РОССИЙСКИЙ УНИВЕРСИТЕТ ДРУЖБЫ НАРОДОВ

Факультет физико-математических и естественных наук

Кафедра прикладной информатики и теории вероятностей

Отчет по лабораторной работе № 2

Дисциплина: Параллельное программирование

Студент: Логинов Сергей Андреевич

Группа: НФИбд-01-18

МОСКВА 2021г

Задание

1. Используя средства OpenMP для редукции действий с массивами, реализуйте параллельные функции по нахождению суммы, минимума и максимума некоторого массива.

   Протестируйте быстродействие параллельных функций, постройте графики.

2. Используйте формулу трапеции для реализации функции trapezoidal, которая аппроксимирует значение интеграла от скалярной функции *f(x)*.

$$\int_a^b f(x)dx = \frac{h}{2}(f(a) + f(b)) + \sum_{i=1}^{n-1} f(x_i)$$

Задание 1

В файле reduction.f90 допишем код для описания трех функций: сумма, максимум и минимум массива:

```
function omp_sum(A, ths_num) result(S)
    implicit none
    include "omp_lib.h"
    real(real64), dimension(1:), intent(in) :: A
    integer(int32), intent(in) :: ths_num
    real(real64) :: S
    integer :: i, th ! счетчик и переменная для хранения номера потока
    real(real64), allocatable :: arr_for_sum(:) ! массив для суммы
каждого потока

    if( .not. allocated(arr_for_sum) ) allocate(arr_for_sum(1:ths_num))
! выделяем память с проверкой

    !$omp parallel private(th, i) num_threads(ths_num) ! для каждого
потока будут свои экземпляры th и i
```

```
    th = omp_get_thread_num() + 1 ! к номеру потока доабив единицу для упрощения работы в цикле

    do i = 1, size(A), ths_num ! шаг - количество потоков
      arr_for_sum(th) = arr_for_sum(th) + A(i) ! к элементам массива каждый поток с каждой итерацией прибавляет свой элемент
    end do

    !$omp end parallel

    S = sum(arr_for_sum) ! находим сумму массива, в котором элементы - результаты суммирования каждого потока


  end function omp_sum
```

```
function omp_max(A, ths_num) result(M)
    implicit none
    include "omp_lib.h"
    real(real64), dimension(1:), intent(in) :: A
    integer(int32), intent(in) :: ths_num
    real(real64) :: M
    integer(int64) :: length, i
    real(real64), allocatable :: arr_for_max(:) ! массив для хранения максимумов из каждого потока

    if( .not. allocated(arr_for_max) ) allocate(arr_for_max(ths_num)) ! выделение памяти


    !$omp parallel private(i, length) num_threads(ths_num)

    length = omp_get_thread_num()+1 ! упрощаем работу в цикле
    do i = 1, ths_num, ths_num
```

```
      arr_for_max(length) = A(length) ! заполняем массив для максимумов первыми элементами массива A для дальнейшего сравнения
    end do
    do i = 1, size(A) - ths_num, ths_num ! цикл сравнения элементов вида a(n) и a(n+1) но с учетом выполнения в потоках
      if (arr_for_max(length) < A(length + ths_num)) then
        arr_for_max(length) = A(length + ths_num)
      end if
    end do
    !$omp end parallel
    M = maxval(arr_for_max) ! находим максимальное значение из тех, которые нашел каждый поток
  end function omp_max
```

```
! минимум находим по тому же принципу, что и максимум
  function omp_min(A, ths_num) result(M)
    implicit none
    include "omp_lib.h"
    real(real64), dimension(1:), intent(in) :: A
    integer(int32), intent(in) :: ths_num
    real(real64) :: M
    integer(int64) :: length, i
    real(real64), allocatable :: arr_for_min(:)

    if( .not. allocated(arr_for_min) ) allocate(arr_for_min(ths_num))

    !$omp parallel private(i, length) num_threads(ths_num)

    length = omp_get_thread_num()+1
    do i = 1, ths_num, ths_num
      arr_for_min(length) = A(length)
    end do
    do i = 1, size(A) - ths_num, ths_num
      if (arr_for_min(length) > A(length + ths_num)) then
        arr_for_min(length) = A(length + ths_num)
```

```
          end if
        end do
        !$omp end parallel
        M = minval(arr_for_min)
      end function omp_min
```

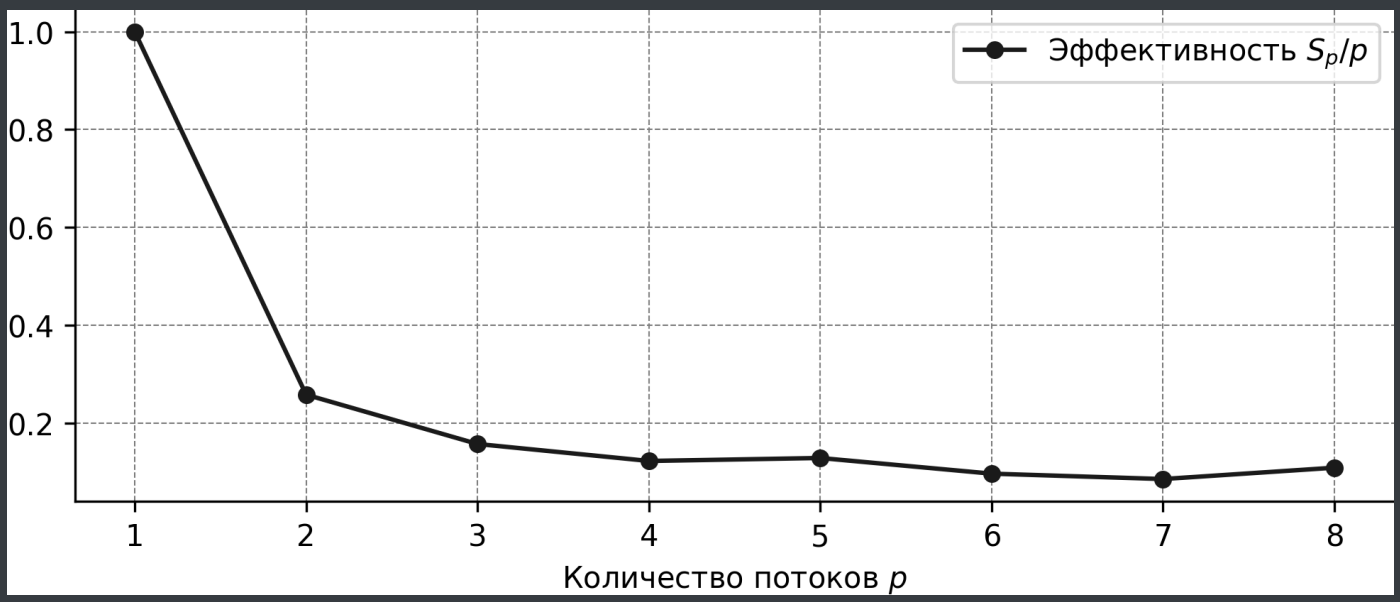
Запустим тест три раза. Простестируем сумму, максимум и минимум:

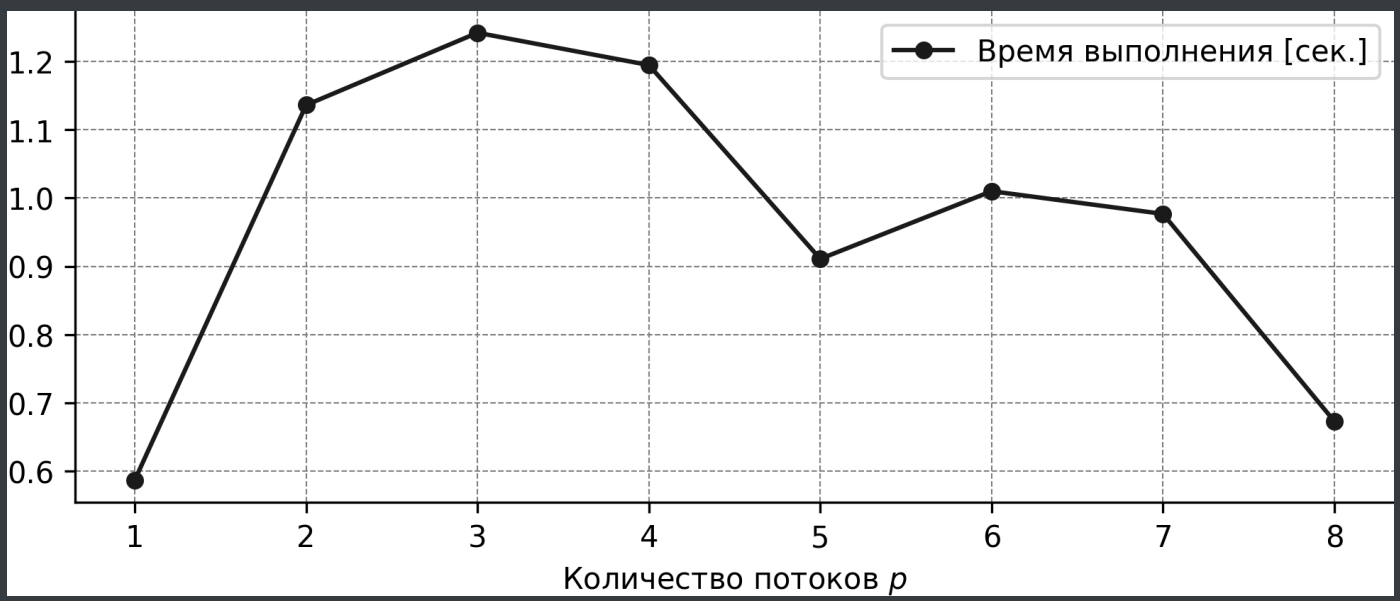
```
(base) sergejloginov@MacBook-Air-Sergej test %gfortran -fopenmp test_reduction.f90
(base) sergejloginov@MacBook-Air-Sergej test % ./a.out sum
# Тестируем суммирование
#  th,         Avg. time
   1,0.34759740000008604
   2,0.18592650000064168
   3,0.12973780000174884
   4,0.98030799996922713E-1
   5,0.10284229999815579
   6,0.10550530000182333
   7,0.10931439999549183
   8,0.11107679999840912
   9,0.12155430000275373
  10,0.13252630000060889
  11,0.13845989999826996
  12,0.15018019999552051
  13,0.15967139999847857
  14,0.16015609999594743
  15,0.17542090000060853
  16,0.21670929999963845
(base) sergejloginov@MacBook-Air-Sergej test % ./a.out max
# Тестируем нахождение максимума
#  th,         Avg. time
   1,0.20586290000064766
   2,0.10517140000010840
   3,0.72303599998122081E-1
   4,0.55207099998369813E-1
   5,0.50193100000615234E-1
   6,0.46131099999183786E-1
   7,0.42949500001850535E-1
   8,0.40869000001111995E-1
   9,0.44081600001663904E-1
  10,0.42123099998570980E-1
  11,0.44219800000428224E-1
  12,0.46759499999461697E-1
  13,0.44380999996792528E-1
  14,0.41468199997325428E-1
  15,0.41246399999363345E-1
  16,0.41461999996681695E-1
(base) sergejloginov@MacBook-Air-Sergej test % ./a.out min
# Тестируем нахождение минимума
#  th,         Avg. time
   1,0.20731419999792705
   2,0.10513799999898765
   3,0.72373500000685445E-1
   4,0.56149099997128360E-1
   5,0.49868999997090925E-1
   6,0.46408700002939438E-1
   7,0.42971299999044268E-1
   8,0.43681799998739730E-1
   9,0.45608900001388973E-1
  10,0.44493699999293312E-1
  11,0.42266900002141478E-1
  12,0.41981900000246243E-1
  13,0.43317400003434155E-1
  14,0.48799299998790957E-1
  15,0.46761399999377319E-1
  16,0.45669599995017053E-1
```

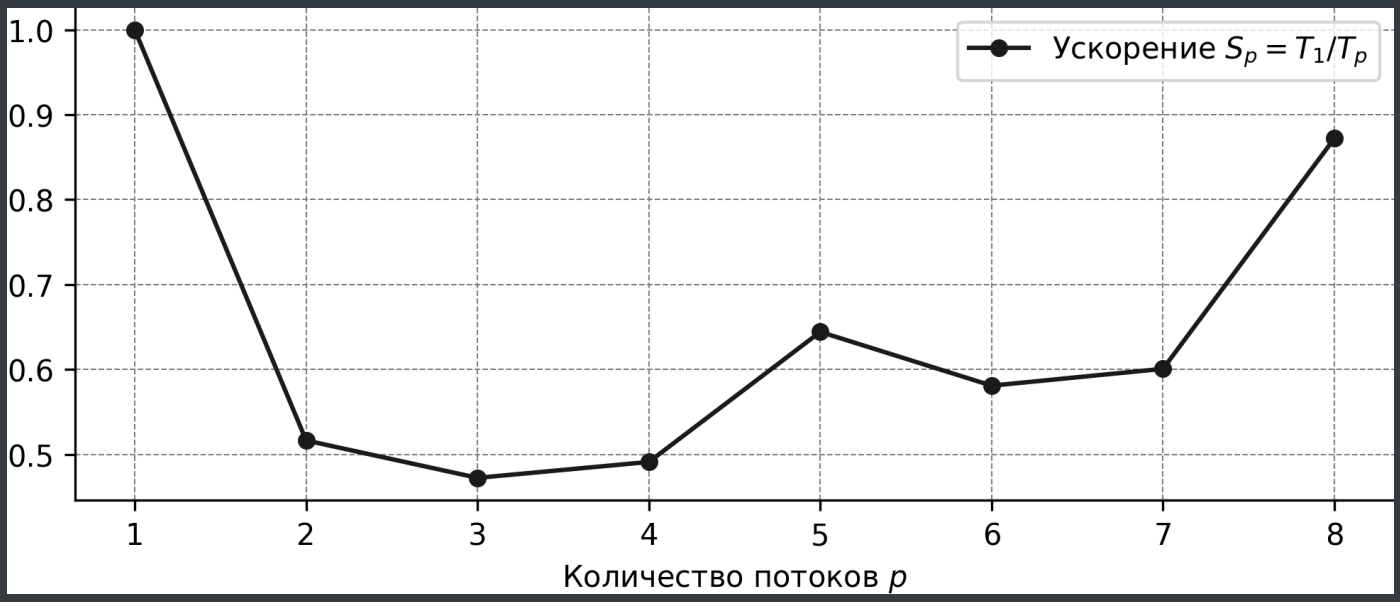
Для построения графиков мне пришлось воспользоваться другим устройством(windows-машина), в котором при компиляции появилось сообщение о несоотвествии числа потоков в программе и оптимальным для устройства. Изменил число потоков в тесте на 8 и получил следующие графики:

```
C:\Users\panda\lab02-stud\test>test_r.exe sum | python plot.py
Считали данные, строим графики... сохраняем изображения
img\ существует

C:\Users\panda\lab02-stud\test>test_r.exe max | python plot.py
Считали данные, строим графики... сохраняем изображения
img\ существует

C:\Users\panda\lab02-stud\test>test_r.exe min | python plot.py
Считали данные, строим графики... сохраняем изображения
img\ существует
```

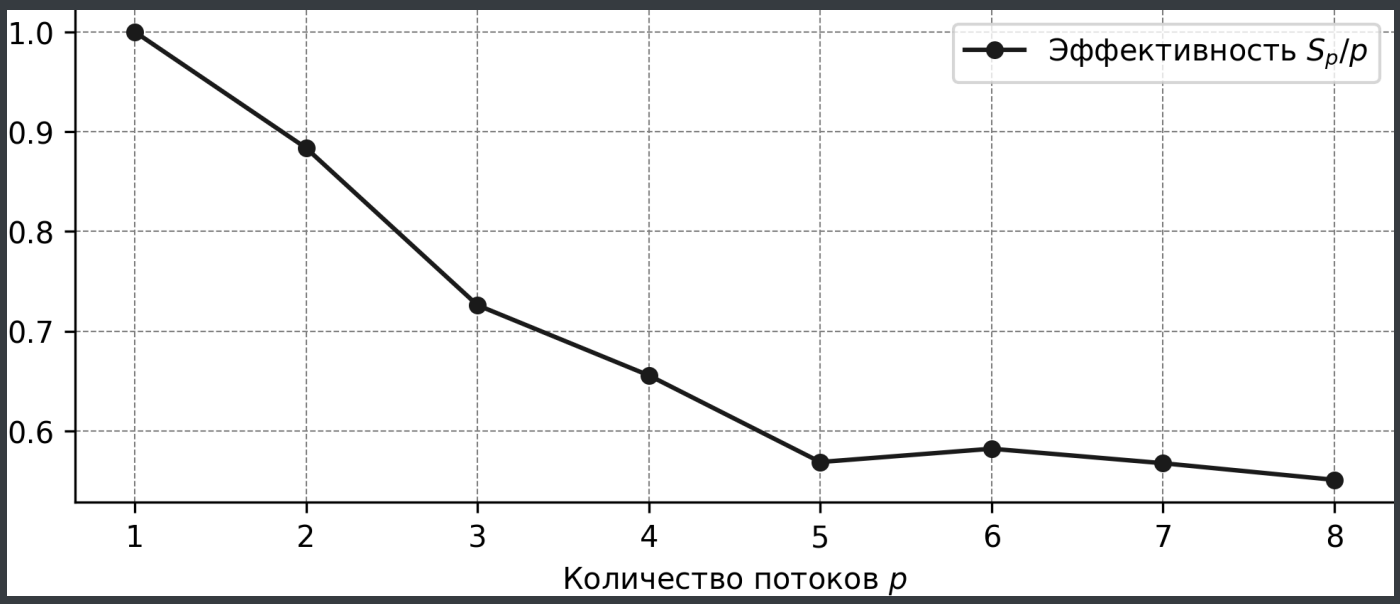
Функция omp_sum:

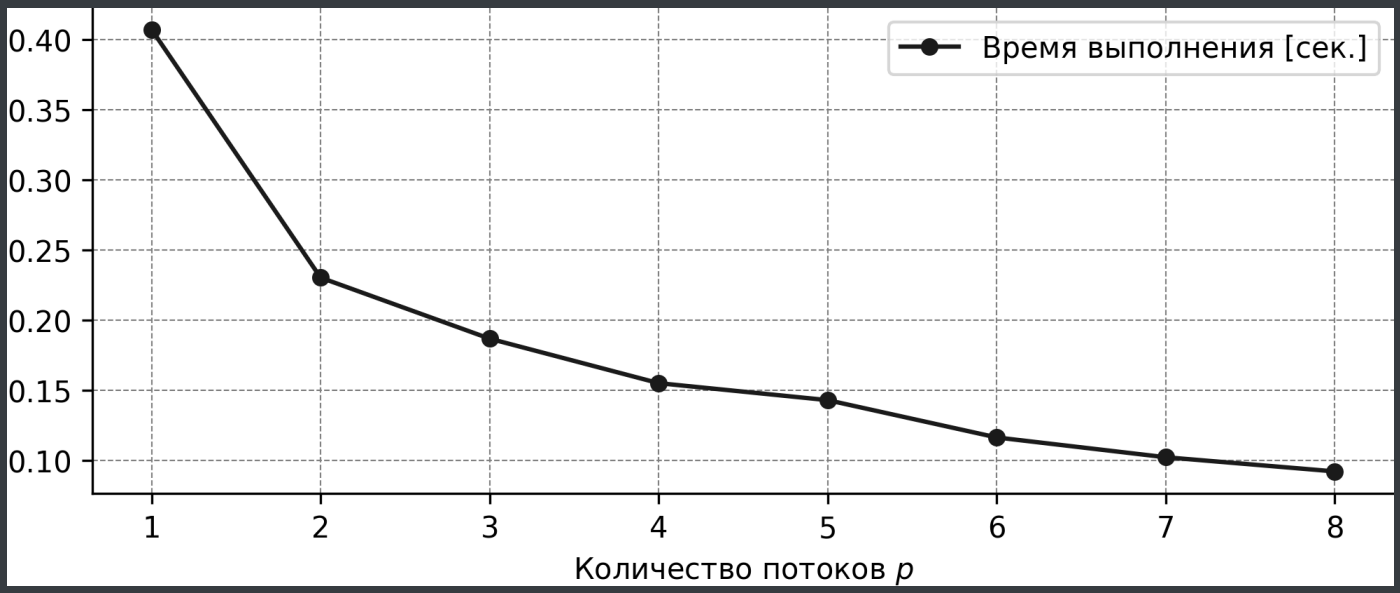


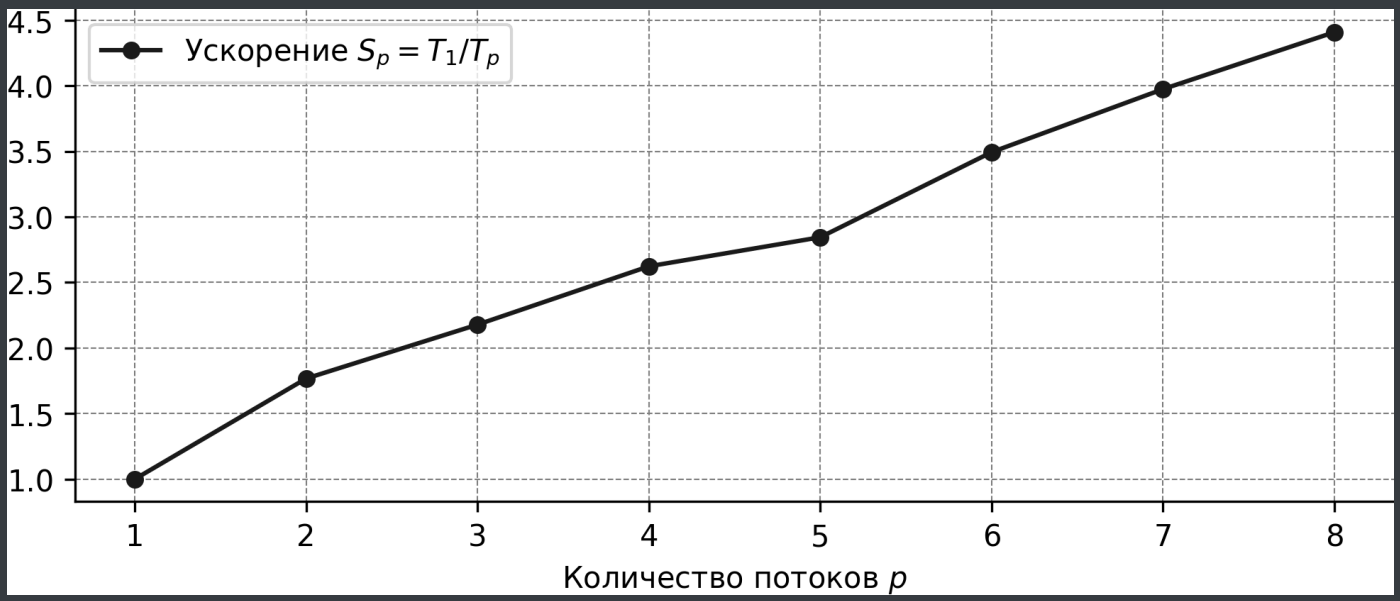




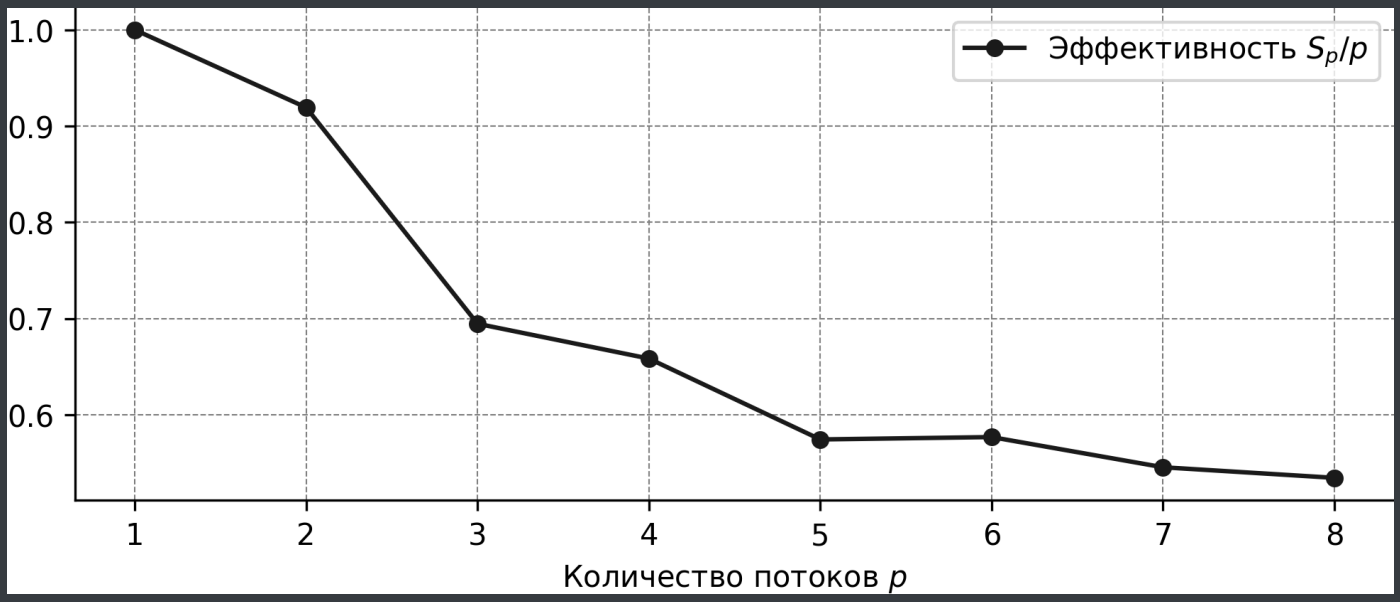
Функция omp_max:

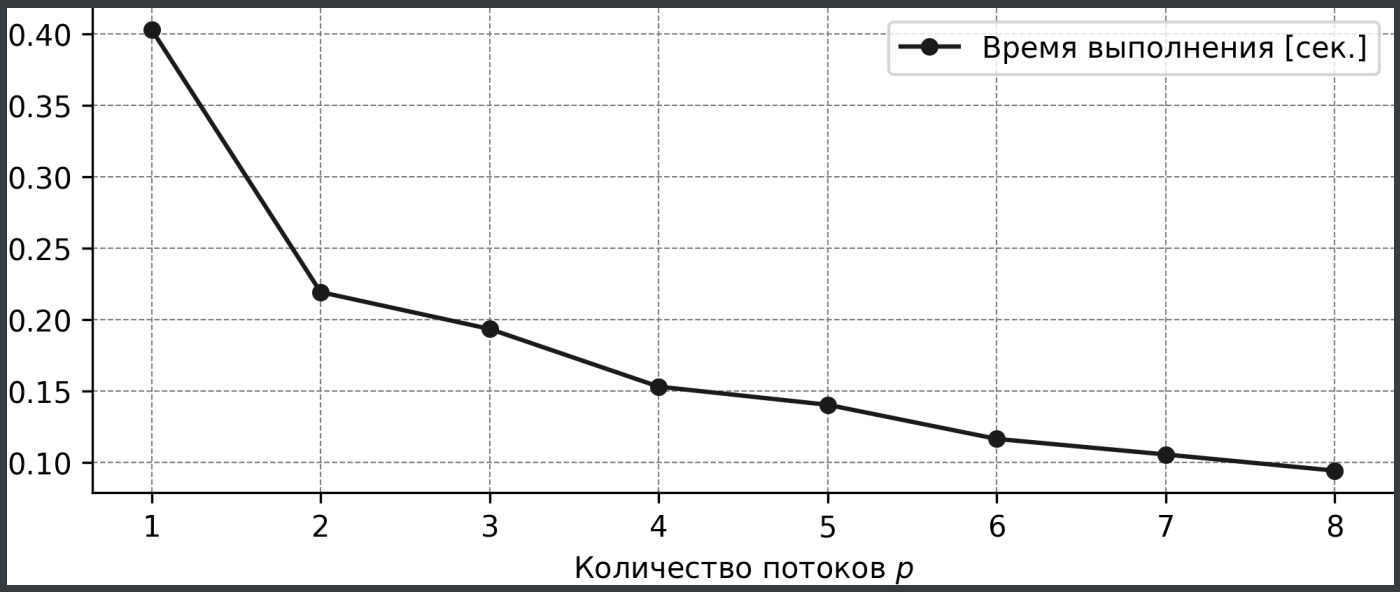


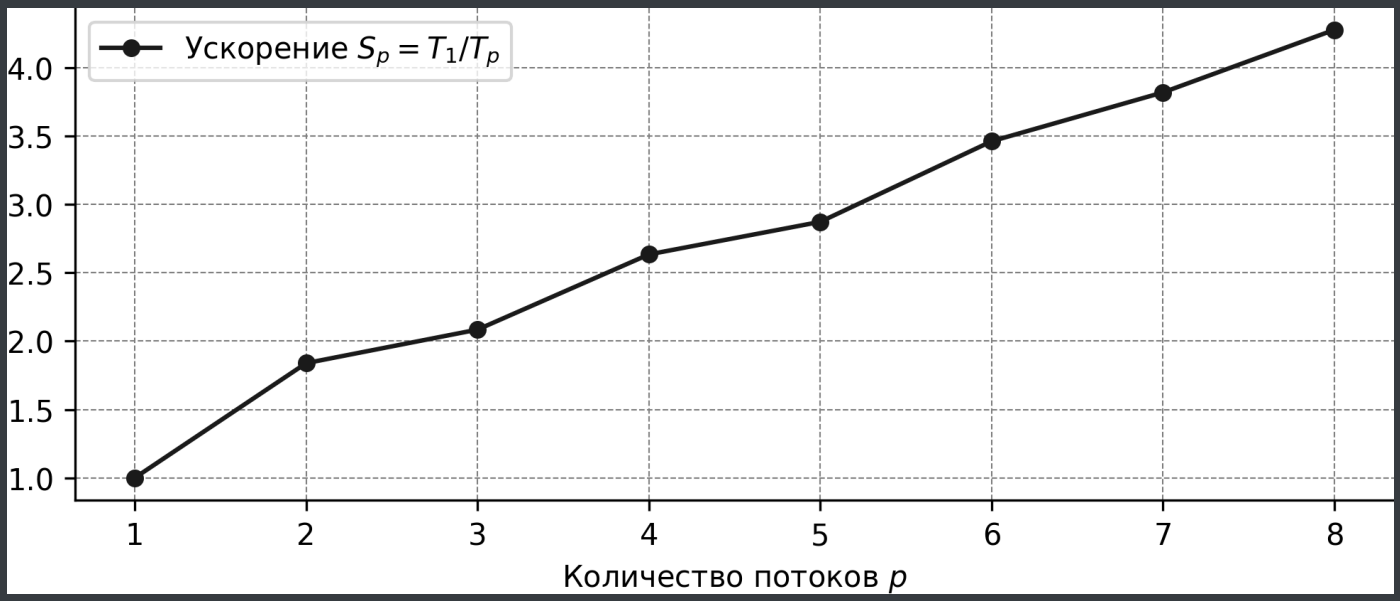




Функция omp_min:


Эффективность S_p/p
1.0
0.9
0.8
0.7
0.6
1
2
3
4
5
6
7
8
Количество потоков p


Время выполнения [сек.]
0.40
0.35
0.30
0.25
0.20
0.15
0.10
1
2
3
4
5
6
7
8
Количество потоков p


Ускорение $S_p = T_1/T_p$
4.0
3.5
3.0
2.5
2.0
1.5
1.0
1
2
3
4
5
6
7
8
Количество потоков p

2. В файле trapezoidal_rule.f90 допишем код в соответствии с формулой

$$\int_a^b f(x)dx = \frac{h}{2}(f(a) + f(b)) + \sum_{i=1}^{n-1} f(x_i)$$

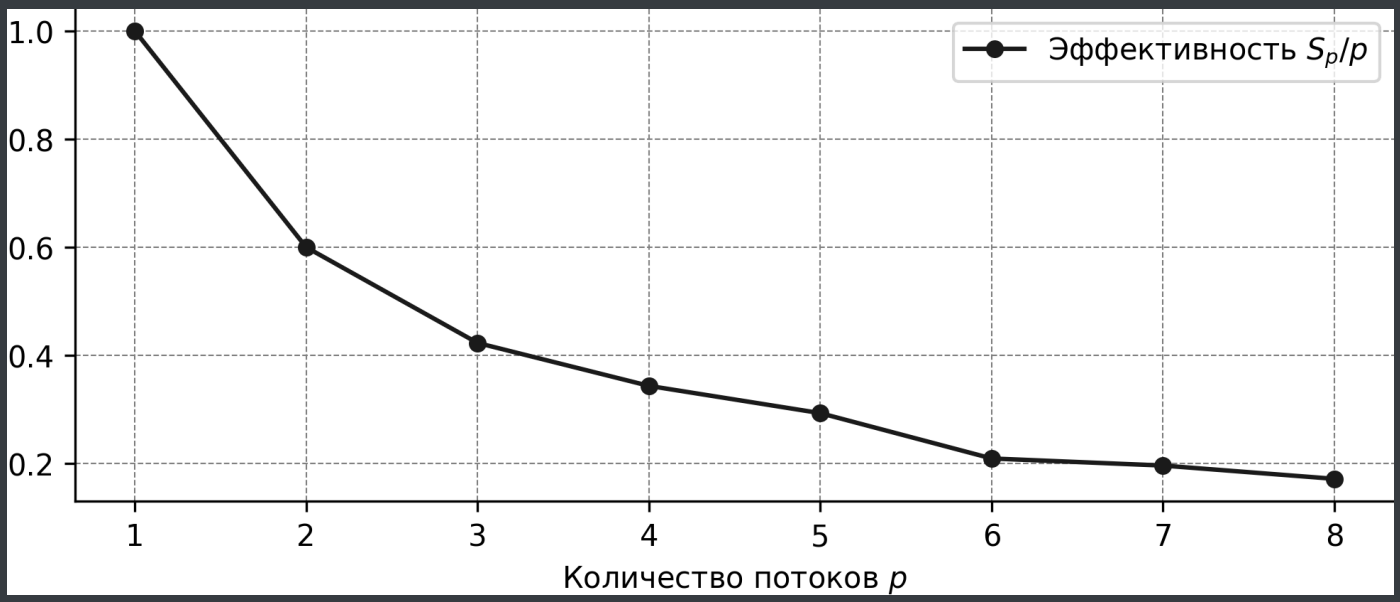
```
function trapezoidal(func, a, b, n, threads_num) result (res)
    implicit none
    include "omp_lib.h"
    procedure(f) :: func
    real(real64), intent(in) :: a, b
    integer(int64), intent(in) :: n
    integer(int64), intent(in) :: threads_num
    real(real64) :: h, res
    integer :: i, th
    real(real64), allocatable :: arr_tr(:)

    if( .not. allocated(arr_tr) ) allocate(arr_tr(threads_num))
    h = (b - a)/n ! вычисляем шаг как разница между верхним и нижним
пределом
    !интегрирования деленная на число точек разбиения
    !$omp parallel private(i) num_threads(threads_num)
    th = omp_get_thread_num() + 1
    do i = 1, n-1, threads_num
      arr_tr(th) = arr_tr(th) + func(a + i * h) * h ! суммируем, res
общая для всех потоков
    end do
    !$omp end parallel
    res = sum(arr_tr) + h / 2 * (func(a) + func(b)) ! дописываем
формулу, получив значение суммы


  end function trapezoidal
```

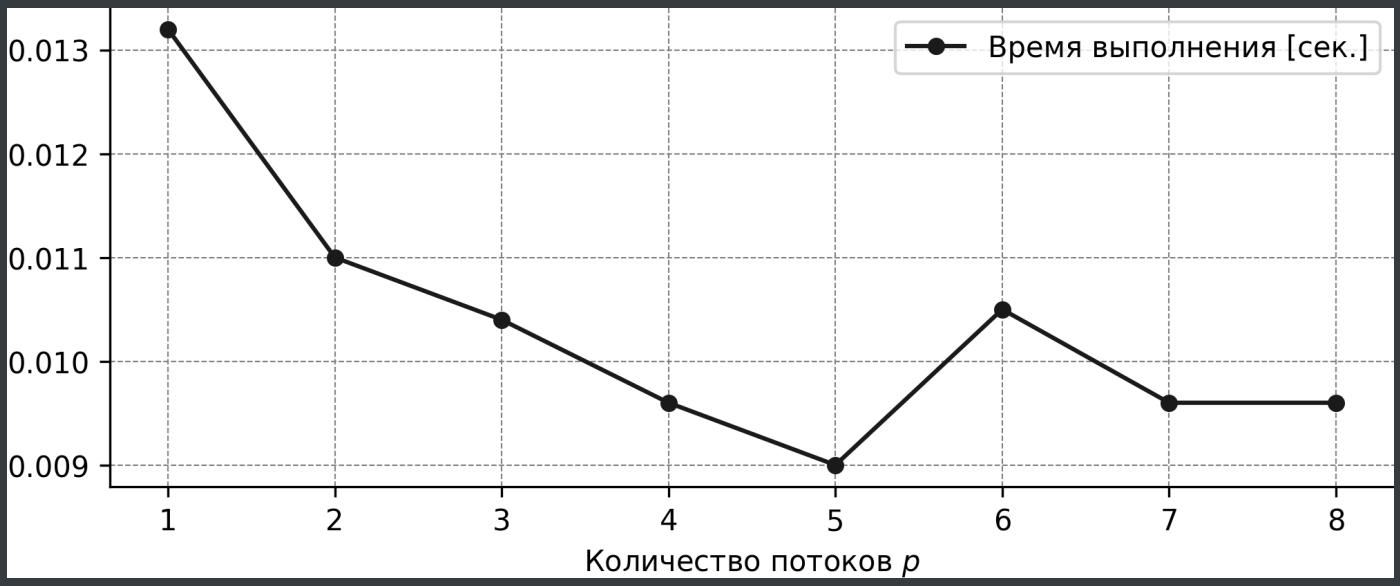
Запустим тест:

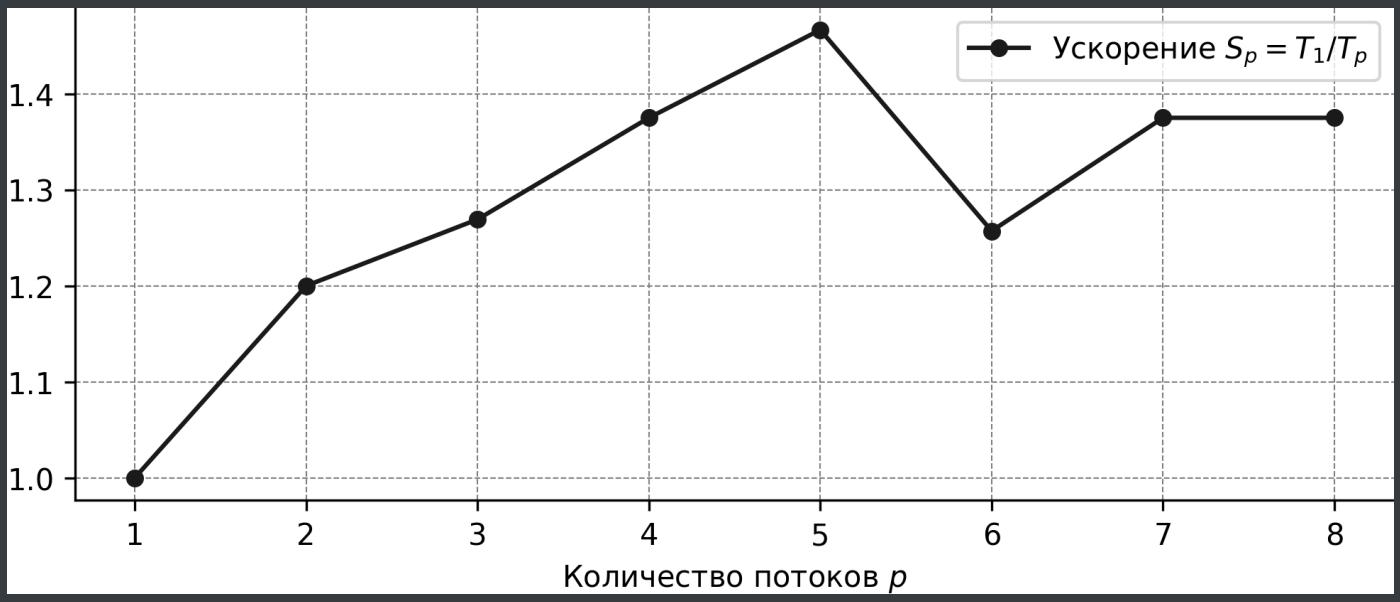
```
(base) sergejloginov@MacBook-Air-Sergej test % gfortran -fopenmp test_trapezoidal.f90
(base) sergejloginov@MacBook-Air-Sergej test % ./a.out
#  th,         Avg. time,        abs. error,             error
   1,0.78410999994957820E-2,15.999984000000147,7.9999920000000735
   2,0.23606999980984255E-2,37.003904867523609,18.501952433761804
   3,0.15271999989636242E-2,1.3287194912720000,0.66435974563599998
   4,0.16181999992113561E-2,0.95337996843600026,0.47668998421800013
   5,0.17071999987820165E-2,0.27275524676400620,0.13637762338200310
   6,0.15194000006886199E-2,6.9841943678479943,3.4920971839239972
   7,0.14225999999325724E-2,0.37801242987999961,0.18900621493999981
   8,0.13452999992296100E-2,0.13025487341999997,0.65127436709999986E-1
   9,0.14955000020563602E-2,0.80468254343599721,0.40234127171799861
  10,0.14342000009492039E-2,3.7417947956599988,1.8708973978299994
  11,0.13766999996732920E-2,0.90704056709600001,0.45352028354800000
  12,0.13082000048598274E-2,0.14699301907600004,0.73496509538000021E-1
  13,0.14202000020304695E-2,3.8196758188239972,1.9098379094119986
  14,0.13994999986607581E-2,1.6384280218720004,0.81921401093600021
  15,0.13982000004034490E-2,0.94954056163999967,0.47477028081999983
  16,0.13796000013826413E-2,4.9172003570479941,2.4586001785239970
```

С построением графиков вышли такие же проблемы:

```
C:\Users\panda\lab02-stud\test>test_tr.exe | python plot.py
Вы задали неоптимальное (16) количество потоков!
Ваш процессор эффективно поддерживает лишь 8 потоков.

C:\Users\panda\lab02-stud\test>gfortran -fopenmp test_trapezoidal.f90 -o test_tr.exe

C:\Users\panda\lab02-stud\test>test_tr.exe | python plot.py
Считали данные, строим графики... сохраняем изображения
img\ существует
```




Время выполнения [сек.]
0.013
0.012
0.011
0.010
0.009
1
2
3
4
5
6
7
8
Количество потоков p


Ускорение $S_p = T_1/T_p$
1.4
1.3
1.2
1.1
1.0
1
2
3
4
5
6
7
8
Количество потоков p