МОСКОВСКИЙ АВИАЦИОННЫЙ ИНСТИТУТ
(НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ)

Институт №8 «Компьютерные науки и прикладная математика»
Кафедра 806 «Вычислительная математика и программирование»

Лабораторная работа №*4*
по курсу «Программирование графических процессоров»

Работа с матрицам. Метод Гаусса.

Выполнил: *Чистяков К.С.*
Группа: *8О-406Б-21*
Преподаватель: А.Ю. Морозов

Москва, 2025

Условие

1. **Цель работы.** Использование объединения запросов к глобальной памяти. Реализация метода Гаусса с выбором главного элемента по столбцу. Ознакомление с библиотекой алгоритмов для параллельных расчетов Thrust. Использование двухмерной сетки потоков. Исследование производительности программы с помощью утилиты nvprof.
2. **Вариант задания.** Вариант 3. Решение квадратной СЛАУ.
   Необходимо решить систему уравнений Ax = b, где A — квадратная матрица n x n, b — вектор-столбец свободных коэффициентов длинной n, x — вектор неизвестных.
   Входные данные. На первой строке задано число n - размер матрицы. В следующих n строках, записано по n вещественных чисел — элементы матрицы. Далее записываются n элементов вектора свободных коэффициентов. $n \leq 10^4$.
   Выходные данные. Необходимо вывести n значений, являющиеся элементами вектора неизвестных x.

Программное и аппаратное обеспечение

Графический процессор (Google Colab)

Compute capability	7.5
Name	Tesla T4
Total Global Memory	15828320256
Shared memory per block	49152
Registers per block	65536
Warp size	32
Max threads per block	(1024, 1024, 64)
Max block	(2147483647, 65535, 65535)
Total constant memory	6553
Multiprocessors count	40

Процессор AMD Ryzen 5 4500U with Radeon Graphics (2.38 GHz)

Technology	7 nm
Cores	6

Threads	6
Core Speed	1390 MHz
Cache (L1 Data)	6 x 32 KBytes (8-way)
Cache (L1 Inst.)	6 x 32 KBytes (8-way)
Cache (Level 2)	6 x 512 KBytes (8-way)
Cache (Level 3)	2 x 4 MBytes (16-way)

Оперативная память

Number of modules	2
Module Manuf.	Samsung
Module Size	8 GBytes (total - 16 GBytes)
Generation	DDR4

Жесткий диск - 512 ГБ SSD
OS - Windows 10 Домашняя + Linux
IDE - VS Code + Google Colab
Compiler - nvcc, gcc

Метод решения

Сначала нам надо привести матрицу к верхнетреугольному виду, а затем с помощью обратного хода найти элементы вектора неизвестных x. Для этого мы в каждом столбце i выбираем главный элемент (наибольший по модулю) и если этот элемент не находится в строке i, то меняем строку с этим элементом со строкой i местами. Затем проходимся прямым ходом (метод Гаусса), чтобы занулить все нижние элементы под текущим столбцом. Делаем так пока не приведём матрицу к верхнетругольному виду. После этого пользуемся методом обратной подстановки и находим элементы вектора неизвестных x.

Описание программы

Вся программа описана в одном файле.

- CSC(call) — функция для отлова ошибок CUDA
- comparator — компаратор для определения максимального элемента по модулю
- int findMaxRow(double* d_A_ptr, int n, int i) — функция для нахождения индекса строки с максимальным по модулю элементом в столбце i.
- __global__ void swapA(double* A, int n, int i, int maxRow) — функция для перестановки строк для матрицы A

- __global__ void swapB(double* b, int n, int i, int maxRow) — функция для перестановки строк для вектора B
- __global__ void kernelA(double* A, int n, int i) — функция для обнуления поддиагональных элементов в i-м столбце (прямой ход) для матрицы A
- __global__ void kernelB(double* A, double* b, int n, int i) — функция для обнуления поддиагональных элементов в i-м столбце (прямой ход) для вектора b
- int main() — главная функция, где происходит считывание размера матрицы, ввод матрицы A и вектора b, выделение и копирование данных с CPU на GPU и наоборот, запуск всех функций, выполнение обратного хода, вывод решения и освобождение памяти.

Ядро для перемены строк местами для матрицы A:
```
__global__ void swapA(double* A, int n, int i, int maxRow) {
  int idx = blockIdx.x * blockDim.x + threadIdx.x;
  int offsetx = blockDim.x * gridDim.x;
  for (int j = idx; j < n; j += offsetx) {
    double temp = A[j * n + i];
    A[j * n + i] = A[j * n + maxRow];
    A[j * n + maxRow] = temp;
  }
}
```

Ядро для перемены строк местами для вектора B:
```
__global__ void swapB(double* b, int n, int i, int maxRow) {
  int idx = blockIdx.x * blockDim.x + threadIdx.x;
  if (idx == 0) {
    double temp_b = b[i];
    b[i] = b[maxRow];
    b[maxRow] = temp_b;
  }
}
```

Ядро для выполнения прямого хода метода Гаусса для матрицы A:
```
__global__ void kernelA(double* A, int n, int i) {
  int idx = blockIdx.x * blockDim.x + threadIdx.x;
  int idy = blockIdx.y * blockDim.y + threadIdx.y;
  int offsetx = blockDim.x * gridDim.x;
  int offsety = blockDim.y * gridDim.y;

  for (int r = idx + i + 1; r < n; r += offsetx) {
    if (r > i) {
      double temp = A[i * n + r] / A[i * n + i];
      for (int c = idy + i + 1; c < n; c += offsety) {
        A[c * n + r] -= temp * A[c * n + i];
```

```
    }
   }
  }
}
```

Ядро для выполнения прямого хода метода Гаусса для вектора b:

```
__global__ void kernelB(double* A, double* b, int n, int i) {
  int idx = blockIdx.x * blockDim.x + threadIdx.x;
  int idy = blockIdx.y * blockDim.y + threadIdx.y;
  int offsetx = blockDim.x * gridDim.x;

  for (int r = idx + i + 1; r < n; r += offsetx) {
    if (r > i) {
      double temp = A[i * n + r] / A[i * n + i];
      if (idy == 0) b[r] -= temp * b[i];
    }
  }
}
```

Результаты

Замеры времени работы ядер с различными конфигурациями для kernel и фиксированной конфигурацией для swap <<<512, 512>>> . Тесты будут следующих размеров: n = 2, n = 100, n = 1000, n = 10000. А также будет приведён замер работы аналогичной программы без использования технологии CUDA - на CPU.

1. **Конфигурация <<< dim3(2, 2), dim3(4, 4) >>>**

n	Время (в мс)
2	1.195015
100	14.458192
1000	1810.556596
10000	долго

2. **Конфигурация <<< dim3(8, 8), dim3(16, 16) >>>**

n	Время (в мс)
2	1.213471
100	8.906913

1000	156.686411
10000	31174.859851

3. **Конфигурация <<< dim3(32, 32), dim3(32, 32) >>>**

n	Время (в мс)
2	1.584428
100	16.301694
1000	321.400121
10000	31156.213560

4. **CPU**

n	Время (в мс)
2	0
100	1,221
1000	1174,737
10000	долго

Использование nvprof

Для теста из примера:

```
==70540== Profiling result:
            Type  Time(%)      Time     Calls       Avg       Min       Max  Name
 GPU activities:   39.23%  65.535us         2  32.767us  25.920us  39.615us  kernelA(double*, int, int)
                   31.86%  53.214us         2  26.607us  25.855us  27.359us  kernelB(double*, double*, int, int)
                    5.44%  9.0880us         2  4.5440us  4.0640us  5.0240us  void thrust::THRUST_200400_750_NS::cuda_cub::core::_kernel_agent<thrust::THRUST_2004
                    4.85%  8.0960us         1  8.0960us  8.0960us  8.0960us  swapA(double*, int, int, int)
                    4.43%  7.3920us         4  1.8480us  1.6640us  2.1440us  [CUDA memcpy DtoH]
                    4.33%  7.2320us         1  7.2320us  7.2320us  7.2320us  swapB(double*, int, int, int)
                    3.35%  5.5990us         2  2.7990us  2.7840us  2.8150us  void cub::CUB_200400_750_NS::detail::for_each::static_kernel<cub::CUB_200400_750_NS:
                    3.14%  5.2480us         2  2.6240us  2.4320us  2.8160us  void cub::CUB_200400_750_NS::detail::for_each::static_kernel<cub::CUB_200400_750_NS:
                    2.28%  3.8080us         2  1.9040us  1.8880us  1.9200us  void cub::CUB_200400_750_NS::detail::for_each::static_kernel<cub::CUB_200400_750_NS:
                    1.09%  1.8240us         2     912ns     672ns  1.1520us  [CUDA memcpy HtoD]
      API calls:   97.85%  88.616ms         6  14.769ms  3.2470us  88.587ms  cudaMalloc
                    1.34%  1.2144ms         1  1.2144ms  1.2144ms  1.2144ms  cudaFuncGetAttributes
                    0.18%  158.54us         8  19.817us     976ns  43.134us  cudaDeviceSynchronize
                    0.16%  147.12us         6  24.520us  2.6880us  125.90us  cudaFree
                    0.15%  133.02us       114  1.1660us     108ns  54.474us  cuDeviceGetAttribute
                    0.13%  120.95us        14  8.6390us  3.8180us  43.643us  cudaLaunchKernel
                    0.06%  50.585us         4  12.646us  5.4930us  21.700us  cudaMemcpy
                    0.05%  43.109us        10  4.3100us  1.7310us  6.6920us  cudaStreamSynchronize
                    0.04%  35.731us         2  17.865us  13.271us  22.460us  cudaMemcpyAsync
                    0.01%  11.672us         1  11.672us  11.672us  11.672us  cuDeviceGetName
                    0.01%  10.039us       100     100ns      81ns     781ns  cudaGetLastError
                    0.01%  5.8920us        17     346ns     200ns  1.0810us  cudaGetDevice
                    0.01%  5.2050us         1  5.2050us  5.2050us  5.2050us  cuDeviceGetPCIBusId
                    0.00%  2.6460us         6     441ns     196ns  1.4130us  cudaDeviceGetAttribute
                    0.00%  1.7860us         3     595ns     131ns  1.3700us  cuDeviceGetCount
                    0.00%  1.7710us         2     885ns     193ns  1.5780us  cuDeviceGet
                    0.00%  1.0920us        10     109ns      85ns     183ns  cudaPeekAtLastError
                    0.00%     515ns         1     515ns     515ns     515ns  cuModuleGetLoadingMode
                    0.00%     455ns         1     455ns     455ns     455ns  cuDeviceTotalMem
                    0.00%     291ns         1     291ns     291ns     291ns  cudaGetDeviceCount
                    0.00%     281ns         1     281ns     281ns     281ns  cuDeviceGetUuid
```

Использование ncu

Для n = 20:

Для ядра выполняющего прямой ход для матрицы A:

```
kernelA(double *, int, int) (32, 32, 1)x(32, 32, 1), Device 0, CC 7.5, Invocations 20
  Section: Command line profiler metrics
  ------------------------------------------------------- ----------- ------- -------- --------
  Metric Name                                             Metric Unit Minimum  Maximum  Average
  ------------------------------------------------------- ----------- ------- -------- --------
  l1tex__data_bank_conflicts_pipe_lsu_mem_shared_op_ld.avg                0.00     0.00     0.00
  l1tex__data_bank_conflicts_pipe_lsu_mem_shared_op_ld.max                0.00     0.00     0.00
  l1tex__data_bank_conflicts_pipe_lsu_mem_shared_op_ld.min                0.00     0.00     0.00
  l1tex__data_bank_conflicts_pipe_lsu_mem_shared_op_ld.sum                0.00     0.00     0.00
  l1tex__data_bank_conflicts_pipe_lsu_mem_shared_op_st.avg                0.00     0.00     0.00
  l1tex__data_bank_conflicts_pipe_lsu_mem_shared_op_st.max                0.00     0.00     0.00
  l1tex__data_bank_conflicts_pipe_lsu_mem_shared_op_st.min                0.00     0.00     0.00
  l1tex__data_bank_conflicts_pipe_lsu_mem_shared_op_st.sum                0.00     0.00     0.00
  l1tex__t_sectors_pipe_lsu_mem_global_op_ld.avg               sector     0.00   155.45    95.60
  l1tex__t_sectors_pipe_lsu_mem_global_op_ld.max               sector     0.00   486.00   276.15
  l1tex__t_sectors_pipe_lsu_mem_global_op_ld.min               sector     0.00     0.00     0.00
  l1tex__t_sectors_pipe_lsu_mem_global_op_ld.sum               sector     0.00 6,218.00 3,824.20
  l1tex__t_sectors_pipe_lsu_mem_global_op_st.avg               sector     0.00     2.38     0.86
  l1tex__t_sectors_pipe_lsu_mem_global_op_st.max               sector     0.00    95.00    34.50
  l1tex__t_sectors_pipe_lsu_mem_global_op_st.min               sector     0.00     0.00     0.00
  l1tex__t_sectors_pipe_lsu_mem_global_op_st.sum               sector     0.00    95.00    34.50
  sm__sass_inst_executed_op_local.avg                            inst     0.00     0.00     0.00
  sm__sass_inst_executed_op_local.max                            inst     0.00     0.00     0.00
  sm__sass_inst_executed_op_local.min                            inst     0.00     0.00     0.00
  sm__sass_inst_executed_op_local.sum                            inst     0.00     0.00     0.00
  smsp__branch_targets_threads_divergent                     branches     0.00 1,024.00    51.20
  ------------------------------------------------------- ----------- ------- -------- --------
```

Для ядра выполняющего прямой ход для вектора B:

```
kernelB(double *, double *, int, int) (32, 32, 1)x(32, 32, 1), Device 0, CC 7.5, Invocations 20
  Section: Command line profiler metrics
  ------------------------------------------------------- ----------- ------- ------- -------
  Metric Name                                             Metric Unit Minimum Maximum Average
  ------------------------------------------------------- ----------- ------- ------- -------
  l1tex__data_bank_conflicts_pipe_lsu_mem_shared_op_ld.avg                0.00    0.00    0.00
  l1tex__data_bank_conflicts_pipe_lsu_mem_shared_op_ld.max                0.00    0.00    0.00
  l1tex__data_bank_conflicts_pipe_lsu_mem_shared_op_ld.min                0.00    0.00    0.00
  l1tex__data_bank_conflicts_pipe_lsu_mem_shared_op_ld.sum                0.00    0.00    0.00
  l1tex__data_bank_conflicts_pipe_lsu_mem_shared_op_st.avg                0.00    0.00    0.00
  l1tex__data_bank_conflicts_pipe_lsu_mem_shared_op_st.max                0.00    0.00    0.00
  l1tex__data_bank_conflicts_pipe_lsu_mem_shared_op_st.min                0.00    0.00    0.00
  l1tex__data_bank_conflicts_pipe_lsu_mem_shared_op_st.sum                0.00    0.00    0.00
  l1tex__t_sectors_pipe_lsu_mem_global_op_ld.avg               sector     0.00    0.25    0.17
  l1tex__t_sectors_pipe_lsu_mem_global_op_ld.max               sector     0.00   10.00    6.75
  l1tex__t_sectors_pipe_lsu_mem_global_op_ld.min               sector     0.00    0.00    0.00
  l1tex__t_sectors_pipe_lsu_mem_global_op_ld.sum               sector     0.00   10.00    6.75
  l1tex__t_sectors_pipe_lsu_mem_global_op_st.avg               sector     0.00    0.12    0.07
  l1tex__t_sectors_pipe_lsu_mem_global_op_st.max               sector     0.00    5.00    2.75
  l1tex__t_sectors_pipe_lsu_mem_global_op_st.min               sector     0.00    0.00    0.00
  l1tex__t_sectors_pipe_lsu_mem_global_op_st.sum               sector     0.00    5.00    2.75
  sm__sass_inst_executed_op_local.avg                            inst     0.00    0.00    0.00
  sm__sass_inst_executed_op_local.max                            inst     0.00    0.00    0.00
  sm__sass_inst_executed_op_local.min                            inst     0.00    0.00    0.00
  sm__sass_inst_executed_op_local.sum                            inst     0.00    0.00    0.00
  smsp__branch_targets_threads_divergent                     branches     0.00    1.00    0.05
  ------------------------------------------------------- ----------- ------- ------- -------
```

Для ядра выполняющего перемену строк для матрицы A:

```
swapA(double *, int, int, int) (512, 1, 1)x(512, 1, 1), Device 0, CC 7.5, Invocations 19
  Section: Command line profiler metrics
  ------------------------------------------------------- ----------- ------- ------- -------
  Metric Name                                             Metric Unit Minimum Maximum Average
  ------------------------------------------------------- ----------- ------- ------- -------
  l1tex__data_bank_conflicts_pipe_lsu_mem_shared_op_ld.avg                0.00    0.00    0.00
  l1tex__data_bank_conflicts_pipe_lsu_mem_shared_op_ld.max                0.00    0.00    0.00
  l1tex__data_bank_conflicts_pipe_lsu_mem_shared_op_ld.min                0.00    0.00    0.00
  l1tex__data_bank_conflicts_pipe_lsu_mem_shared_op_ld.sum                0.00    0.00    0.00
  l1tex__data_bank_conflicts_pipe_lsu_mem_shared_op_st.avg                0.00    0.00    0.00
  l1tex__data_bank_conflicts_pipe_lsu_mem_shared_op_st.max                0.00    0.00    0.00
  l1tex__data_bank_conflicts_pipe_lsu_mem_shared_op_st.min                0.00    0.00    0.00
  l1tex__data_bank_conflicts_pipe_lsu_mem_shared_op_st.sum                0.00    0.00    0.00
  l1tex__t_sectors_pipe_lsu_mem_global_op_ld.avg                sector    0.88    1.00    0.93
  l1tex__t_sectors_pipe_lsu_mem_global_op_ld.max                sector   35.00   40.00   37.21
  l1tex__t_sectors_pipe_lsu_mem_global_op_ld.min                sector    0.00    0.00    0.00
  l1tex__t_sectors_pipe_lsu_mem_global_op_ld.sum                sector   35.00   40.00   37.21
  l1tex__t_sectors_pipe_lsu_mem_global_op_st.avg                sector    1.00    1.00    1.00
  l1tex__t_sectors_pipe_lsu_mem_global_op_st.max                sector   40.00   40.00   40.00
  l1tex__t_sectors_pipe_lsu_mem_global_op_st.min                sector    0.00    0.00    0.00
  l1tex__t_sectors_pipe_lsu_mem_global_op_st.sum                sector   40.00   40.00   40.00
  sm__sass_inst_executed_op_local.avg                             inst    0.00    0.00    0.00
  sm__sass_inst_executed_op_local.max                             inst    0.00    0.00    0.00
  sm__sass_inst_executed_op_local.min                             inst    0.00    0.00    0.00
  sm__sass_inst_executed_op_local.sum                             inst    0.00    0.00    0.00
  smsp__branch_targets_threads_divergent                    branches    0.00    0.00    0.00
  ------------------------------------------------------- ----------- ------- ------- -------
```

Для ядра выполняющего перемену строк для вектора B:

```
swapB(double *, int, int, int) (512, 1, 1)x(512, 1, 1), Device 0, CC 7.5, Invocations 19
  Section: Command line profiler metrics
  ------------------------------------------------------- ----------- ------- ------- -------
  Metric Name                                             Metric Unit Minimum Maximum Average
  ------------------------------------------------------- ----------- ------- ------- -------
  l1tex__data_bank_conflicts_pipe_lsu_mem_shared_op_ld.avg                0.00    0.00    0.00
  l1tex__data_bank_conflicts_pipe_lsu_mem_shared_op_ld.max                0.00    0.00    0.00
  l1tex__data_bank_conflicts_pipe_lsu_mem_shared_op_ld.min                0.00    0.00    0.00
  l1tex__data_bank_conflicts_pipe_lsu_mem_shared_op_ld.sum                0.00    0.00    0.00
  l1tex__data_bank_conflicts_pipe_lsu_mem_shared_op_st.avg                0.00    0.00    0.00
  l1tex__data_bank_conflicts_pipe_lsu_mem_shared_op_st.max                0.00    0.00    0.00
  l1tex__data_bank_conflicts_pipe_lsu_mem_shared_op_st.min                0.00    0.00    0.00
  l1tex__data_bank_conflicts_pipe_lsu_mem_shared_op_st.sum                0.00    0.00    0.00
  l1tex__t_sectors_pipe_lsu_mem_global_op_ld.avg                sector    0.05    0.05    0.05
  l1tex__t_sectors_pipe_lsu_mem_global_op_ld.max                sector    2.00    2.00    2.00
  l1tex__t_sectors_pipe_lsu_mem_global_op_ld.min                sector    0.00    0.00    0.00
  l1tex__t_sectors_pipe_lsu_mem_global_op_ld.sum                sector    2.00    2.00    2.00
  l1tex__t_sectors_pipe_lsu_mem_global_op_st.avg                sector    0.05    0.05    0.05
  l1tex__t_sectors_pipe_lsu_mem_global_op_st.max                sector    2.00    2.00    2.00
  l1tex__t_sectors_pipe_lsu_mem_global_op_st.min                sector    0.00    0.00    0.00
  l1tex__t_sectors_pipe_lsu_mem_global_op_st.sum                sector    2.00    2.00    2.00
  sm__sass_inst_executed_op_local.avg                             inst    0.00    0.00    0.00
  sm__sass_inst_executed_op_local.max                             inst    0.00    0.00    0.00
  sm__sass_inst_executed_op_local.min                             inst    0.00    0.00    0.00
  sm__sass_inst_executed_op_local.sum                             inst    0.00    0.00    0.00
  smsp__branch_targets_threads_divergent                    branches    0.00    0.00    0.00
  ------------------------------------------------------- ----------- ------- ------- -------
```

Выводы

В ходе выполнения лабораторной работы я научился использовать объединение запросов к глобальной памяти. Реализовал метода Гаусса с выбором главного элемента по столбцу. Ознакомился с библиотекой алгоритмов для параллельных расчетов Thrust.

Использовал двухмерную сетку потоков. Исследовал производительность программы с помощью утилиты nvprof.

Этот алгоритм может применяться в области численного моделирования и вычислительной физики (гидродинамика, термодинамика, электродинамика), в области компьютерной графики и обработки изображений (глобальное освещение и трассировка лучей, фильтрация изображений, калибровка камер и реконструкция 3D-сцен, в области машинного обучения и анализа данных (нейронные сети) и т.д.

В процессе написания программы основными сложностями стали: понять принцип связи двумерной сетки потоков и элементов матрицы, а также понять, как использовать Thrust для нахождения максимального элемента в столбце.

Использование nvprof и ncu помогло проанализировать программу и оптимизировать более-менее код. Как видно из приведённых выше снимков экрана дивергенция есть, но она незначительная.

Сравнивая время работы ядер с различными конфигурациями и размерами матриц, а также аналогичной программы без использования технологии CUDA - на CPU, можно заметить, что программа на CPU выигрывает в относительно небольших тестах, но сильно проигрывает GPU с конфигурациями больше самой маленькой (маленькая выдаёт примерно те же цифры на тесте n = 1000). То есть, в целом можно сказать, что конфигурации с достаточным количеством потоков всегда выигрывают по скорости у CPU, причём значительно.