1,0	0,5	1	Изотропная	–
2,0	0,25	0,5	Рэлеевская	–
3,0	0,75	0,75	Хеньи-Гринстейна	1
4,0	0,1	0,25	Ламбертовскае	–
5,0	0,3	0,6	Ми-рассеяние	1

После настройки физической модели была проведена серия численных экспериментов ($n = 10^8$), направленных на оценку производительности параллельной реализации. В табл. 2 представлены результаты для случая, когда увеличивается только количество блоков.

Таблица 2. Результаты моделирования при увеличении числа блоков

$\boldsymbol{b; t}$	1; 1	2; 1	4; 1	8; 1	16; 1	32; 1
$\boldsymbol{T}$(мс)	141601,344	70855,578	35150,875	17509,902	8768,73	4414,209
$\boldsymbol{S_{CUDA}}$	1	1,9984502	4,028387458	8,086929556	16,14844385	32,07853185
$\boldsymbol{E_{CUDA}}$	1	0,9992251	1,007096865	1,010866194	1,00927774	1,00245412

В табл. 3 представлены результаты для случая, когда увеличивается только количество нитей.

Таблица 3. Результаты моделирования при увеличении числа нитей

$\boldsymbol{b; t}$	1; 1	1; 2	1; 4	1; 8	1; 16
$\boldsymbol{T}$(мс)	141601,344	110064,977	84014,203	60231,793	41431,469
$\boldsymbol{S_{CUDA}}$	1	1,286524995	1,685445305	2,350940209	3,417724436
$\boldsymbol{E_{CUDA}}$	1	0,643262498	0,421361326	0,293867526	0,213607777

Таблица 3. Результаты моделирования при увеличении числа нитей (продолжение)

$\boldsymbol{b; t}$	1; 32	1; 64	1; 128	1; 256
$\boldsymbol{T}$(мс)	27332,418	13647,12	6838,303	3741,826
$\boldsymbol{S_{CUDA}}$	5,180710466	10,37591404	20,70708829	37,84284571
$\boldsymbol{E_{CUDA}}$	0,161897202	0,162123657	0,161774127	0,147823616

В табл. 4 представлены результаты для случая, когда блоки и нити увеличиваются пропорционально.

Таблица 4. Результаты моделирования при увеличении числа блоков и нитей пропорционально.

$\boldsymbol{b; t}$	1; 1	2; 2	4; 4	8; 8	16; 16	32; 32
$\boldsymbol{T}$(мс)	141601,344	55301,902	21033,213	7528,599	2589,577	866,461
$\boldsymbol{S_{CUDA}}$	1	2,560514899	6,732273571	18,80845878	54,68126416	163,4249482
$\boldsymbol{E_{CUDA}}$	1	0,640128725	0,420767098	0,293882169	0,213598688	0,159594676

2.2. Параллельная версия с записью

Перед запуском основного моделирования и его записи выполняется тестовый прогон, охватывающий 10% от исходного числа экспериментов. Один эксперимент соответствует построению одной траектории. Траектория в данном контексте – это полный путь частицы от её рождения до завершения движения (поглощения или выхода за пределы области). Она представляет собой последовательность шагов, для каждого из которых фиксируются координаты и значение μ. Таким образом, каждая траектория хранит историю движения частицы и может быть полностью восстановлена при анализе. На основании результата тестового прогона определяется эксперимент с наибольшим количеством шагов. Полученная информация используется для выделения памяти, необходимой для хранения полной записи. Пользователь может задать число экспериментов (n_w), которые требуется сохранить, либо выбрать сохранение всех экспериментов. Запись производится в формате *.bin*.

Во время моделирования каждый эксперимент порождает одну траекторию. Графический процессор (GPU) рассчитывает движение частицы шаг за шагом, формируя её траекторию. При необходимости эти данные временно сохраняются в

общую память, доступную как GPU, так и центральному процессору (CPU). Когда траектория завершается (частица поглощается или выходит за пределы области), GPU отмечает готовность данных, после чего CPU немедленно переносит их в файл. Благодаря такому разделению задач процесс построения новых траекторий на GPU и запись завершённых на CPU выполняются параллельно, что ускоряет моделирование и снижает задержки.

2.2.1. Результаты программы

Перед запуском моделирования формируется таблица параметров слоёв. В таблице 1 представлена конфигурация, использованная для серии тестов. После настройки физической модели была проведена серия экспериментов, в ходе которых смоделировано $n = 10^8$ траекторий, из которых для записи было выбрано $n_w = 10^5$. Эти данные использовались для оценки производительности параллельной реализации. В табл. 5 представлены результаты экспериментов при увеличении только количества блоков.

Таблица 5. Результаты моделирования при увеличении числа блоков (с записью)

$b; t$	1; 1	2; 1	4; 1	8; 1	16; 1	32; 1
T(мс)	141621,641	70876,75	35170,23	17529,1	8789,374	4434,394
S_{CUDA}	1	1,998139601	4,02674765	8,079230594	16,11282453	31,93709016
E_{CUDA}	1	0,999069801	1,006686912	1,009903824	1,007051533	0,998034068

В табл. 6 представлены результаты экспериментов при увеличении только количества нитей.

Таблица 6. Результаты моделирования при увеличении числа нитей (с записью)

$b; t$	1; 1	1; 2	1; 4	1; 8	1; 16
T(мс)	141621,641	110085,578	84036,445	60250,508	41453,621
S_{CUDA}	1	1,286468614	1,685240743	2,350546837	3,416387702
E_{CUDA}	1	0,643234307	0,421310186	0,293818355	0,213524231

Таблица 6. Результаты моделирования при увеличении числа нитей (продолжение)

$b; t$	1; 32	1; 64	1; 128	1; 256
T(мс)	27355,682	13668,245	6864,459	3761,885
S_{CUDA}	5,177046619	10,36136249	20,63114384	37,64645676
E_{CUDA}	0,161782707	0,161896289	0,161180811	0,147056472

В табл. 7 представлены результаты для случая, когда блоки и нити увеличиваются пропорционально.

Таблица 7. Результаты моделирования при увеличении числа блоков и нитей пропорционально (с записью).

$b; t$	1; 1	2; 2	4; 4	8; 8	16; 16	32; 32
T(мс)	141621,641	55323,145	21056,232	7548,394	2608,889	886,318
S_{CUDA}	1	2,559898592	6,725877688	18,76182417	54,28427235	159,7864886
E_{CUDA}	1	0,639974648	0,420367355	0,293153503	0,212047939	0,156041493

3. Обсуждение

3.1. Параллельная версия без записи

Как видно (табл. 2, 3 и 4), при различных стратегиях увеличения вычислительных ресурсов (только блоки, только нити, либо пропорциональное увеличение и блоков, и нитей) наблюдаются разные закономерности.

1. Увеличение только числа блоков (табл. 2).

Время выполнения уменьшается почти линейно с ростом количества блоков. Ускорение S_{CUDA} близко к теоретическому. Следовательно эффективность E_{CUDA} тоже близко к теоретическому.

2. Увеличение только числа нитей (табл.3).

Время выполнения также сокращается по мере увеличения числа нитей. Ускорение S_{CUDA} растёт, однако с заметно меньшей скоростью, чем в первом случае. Эффективность также показывает устойчивую тенденцию к снижению – от 1 (при одной нити) до всего 0,146 (при 256 нитях). Это говорит о том, что при увеличении числа нитей рост параллелизма сопровождается значительными накладными расходами и ограничениями архитектуры, из-за чего производительность на одну «единицу ресурса» быстро падает.

3. Пропорциональное увеличение числа блоков и нитей (табл. 4).

Здесь достигается наибольший прирост производительности. Ускорение растёт быстрее, чем в двух предыдущих случаях: при 32×32 достигается ускорение более чем в 157 раз. Однако эффективность постепенно снижается, что объясняется ростом параллелизма выше оптимального уровня для данной задачи — накладные расходы на синхронизацию и распределение ресурсов становятся заметнее.

3.2. Параллельная версия с записью

Как видно (табл. 5, 6 и 7), при различных стратегиях увеличения вычислительных ресурсов (только блоки, только нити, либо пропорциональное увеличение и блоков, и нитей) наблюдаются следующие закономерности:

1. Увеличение только числа блоков (табл. 5).

Время выполнения уменьшается практически линейно с ростом количества блоков. Ускорение S_{CUDA} близко к теоретическому значению. Следовательно, эффективность E_{CUDA} тоже близко к теоретическому значению.

2. Увеличение только числа нитей (табл. 6).

Рост числа нитей также сокращает время выполнения, причём ускорение S_{CUDA} увеличивается довольно быстро. Однако эффективность E_{CUDA} заметно падает: при 256 нитях она составляет лишь $\sim 0{,}146$. Это связано с накладными расходами и

ограничениями архитектуры, что снижает производительность на каждую дополнительную нить.

3. Пропорциональное увеличение числа блоков и нитей (табл. 7).

При таком подходе достигается максимальное ускорение. Например, при конфигурации 256 нитей на блок достигается ускорение почти в 38 раз. Однако эффективность постепенно снижается, что объясняется ростом параллелизма выше оптимального уровня и возрастанием затрат на синхронизацию и управление ресурсами.

Следует отметить, что в проведённых экспериментах фиксировалось лишь $n = 10^5$ траекторий, и этого обычно достаточно для практических задач. При увеличении числа записываемых экспериментов, например до $n = 10^8$ при параметрах $b = 32$ и $t = 32$, время чистого моделирования составило 372,647мс, тогда как совокупное время с учётом записи достигло 2042,193мс. Таким образом, дополнительная задержка составила порядка 2 секунд. При масштабировании на большие значения n, t и b вклад операции записи в общее время выполнения возрастает.

Заключение

В работе были разработаны и протестированы две версии симуляционной программы: без записи данных и с записью в файл. Первая версия показала максимальную скорость вычислений, так как исключала задержки на операции ввода-вывода. При этом при увеличении числа блоков достигалось почти линейное ускорение, тогда как рост числа нитей сопровождался постепенным снижением эффективности из-за накладных расходов. Наилучшие результаты были получены при пропорциональном увеличении числа блоков и нитей: ускорение превысило 160 раз. При этом программа формировала статистику по количеству рассеяний, поглощений и вылетов, что обеспечивало возможность оперативной оценки тенденций и корректировки параметров моделирования.

Вторая версия программы позволяет не только собирать статистику по рассеяниям, поглощениям и вылетам, но и дополнительно записывать траектории частиц в файл. Это сделало её более удобной для практических исследований. Ускорение также возрастало, но максимальное значение составило около 38 раз, а эффективность снижалась при увеличении числа нитей и блоков. Дополнительные задержки, вызванные записью, были малозаметны при небольшом объёме данных, но становились существенными в большом числе экспериментов. Таким образом, удобно сначала использовать версию без записи для быстрой оценки общей картины, а затем запускать версию с записью в уже настроенной среде для детального анализа результатов.

Литература

1. **Ермаков С. М., Михайлов Г. А.** Курс статистического моделирования // Москва: Наука. 1976. 320с.
2. **Марчук Г. И., Михайлов Г. А., Назаралиев М. А. и др.** Метод Монте-Карло в атмосферной оптике / под ред. Г. И. Марчука // Новосибирск: Наука. 1976. 280с.
3. **Ермаков С. М., Михайлов Г. А.** Статистическое моделирование // Москва: Наука. 1982. 296с.
4. **Зеге Э. П., Иванов А. П., Кацев И. Л.** Перенос изображения в рассеивающей среде // Минск: Наука и техника. 1985. 327 с.
5. **Ландсберг Г. С.** Оптика. 5-е изд. // Москва: Наука. 1976. 928с.
6. **Mundy W. C., Roux J. A., Smith A. M.** Mie scattering by spheres in an absorbing medium // Journal of the Optical Society of America. 1974. Т. 64. № 12. С. 1593-1593.
7. **Grama A.** Introduction to Parallel Computing // Pearson Education. 2003. 636с.
8. **Сандерс Дж., Кэндрот Э.** Технология CUDA в примерах. Введение в программирование графических процессоров // Москва: ДМК-Пресс. 2018. 352с.