УДК 519.63

В. А. Седайкина

ОПТИМИЗАЦИЯ МЕТОДА FIM ДЛЯ РЕШЕНИЯ ТРЕХМЕРНОГО УРАВНЕНИЯ ЭЙКОНАЛА

Быстрый итерационный метод (FIM), описанный в статье [1], является удобным и действенным для решения уравнения эйконала для двумерных и небольших трехмерных задач. Однако для решения больших трехмерных задач требуется довольно продолжительное время и большое количество оперативной памяти. Приведена оптимизация этого метода, сокращающая объем хранимой в памяти информации и время, необходимое для решения задачи. Приведены численные эксперименты.

A Fast Iterative Method (FIM) for eikonal equations [1] are convenient and optimal in two-dimensional and small three-dimensional models. But it takes a lot of time and random access memory for big three-dimensional models. This article describes the time and memory optimization of this method for big 3d models. The results of simulations are presented.

Ключевые слова: уравнение эйконала, численное моделирование, оптимизация, быстрый итерационный метод.

Key words: eikonal equation, numerical modeling, optimization, fast iterative method.

Быстрый итерационный метод решения уравнения эйконала

Уравнение эйконала — нелинейное дифференциальное уравнение первого порядка

$$|\tau(x)|^2 - \frac{1}{v^2(x)} = 0, \forall x \in \Omega \quad (1)$$

где Ω — область в R^3; $\tau(x)$ — эйконал; $v(x)$ — функция скорости.

В [1] приведен быстрый итерационный метод (FIM) решения уравнения эйконала.

Краткое описание метода FIM

Дано: скоростная модель и координаты источника (ов).

Найти: решение уравнения (1) во всех точках сетки.

Метод можно разделить на следующие основные пункты.

1. Задается некоторое начальное приближение: точка источника и несколько точек, ближайших к ней, добавляются в «действующий список», представляющий собой массив, элементы которого будут пересчитаны на каждом шаге итерации;

2. Каждый элемент «действующего списка» пересчитывается способом, описанным в [1] и сравнивается со своим прошлым значением (полученным на предыдущей итерации). Если разница между этими значениями по модулю:

а) больше некоторой погрешности (погрешность выбирается пользователем), элемент остается в «действующем списке»;

б) меньше погрешности, то элемент покидает «действующий список». Затем пересчитываются не находящиеся в «действующем списке» соседи этого элемента в основных направлениях (4 соседа в 2D и 6 в 3D случаях). Соседи, чьи значения при пересчете оказались меньше, попадают в «действующий список».

3. Пункт 2 повторяется до тех пор, пока «действующий список» не опустеет. Тогда решение во всей области считается найденным с заданной точностью.

Эффективность данного алгоритма обоснована в статье [1].

Заметим, что узкое место алгоритма в расширяющемся «действующем списке», который постоянно пересчитывается; чем больше становится список, тем больше времени занимает одна итерация. Для двумерной задачи это не является критичным, однако при переходе к трехмерной задаче это становится сильно заметным. Также при решении больших трехмерных задач возникают проблемы, связанные с необходимостью хранения в оперативной памяти больших объемов информации. Рассмотрим этот вопрос подробнее.

Для решения уравнения (1) в области размером $N_x N_y N_z$ (где N_x, N_y, N_z — количество отсчетов по осям O_x, O_y, O_z) требуются:

1) массив C, содержащий скорость в среде, и массив T, содержащий решения уравнения (1). Эти массивы имеют одинаковый размер $4\,N_x N_y N_z$ (тип данных float) или $8\,N_x N_y N_z$ (тип данных double);

2) логический массив x_log (состоящий из 0 и 1), показывающий, находится ли точка области в «действующем списке» или нет. Размер массива $N_x N_y N_z$ (тип данных boolean);

3) массив L, содержащий «действующий список», либо динамический (его размер будет постоянно изменяться при решении задачи), либо статический (с предопределенным размером). В случае изменяющегося размера есть существенная потеря во времени при перезаписи массива, а в случае предопределенного размера — потеря в памяти, так как сразу используется максимально возможный размер, равный $4 N_x N_y N_z$ (тип данных integer). Дополнительно это значение должно быть умножено на 3, поскольку хранятся координаты (x, y, z) для каждой точки.

То есть, например, для решения уравнения (1) в области $801 \cdot 801 \cdot 601$ в оперативной памяти нужно хранить: $801 \cdot 801 \cdot 601 \cdot 4 \cdot 2$ байт (массивы C и T) + $801 \cdot 801 \cdot 601 \cdot 1$ байт (массив x_log) + $801 \cdot 801 \cdot 601 \cdot 3 \cdot 4$ байт (массив L максимально возможного размера) 8097646221 байт $\approx 7{,}5$ Гигабайт.

Это затраты только на хранение основных данных, не учитывая самих расчетов и вспомогательных данных. При таких затратах памяти программа очень сильно теряет в скорости или даже может не работать на некоторых компьютерах. Таким образом, требуется оптимизация, позволяющая уменьшить затраты памяти и увеличить скорость выполнения программы.

Оптимизация

Пусть расчетная область имеет форму прямоугольного параллелепипеда и находится в пределах $x \in [0, N_x]$, $y \in [0, N_y]$, $z \in [0, N_z]$.

Первый шаг оптимизации — это ограничение размера «действующего списка»

Сначала рассчитывается начальное приближение — это куб, в центре которого находится источник. Если источников несколько, рекомендуется выбрать «опорный источник», находящийся примерно посередине области. Расстояние от источника до любой из плоскостей куба равно $9N$ (N — выбранное пользователем число, величина которого определяется опытным путем из следующих соображений: при слишком большом N решение задачи даже в кубе с начальными значениям становится неоправданно долгим, а при слишком маленьком теряется точность. Оптимальным можно считать $N \in [3, 7]$). В этом кубе вычисляются значения уравнения (1) с помощью метода FIM без изменений.

Решение в этой области будет начальным приближением для запуска следующей итерационной схемы:

1. Пусть источник находится в координатах (x_0, y_0, z_0). Расчетная область увеличивается пропорционально по каждой координате в обе стороны на значение N. То есть если

$$x_1 \in [x_0 - 9N; x_0 + 9N], y_1 \in [y_0 - 9N; y_0 + 9N], z_1 \in [z_0 - 9N; z_0 + 9N]$$

суть границы области начального приближения, то границы новой увеличенной области будут следующими:

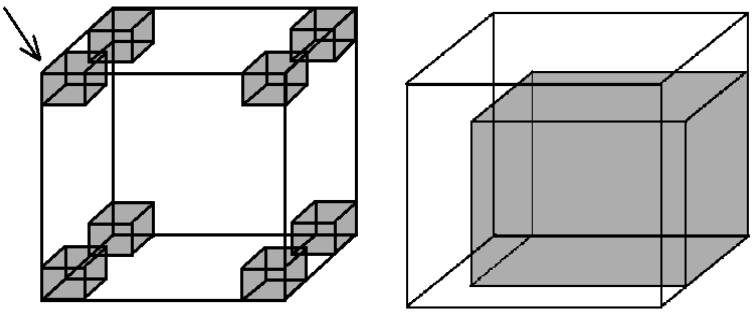
$$x_1\_new \in [\min(x_1) - N; \max(x_1) + N],\ y_1\_new \in [\min(y_1) - N; \max(y_1) + N],$$

$$z_1\_new \in [\min(z_1) - N; \max(z_1) + N].$$

В центре этой новой области находится куб с уже посчитанными значениями, а по бокам — неизвестные значения.

2. Далее расчет происходит по следующей схеме.

2.1. Рассчитываются 8 кубов в углах новой области (рис. 1, *а*).



а б

Рис. 1. Расчет начальных кубов

Размер одной грани каждого такого куба 5*N*. На рисунке 1, *б* изображен увеличенный куб, на который указывает стрелка на рисунке 1а. Серым цветом схематично изображено область, данные в которой посчитаны на предыдущем шаге, а белым — данные, которые должны быть рассчитаны на этом. Размер грани серого куба 4*N*.

Задача решается в каждом из этих восьми кубов. Так как они не пересекаются, эти вычисления можно выполнить параллельно.

2.2. Расчет прямоугольных параллелепипедов, расположенных на ребрах области.

Чтобы иметь возможность распараллеливания, предлагается сначала найти решение задачи в областях, указанных на рисунке 2, *а* (так как они не имеют пересечений), затем в областях, указанных на рисунках 2, *б* и 2, *в*

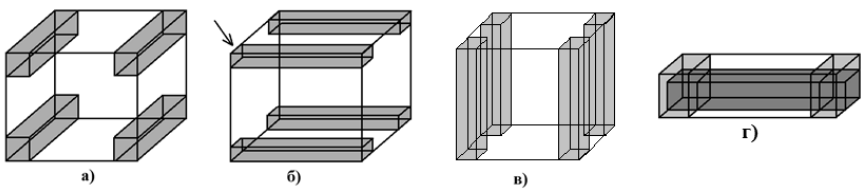


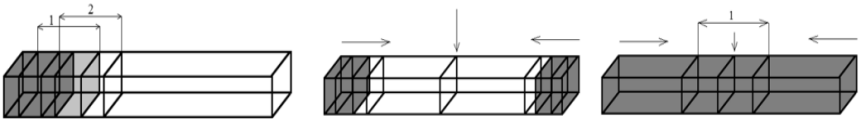
Рис. 2. Расчет прямоугольных параллелепипедов

На рисунке 2, *г* приведено схематическое изображение одной из этих областей (область со стрелкой на рис. 2, *б*). Здесь темно-серым цве-

том изображены значения, полученные на предыдущем шаге итерации, а светло-серым — значения, рассчитанные на текущем шаге итерации, в пункте 2.1, белым цветом — неизвестные значения.

Так как с каждым шагом итерации длина прямоугольных параллелепипедов будет увеличиваться вместе с областью, вычисление значений в них тоже должно быть разбито на части.

Область прямоугольного параллелепипеда, начиная с угла (значения в котором посчитаны в пункте 2.1) разбивается на небольшие области по следующему принципу: новая область имеет длину ребра 5*N*, причем из них 3*N* — известные значения, а 2*N* — неизвестные. На рисунке 3а светло-серым цветом изображены значения, которые должны быть посчитаны при решении задачи в области, отмеченной на рисунке 3а цифрой 1, а белым — в области 2. То есть для расчета новых значений область 2 использует известные данные, рассчитанные ранее в области 1 и в предыдущей области.



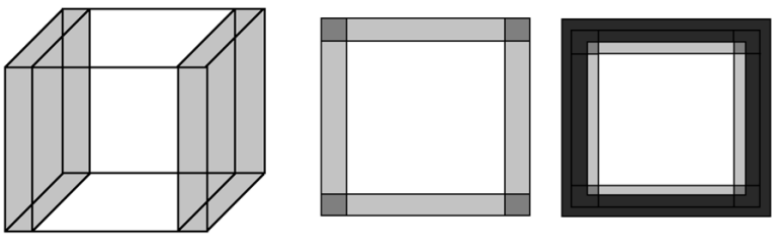
а б в

Рис. 3. Расчет отдельного параллелепипеда

Значения начинают рассчитываться от угловых кубов в сторону центра (схема на рис. 3, *б*). Когда значения во всем параллелепипеде рассчитаны, возле центра (с отступом на 5*N* в одну и в другую сторону) формируется общая область и значения в ней пересчитываются для выравнивания (рис. 3, *в*, область 1).

2.3. Расчет плоскостей.

Чтобы иметь возможность распараллелить вычисление, предлагается рассчитывать попарно независимые участки — две боковых, как на рисунке 4, *а*. Аналогично верхнюю и нижнюю, и переднюю и заднюю. По контуру этих областей известные значения, посчитанные на шаге 2.2, а с внутренней (относительно всей области) стороны — значения, полученные на предыдущем шаге итерации. Для каждой такой области повторяются шаги 2.1—2.2.



а б в

Рис. 4. Расчет плоскостей

Сначала снова считаются значения в угловых кубах, на рисунке 4б — темно-серые области (на рис. 4б и 4в изображены сечения в плоскости *YZ* при $x = \min(x_1)$), а затем — в прямоугольных параллелепипедах, светло-серые области на рисунке 4а. Далее область уменьшается (рис. 4в). Самый темный цвет — значения, рассчитанные на предыдущем шаге (рис. 4б). И шаги 2.1—2.2 повторяются, пока значения во всей области не будут рассчитаны.

3. Решение задачи в увеличенном кубе найдено, теперь этот куб становится начальным приближением:

$$x_1 = x_1\_new\ ,\ y_1 = y_1\_new\ ,\ z_1 = z_1\_new\ .$$

И повторяются шаги 1—3 до тех пор, пока расчетная область не увеличится до размеров исходной области:

$$x_1 \in [0; Nx]\ ,\ y_1 \in [0; Ny]\ ,\ z_1 \in [0; Nz]\ .$$

Второй шаг оптимизации — это разгрузка оперативной памяти

На первом шаге оптимизации размер «действующего списка» был уменьшен. Для сокращения размера остальных массивов предлагается следующая последовательность действий:

1) определить некоторую, не слишком маленькую глубину $N_{z\min}$, такую, что $N_{z\min} < N_z$. Например, $N_{z\min} = N_z / 6$. Выбрать число *M*, равное примерно $N_{z\min} / 3$ или $N_{z\min} / 4$;

2) задать размер матриц C, T, x_log равным $N_x N_z N_{z\min}$;

3) считать из файла, содержащего скоростную модель, данные, соответствующие координатам $x \in [0; N_x]$, $y \in [0; N_y]$, $z \in [0; N_z]$;

4) найти решение задачи в этой области;

5) сохранить значения матрицы **T** в файл;

6) для матриц **C** и **T** переместить значения $C_{x,y,z} = C_{x,y,N_{z\min}-M+z}$, $T_{x,y,z} = T_{x,y,N_{z\min}-M+z}$ для $x \in [0; N_x]$, $y \in [0; N_y]$, $z \in [0; M]$.

Остальные значения матрицы **T** (то есть для $x \in [0; N_x]$, $y \in [0; N_y]$, $z \in [M+1; N_{z\min}]$) сделать равными значениям по умолчанию, а для матрицы C – считать из файла значения, соответствующие координатам $x \in [0; N_x]$, $y \in [0; N_y]$, $z \in [Nz_{\min} + 1, 2 * Nz_{\min} - M]$;

7) сделать всю границу $z = M$ границей источников и подготовить область для решения в соответствии с первым шагом оптимизации, где размер расчетной области будет увеличиваться только по оси *Z* (так как по остальным осям область максимальна);

8) повторять шаги 4)—8) до тех пор, пока не будет исчерпана вся исходная расчетная область.

Таким образом, массивы C, T, x_log теперь имеют размер $N_x N_y N_z / 6$ (в зависимости от возможностей компьютера, размер может быть увеличен или уменьшен). Массив L размера $11^3 \cdot N^3 \cdot 3$, где $N \leqslant 7$.

Повторим произведенные ранее расчеты. Теперь для решения уравнения (1) в области размером 801·801·601 в оперативной памяти нужно хранить: 801·801·601/(6·4·2) байт (массивы C и T) + 801·801·601/(6·1) байт (массив x_log) + 77·77·77·3·4 байт (массив L), что составляет $583881697,5$ байт $\approx 0,5$ Гбайт. Это значение вполне реально для обычного персонального компьютера.

Если заменить тип float на тип double для повышения точности вычислений, в оперативной памяти будет храниться примерно $1,03$ Гб.

Численные эксперименты

Эксперимент с моделью размером 801·801·601 не может быть рассчитан для неоптимизированной задачи из-за нехватки памяти. Для оптимизированной задачи модель такого размера вычисляется в течение часа. Далее приведен эксперимент с задачей меньшего размера — 401·401·300 с точечным источником в координатах (201, 201, 0).

На рисунке 5 изображено сечение YZ при $x = 201$ (значение в отсчетах). Цветом изображена скорость в м/с.

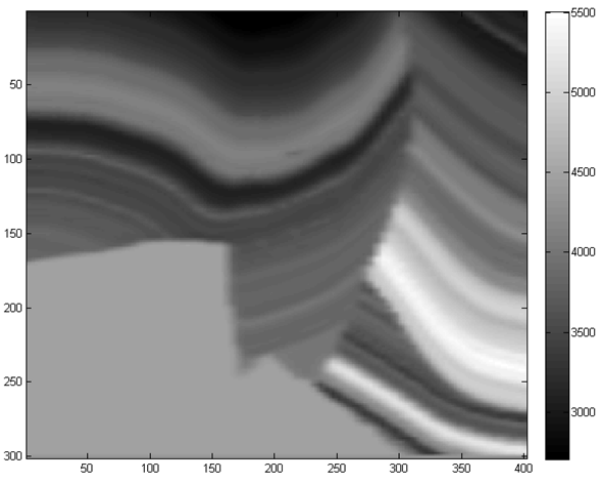
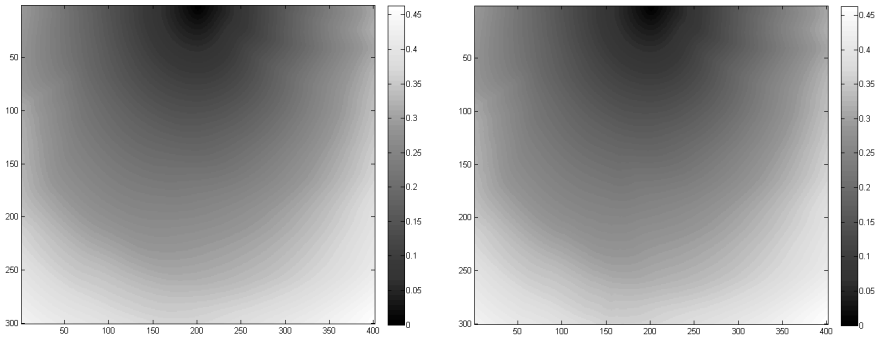


Рис. 5. Скоростная модель

Рисунок 6, *а* — это результат неоптимизированного решения, а рисунок 6, *б* — оптимизированного (изображение в том же сечении, что и на рисунке 5).

Относительная погрешность между этими вычислениями равна примерно 0,34%.

Время вычисления неоптимизированной задачи составляет 3 часа 45 секунд. Время вычисления оптимизированной задачи мало — 10 минут 49 секунд. Прирост во времени выполнения задачи большой — примерно в 16,7 раза.



а б

Рис. 6. Результаты решения

Список литературы

1. *Jeong W.-K., Whitaker R. T.* A Fast Iterative Method for Eikonal Equations. // SIAM Journal on Scientific Computing. 2008. Vol. 30, № 5. P. 2512–2534.

Об авторе

Валерия Александровна Седайкина — науч. сотр., Балтийский федеральный университет им. И. Канта, Калининград.
Email: VSedaikina@kantiana.ru

About the author

Valeria Sedaikina — researcher, I. Kant Baltic Federal University, Kaliningrad.
Email: VSedaikina@kantiana.ru