МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ
БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
ФИЗИЧЕСКИЙ ФАКУЛЬТЕТ
Кафедра компьютерного моделирования

Лешкевич Антон Сергеевич

ПАРАЛЛЕЛЬНЫЕ АЛГОРИТМЫ ЧИСЛЕННОГО РЕШЕНИЯ МНОГОМЕРНОЙ ЗАДАЧИ СТЕФАНА

Курсовая работа

Научный руководитель:
Старший преподаватель
Козловский А.Н.
Старший преподаватель
Тимощенко И.А.

Допущен к защите
«___» _____ 2020 г.
Зав. кафедрой компьютерного моделирования
Романов О.Г.

МИНСК, 2020

Оглавление

ВВЕДЕНИЕ 3

ЗАДАЧА СТЕФАНА 4

РАЗНОСТНАЯ СХЕМА 8

МНОГОПОТОЧНОСТЬ 9

РЕАЛИЗАЦИЯ АЛГОРИТМА 11

СИНХРОНИЗАЦИЯ ДАННЫХ 12

АНАЛИЗ РЕЗУЛЬТАТОВ 13

ВЫВОД 16

СПИСОК ИСПОЛЬЗУЕМОЙ ЛИТЕРАТУРЫ 17

ВВЕДЕНИЕ

Стремительный рост вычислительных мощностей современных компьютеров привёл к необходимости усовершенствования старых и разработки новых алгоритмов обработки данных, которые совершенно непригодны в современных реалиях в виду ряда недостатков, главный из которых – медленная обработка данных, которых с каждым днем становится все больше. Для расчета прикладных задач приходится иметь дело с огромными объемами данных, которые требуют порой сложных и монотонных вычислений, и это же, в короткие сроки. Именно здесь потребуется параллельная обработка, которая может разделить задачу на независимые подзадачи и обработать ее несколькими системами для быстрого вывода результатов [1]-[2]. В контексте данной работы рассмотрится задача о фазовом переходе и разработка параллельного алгоритма для ее решения. Соответственно, можно поставить перед собой следующие цели:

1. Изучение методов математического моделирования процессов плавления
2. Разработка и реализация параллельных алгоритмов решения трехмерной задачи Стефана
3. Анализ времени выполнения, ускорения, эффективности, стоимости и масштабируемости реализованных алгоритмов

ЗАДАЧА СТЕФАНА

В настоящей работе предлагается экономичная разностная схема сквозного счета для численного решения двухфазной задачи Стефана в случае трех пространственных переменных.

Схема сквозного счета характеризуется тем, что граница раздела фаз явно не выделяется и используются однородные разностные схемы. Основную роль при том играет принцип «размазывания» коэффициента теплоемкости по температуре, который тем самым не зависит от числа измерений. Явная схема для одномерной задачи рассматривалась в [3]. Неоднородные неявные схемы с явным выделением границы раздела фаз применялись в [4].

Для решения многомерного квазилинейного уравнения теплопроводности с размазанными коэффициентами теплоемкости и теплопроводности можно применить локально-одномерный метод, предложенный и обоснованный в [5], [6]. Он состоит в поэтапном решении по разным пространственным переменным одномерных уравнений теплопроводности при помощи безусловно устойчивых неявных схем.

При изучении тепловых процессов с фазовыми переходами вещества из одного состояния в другое приходится сталкиваться со следующей задачей. В каждой из двух или нескольких фаз справедливо уравнение теплопроводности:

$$c(u)\frac{\partial u}{\partial t} = \mathrm{div}\big(k(u)\mathrm{grad}(u)\big) + f$$

Где $u = u(r,t)$ – температура в момент времени t в точке с радиус-вектором $r = r(x_1, \dots, x_p)$, $k = k(u)$ – коэффициент теплопроводности, $c = c(u)$ – коэффициент теплоемкости (на единицу объема), $f = f(r,t)$ – плотность тепловых источников. Граница раздела фаз определяется условием, что температура вдоль этой границы равна температуре u^* фазового перехода, т. е. $u(r,t) = u^*$. Это соотношение является уравнением для определения $R(t)$ – положения границы фазового перехода в момент времени t. В общем виде можно считать, что уравнение границы фазового перехода имеет вид $\Phi(u) = 0$, где в качестве аргумента $u = u(r,t)$. Формулируем условие на границе фазового перехода. Пусть 1 — индекс фазы, у которой $u < u^*$ и 2 – индекс фазы у которой $u > u^*$. Так как $\mathrm{grad}(\Phi)$ направлен по нормали к поверхности раздела фаз, то нормальная составляющая теплового потока через нее равна:

$$Q_{1,2} = -\left(k\mathrm{grad}(u), \quad \frac{\mathrm{grad}(\Phi)}{|\mathrm{grad}(\Phi)|}\right)_{1,2}$$

Разность потоков $Q_1 - Q_2$ равна произведению энтальпии фазового перехода λ на нормальную компоненту скорости движения границы раздела фаз:

$$Q_1 - Q_2 = \lambda\left(\frac{dR}{dt}, \quad \frac{\mathrm{grad}(\Phi)}{|\mathrm{grad}(\Phi)|}\right)$$

Пользуясь тем, что вдоль границы раздела фаз:

$$\frac{d}{dt}\Phi(R(t),t) = \frac{\partial\Phi}{\partial t} + \left(\frac{dR}{dt}, \quad \mathrm{grad}(\Phi)\right) = 0$$

Можно записать:

$$u = u^*$$

$$\left(\left(k\mathrm{grad}(u)\right)_1 - \left(k\mathrm{grad}(u)\right)_2, \mathrm{grad}(\Phi)\right) + \lambda\frac{\partial\Phi}{\partial t} = 0 \text{ при } r = R(t)$$

Будем предполагать, что имеется только две фазы, так что

$$c(u) = \begin{cases} c_1(u) \text{ при } u < u^*, \\ c_2(u) \text{ при } u > u^* ; \end{cases} k(u) = \begin{cases} k_1(u) \text{ при } u < u^*, \\ k_2(u) \text{ при } u > u^* . \end{cases}$$

Так же будем предполагать, что функции достаточное число раз дифференцируемы и ограничены снизу постоянными m_1 и m_2, так что $c_s(u) \geq m_1 > 0$ и $k_s(u) \geq m_2 > 0$.

Так как при фазовом переходе энергия w как функция температуры испытывает скачок величины λ, которая называется теплотой (или энтальпией) фазового перехода. Поэтому можно написать:

$$w = \int_0^u c(u)du + \lambda\eta(u - u^*), \quad \eta(\zeta) = \begin{cases} 1, & \zeta \geq 0 \\ 0, & \zeta < 0 \end{cases}$$

Подставляя в это выражение уравнение энергии

$$\frac{\partial w}{\partial t} = \mathrm{div}\big(k\mathrm{grad}(u)\big) + f$$

И учитывая, что $\frac{d\eta(\zeta)}{d\zeta} = \delta(\zeta)$ есть дельта-функция Дирака, получим

$$(c(u) - \lambda\delta(u - u^*))\frac{\partial u}{\partial t} = \mathrm{div}\big(k(u)\mathrm{grad}(u)\big) + f \qquad (1)$$

Это уравнение позволяет решить задачу Стефана без явного выделения границы раздела фаз.

Ограничимся рассмотрением двухфазной задачи. Видно, что $c(u)$ и $\lambda\delta(u-u^*)$ входят в уравнение (1) одинаковым образом; $\lambda\delta(u-u^*)$ представляет собой сосредоточенную теплоемкость (на поверхности $u=u^*$). Для перехода к разностной схеме заменим дельта-функцию приближенно дельтаобразной, или размазанной, дельта-функцией $\delta(u-u^*,\Delta)\geq 0$, где Δ – величина полуинтервала, на котором отлична от нуля $\delta(u-u^*,\Delta)$. Это размазывание (сглаживание) эквивалентно замене на интервале $(u-u^*,\ u+u^*)$ разрывной функции $\eta(u-u^*)$ непрерывной функцией $\eta(u-u^*,\ \Delta)$ такой, что $\eta'(\zeta,\ \Delta)=\delta(\zeta,\Delta)$.

Итак, введем сглаженную теплоемкость $\check{c}(u)=c(u)+\ \delta(u-u^*,\Delta)$ из условий:

1. $\check{c}(u)=c_1(u)$, при $u<u^*-\Delta$
2. $\check{c}(u)=c_2(u)$, при $u<u^*+\Delta$
3. Изменение энтальпии на интервале $(u^*-\Delta,\ u^*+\Delta)$ сохраняется, т.е.

$$\int_{u^*-\Delta}^{u^*+\Delta}\check{c}(u)du\ =\int_{u^*-\Delta}^{u^*}c_1(u)du\ +\int_{u^*}^{u^*+\Delta}c_2(u)du\ +\lambda$$

Примеры сглаживания коэффициента теплоемкости приведены на рис. 1 и рис. 2.

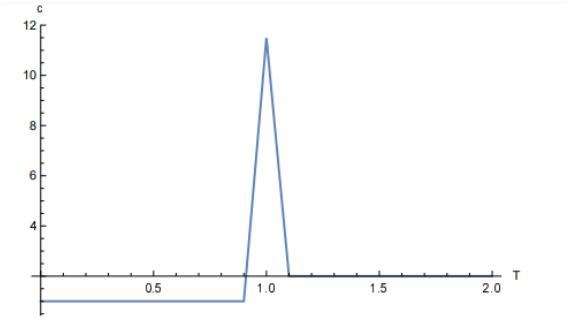

Рисунок 1. Линейно сглаженная теплоемкость

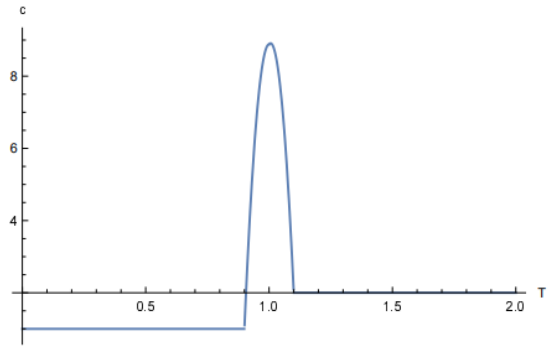

Рисунок 2. Теплоемкость, сглаженная параболами.

На том же интервале проводится сглаживание коэффициента теплопроводности. В итоге получается уравнение со сглаженными коэффициентами:

$$\check{c}(u)\frac{\partial u}{\partial t} = \mathrm{div}\left(\check{k}(u)\mathrm{grad}(u)\right) + f \qquad (2)$$

Дополняя граничными и начальными условиями, получим следующую задачу:

$$\check{c}(u)\frac{\partial u}{\partial t} = \mathrm{div}\left(\check{k}(u)\mathrm{grad}(u)\right) + f$$

$$u_\Gamma = \mu(t), u(x,0) = u_0(x), t \in [0,T] \qquad (3)$$

Сходимость задачи (3) при $\Delta \to 0$ доказана в [7].

РАЗНОСТНАЯ СХЕМА

Для решений системы нелинейных уравнений будем использовать итеративный подход [8]-[9]. Для того чтобы сделать уравнение линейным, коэффициенты будем считать с предыдущей итерации. В общем случае разностная схема для задачи (3):

$$y^{s+r/p}(0) = y^s, \qquad y^s(k) = y^s,$$

$$\tilde{c}\left(y^{s+r/p}(k)\right)\frac{y^{s+r/p}(k+1)-y^{s+(r-1)/p}(k+1)}{\tau} + \Lambda_r\left(y^{s+r/p}(k)\right)y^{s+r/p(k+1)} = f_r(x,t_s), \quad (4)$$

где k -номер итерации, r -измерение, относительно которого решается уравнение, p -количество измерений.

В данной работе решается трехмерная задача Стефана, систему для которой можно записать следующим образом:

1-ое измерение:

$$\tilde{c}\left(y^{s+1/3}(k)\right)\frac{y^{s+1/3}_{i1,i2.i3}(k+1) - y^{s}_{i1,i2,i3}(k+1)}{\tau} + \Lambda_1\left(y^{s+1/3}(k)\right)y^{s+1/3} = \frac{1}{3}f^{s+1/3}_{i1,i2,i3}$$

2-ое измерение:

$$\tilde{c}\left(y^{s+2/3}(k)\right)\frac{y^{s+2/3}_{i1,i2.i3}(k+1) - y^{s+1/3}_{i1,i2,i3}(k+1)}{\tau} + \Lambda_2\left(y^{s+2/3}(k)\right)y^{s+2/3} = \frac{1}{3}f^{s+2/3}_{i1,i2,i3}$$

3-ее измерение:

$$\tilde{c}\left(y^{s+1}(k)\right)\frac{y^{s+1}_{i1,i2.i3}(k+1) - y^{s+2/3}_{i1,i2,i3}(k+1)}{\tau} + \Lambda_3\left(y^{s+1}(k)\right)y^{s+1} = \frac{1}{3}f^{s+1}_{i1,i2,i3}$$

$$\Lambda_r(y)y_{i_r} = -\frac{1}{h_r}\left(\tilde{k}\left(y_{i_r+1/2}\right)\frac{y_{i_r+1}-y_{i_r}}{h_r} - \tilde{k}\left(y_{i_r-1/2}\right)\frac{y_{i_r}-y_{i_r-1}}{h_r}\right), r = 1,2,3$$

$$\tilde{k}\left(y_{i_r+1/2}\right) = \frac{\tilde{k}\left(y_{i_r+1}\right)+\tilde{k}\left(y_{i_r}\right)}{2}$$

МНОГОПОТОЧНОСТЬ

Многопоточность — свойство платформы (например, операционной системы, виртуальной машины и т. д.) или приложения, состоящее в том, что процесс, порождённый в операционной системе, может состоять из нескольких потоков, выполняющихся «параллельно», то есть без предписанного порядка во времени. При выполнении некоторых задач такое разделение может достичь более эффективного использования ресурсов вычислительной машины [1].

Такие потоки называют также потоками выполнения; иногда называют «нитями» или неформально «тредами».

Сутью многопоточности является квазимногозадачность на уровне одного исполняемого процесса, то есть все потоки выполняются в адресном пространстве процесса. Кроме этого, все потоки процесса имеют не только общее адресное пространство, но и общие дескрипторы файлов. Выполняющийся процесс имеет как минимум один (главный) поток.

Многопоточность (как доктрину программирования) не следует путать ни с многозадачностью, ни с многопроцессорностью, несмотря на то, что операционные системы, реализующие многозадачность, как правило, реализуют и многопоточность. К достоинствам многопоточной реализации той или иной системы перед многозадачной можно отнести следующее:

- Упрощение программы в некоторых случаях за счёт использования общего адресного пространства.
- Меньшие относительно процесса временные затраты на создание потока.

К достоинствам многопоточной реализации той или иной системы перед однопоточной можно отнести следующее:

- Упрощение программы в некоторых случаях, за счёт вынесения механизмов чередования выполнения различных слабо взаимосвязанных подзадач, требующих одновременного выполнения, в отдельную подсистему многопоточности.
- Повышение производительности процесса за счёт распараллеливания процессорных вычислений и операций ввода-вывода.

В случае, если потоки выполнения требуют относительно сложного взаимодействия друг с другом, возможно проявление проблем многозадачности, таких как взаимные блокировки.

В многопоточной среде часто возникают задачи, требующие приостановки и возобновления работы одних потоков в зависимости от

работы других. В частности, это задачи, связанные с предотвращением конфликтов доступа при использовании одних и тех же данных или устройств из параллельно исполняемых потоков. Для решения таких задач используются специальные объекты для взаимодействия потоков, такие как взаимоисключения (мьютексы), семафоры, критические секции, события и т.п. Многие из этих объектов являются объектами ядра и могут применяться не только между потоками одного процесса, но и для взаимодействия между потоками разных процессов.

Один из способов параллельного выполнения задач – это использования пула потоков. Пул потоков (рис. 3) представляет собой группу предварительно созданных потоков бездействия, которые готовы к работе. Они предпочтительнее, чем создавать новые потоки для каждой задачи, когда требуется большое количество коротких задач, а не небольшое количество длинных. Это предотвращает возникновение накладных расходов при создании потока большое количество раз [10].

Реализация, в общем случае, будет зависеть от среды, но в упрощенном варианте потребуется следующее:

- Способ создания потоков и удерживания их в состоянии ожидания. Это может быть достигнуто за счет того, что каждый поток будет ждать барьера, пока пул не запустит его.
- Контейнер для хранения созданных потоков, таких как очередь или любая другая структура, которая имеет способ добавить поток в пул и вытащить его.
- Стандартный интерфейс или абстрактный класс для потоков, используемых при выполнении работы. Это может быть абстрактный класс под названием

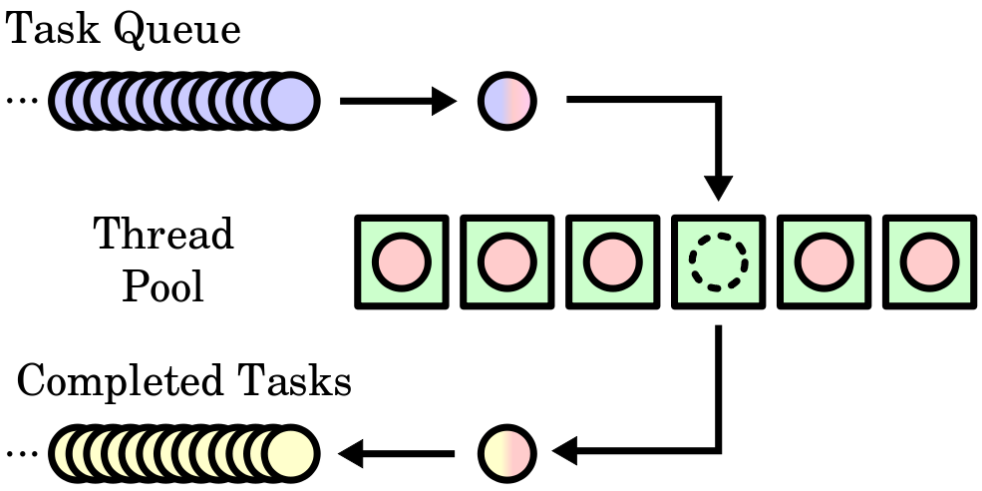

Рисунок 3. Общая структура пула потоков.

РЕАЛИЗАЦИЯ АЛГОРИТМА

Решение трехмерной задачи Стефана подразумевает последовательное решение локально-одномерных задач относительно каждого из трех измерений. Используя схему (4) можно итеративно решить задачу по алгоритму, представленному в следующей блок-схеме (рис. 4).

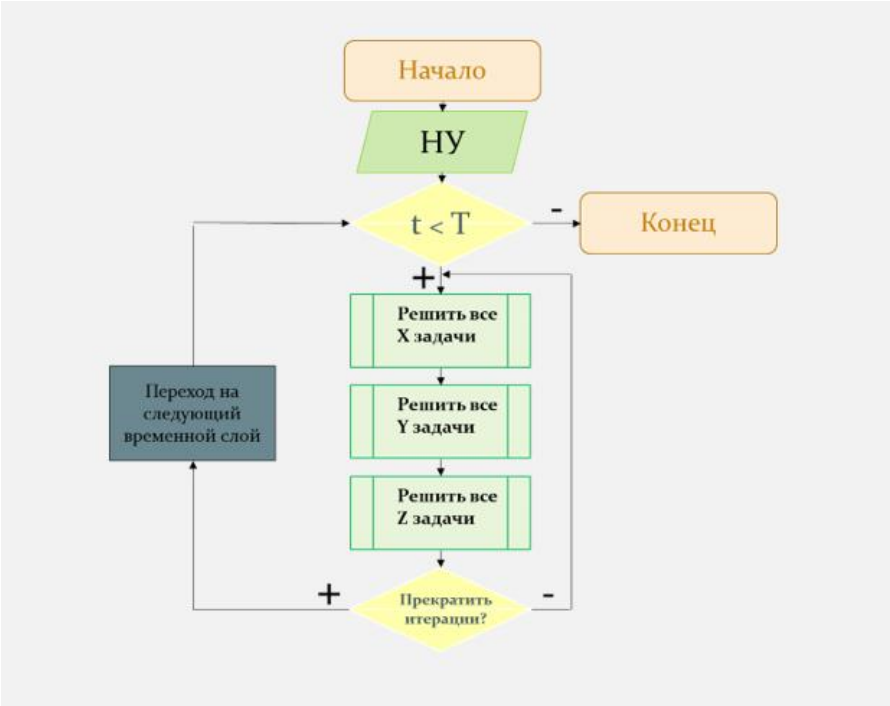

Рисунок 4. Блок схема алгоритма.

Параллельный вариант данного алгоритма подразумевает решение локально-одномерных задача в отдельных потоках. И хоть операционная система позволяет создать достаточно большое количество потоков в рамках одного процесса (их количество ограничено максимальным числом дескрипторов), физически одновременно могут выполняться столько потоков, сколько логических ядер имеется на выполняющем задачу процессоре. Поэтому гораздо целесообразней использовать пул потоков, в котором число рабочих потоков будет определяться характеристиками оборудования, на котором выполняется программа. Для этих целей в языке C++ можно воспользоваться функцией std::thread::hardware_concurrency()[11], вызов которой вернет максимально число потоков, которые могут выполняться физически одновременно.

Таким образом, для модификации уже предложенного алгоритма под параллельные вычисления достаточно лишь воспользоваться пулом потоков и позаботиться о синхронизации данных.

СИНХРОНИЗАЦИЯ ДАННЫХ

Рассмотрим работу одного рабочего потока, решающего локально-одномерную задачу. В процессе работы поток читает данные из одной области памяти и записывает их в другую. Однако в связи с тем, что область памяти для записи никак не пересекается с памятью, в которую пишут другие потоки, о синхронизации в данном случае можно не задумываться [12]. В случае с чтением, синхронизировать данные приходиться лишь при переходе к следующему измерению, для этого достаточно дождаться выполнения всех задач в рамках текущей загруженности пула. Так же, в процессе решения задачи, поток выделяет дополнительную память под свои нужны (к примеру, под прогоночные коэффициенты).

Еще одна возможная проблема – два потока пытаются выделить память и получают указатель на одну и ту же область памяти. Стандарт C++ говорит, что в таком случае все стандартные аллокаторы должны гарантировать потокобезопасность и исключать подобные ситуации [11]. В данной работе в качестве контейнеров использовался std::vector, определение которого:

```
std::vector<T, Allocator=std::allocator<T>>
```

Второй шаблонный аргумент по умолчанию есть стандартный STL аллокатор, который гарантирует потокобезопасность.

АНАЛИЗ РЕЗУЛЬТАТОВ

В данной работе, в качестве примера, была рассмотрена задача со следующими начальными и граничными условиями:

$$y^0 = 10$$

$$y^s_{0,j,k} = y^s_{i_1,j,k} = y^s_{i,0,k} = y^s_{i,i_2,k} = y^s_{i,j,0} = y^s_{i,j,i_3} = 0$$

$$i_1 = i_2 = i_3 = 100$$

$$L = 1, \qquad T = 1$$

$$k_1(u) = 0.2, \qquad k_2(u) = \sin(6u) + 5$$

$$c_1(u) = 2, \qquad c_2 = 20$$

$$u^* = 5, \qquad \lambda = 10, \qquad \Delta = 0.1$$

Результаты работы программы в момент времени $t = T$ представлены на рисунках 5-7.

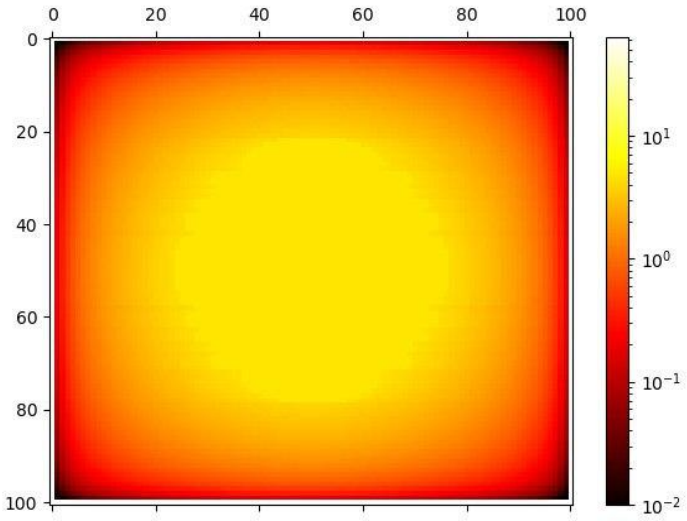

Рисунок 5. Распределение температур в сечении XY, Z=0.5

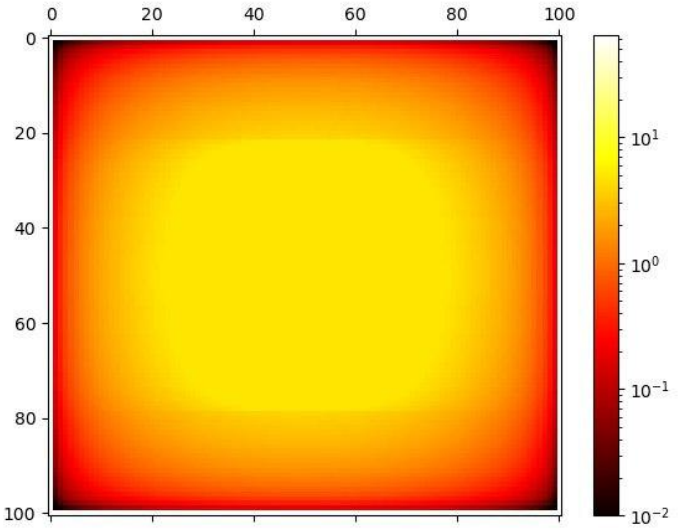

Рисунок 6. Распределение температур в сечении XZ, Y=0.5

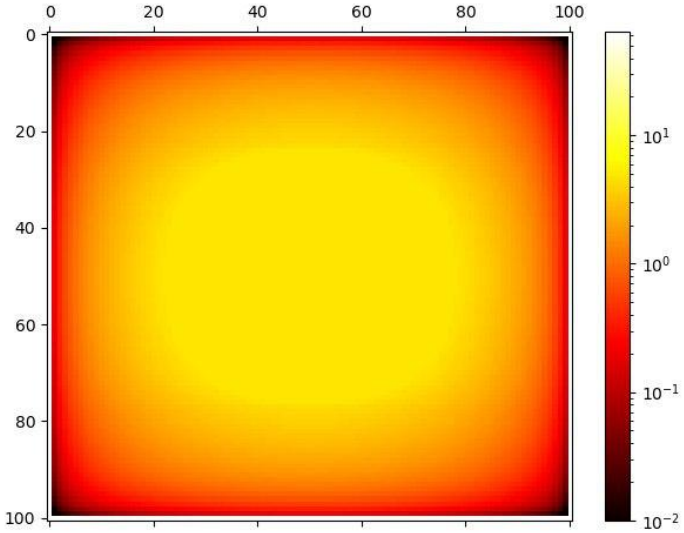

Рисунок 7. Распределение температур в сечении YZ, X=0.5

Зависимость скорости выполнения программы от количества рабочих потоков приведена на рисунке 8. Для наглядности график нарисован в логарифмическом масштабе.

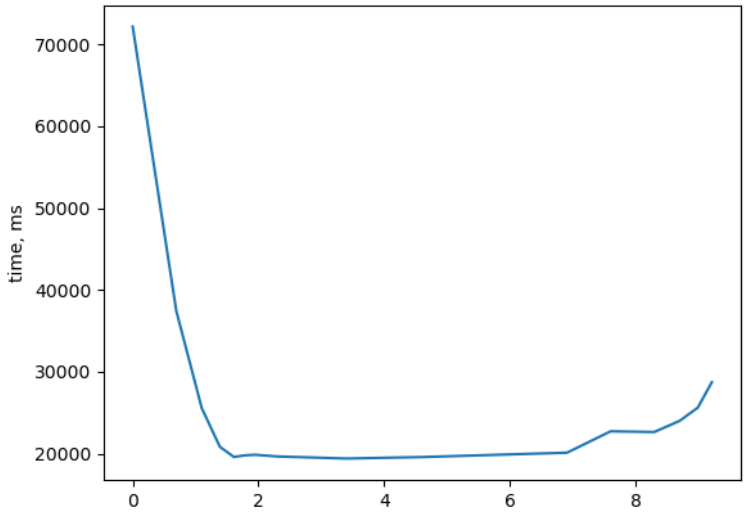

Рисунок 8. Зависимость времени выполнения от количества работающих потоков

Из рисунка 8 видно, что время выполнения программы быстро падает, доходя до некоторого порогового значения, которое в моем случае равно 4. (В машине, на которой производился тест стоит процессов Intel Core i5-2400, который имеет 4 физических ядра). Дальше, по мере увеличение числа работающих потоков время в среднем не изменяется, а при большом количестве даже увеличивается, что можно объяснить накладными расходами на переключение контекста между потоками. Исходя из результата можно смело сказать, что выбранная стратегия полностью себя оправдала и данным подходом удалось достичь максимальной производительности, если сравнивать с бессмысленным выделением большого объема вычислительных ресурсов. Исходный код программы приведен в [13].

ВЫВОД

В ходе выполнения работы был разработан и реализован параллельный алгоритм численного решения трехмерной задачи Стефана на основе локально-одномерной схемы. Главными достоинствами алгоритма являются простота реализации, а также значительное ускорение по времени выполнения. В связи с развитием современной техники, многоядерные процессоры стоят практически в любом домашнем компьютере, что увеличивает потенциальный круг пользователей, а простота реализации позволяет вносить изменения, характерные для специфических задач. Безусловно, результаты можно было бы улучшить, увеличивая количество физических ядер, однако для этой задачи больше подходят графические процессоры. Резюмируя, можно сказать, что данная реализация является компромиссом между скоростью, эффективностью и простотой, масштабируемостью.

СПИСОК ИСПОЛЬЗУЕМОЙ ЛИТЕРАТУРЫ

1. Уильямс, Э Параллельное программирование на C++ в действии. Практика разработки многопоточных программ. / Э. Уильямс — М.: ДМК Пресс, 2012. — 672с.
2. Миллер, Р. Последовательные и параллельные алгоритмы: Общий подход / Р. Миллер, Л. Боксер; пер. с англ. — М.: БИНОМ. Лаборатория знаний, 2006. — 406 с.
3. Каменомостская, С. Л.. О задаче Стефана. / С. Л. Каменомостская. Матем. сб., 1961, 53(95), № 4, 489—514.
4. Ehrlich, L. W. A numerical method of solving a heat flow problem with moving boundary. / L. W. Ehrlich. J. Assoc. Comput. Machinery, 1958, 5, № 2, 161—176.
5. Самарский, А. А. Об одном экономичном разностном методе решения многомерного параболического уравнения в произвольной области. / А. А. Самарский. //Ж. вычисл. матем. и матем. физ, 1962, 2, Ж 5, 787—811.
6. Самарский, А. А. Локально-одномерные разностные схемы на неравномерных сетках. / А. А. Самарский. Ж. вычисл. матем. и матем. физ., 1963, 3, № 3, 431—466.
7. Олейник, О. А. Об одном методе решения общей задачи Стефана. / О. А. Олейник. Докл. АН СССР, 1960, 135, № 5, 1054—1057.
8. Самарский, А. А. Уравнения параболического типа с разрывными коэффициентами и разностные методы их решения. Тр. Всес. совещания по дифф. ур-ниям. / А.А. Самарский. Ереван, Изд-во АН АрмССР, 1960, 148—160. (Ереван, ноябрь 1958)
9. Самарский, А. А. Однородные разностные схемы для нелинейных уравнений параболического типа. / А. А. Самарский. Ж. вычисл. матем. и матем. физ., 1962, 2, № 1, 25—56.
10. Джеффри, Р. Windows для профессионалов: создание эффективных Win32-приложений с учетом специфики 64-разрядной версии Windows / Р. Джеффри, 4-ое изд, 2001
11. International Standard ISO/IEC 14882:2017(E) – Programming Language C++, 2017
12. Методика разработки многопоточных приложений: принципы и практическая реализация / Intel //RSDN Magazine №3 — 2004
13. Лешкевич А.С., vphys project [Электронный ресурс] / А. С. Лешкевич; // Удаленный репозиторий GitHub – Режим доступа: https://github.com/webTer/vphys – Минск, 2020