МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ

«НОВОСИБИРСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ»

Кафедра Вычислительной техники

Отчет

по лабораторной работе №2

по дисциплине «Компьютерная графика»

**«Расчет определителя матрицы методом Гаусса на GPU»**

Факультет: АВТФ

Группа: АВТ-042

Студент: Преподаватель:

Антонов А.Р. Балакин В.В.

Новосибирск 2022

**Цель**

Найти зависимость времени расчета от порядка матрицы при расчете на GPU. Сравнить зависимость расчета на GPU и CPU.

**Задание**

Написать программу для расчета определителя матрицы методом Гаусса на языке программирования C с использованием CUDA. Замерить время расчета для определителя порядков от 100 до 3480 с шагом 20. Построить график зависимости времени от порядка. Сравнить с графиком полученным без использования CUDA.

**Теоретическое описание**

Чтобы вычислить определитель по методу Гаусса, начальная матрица приводится к верхней треугольной форме с помощью элементарных преобразований. После этого определитель находится как произведение элементов главной диагонали.

**Структура программы.**

* Функция которая считает определитель методом Гаусса.
* \_\_global\_\_ void gaussDeterminant(double\* data, double\* dev\_data,double\*dev\_data1, int matrixOrder) {  
   double determinant = 0;  
   unsigned int idx = blockIdx.x; //get index of block  
   //matrix to triangle  
   for (int i = 0; i < matrixOrder- 1; ++i)  
   kernel\_down (dev\_data, dev\_data1, matrixOrder, i, i);  
   //calculate determinant  
   determinant = data[0];  
   for (int i = 1; i < matrixOrder; i++) {  
   determinant \*= data[i \* idx \* matrixOrder + i];  
   };  
    
  }

Функция изменяющая строки для привода матрицы к треугольному виду

* \_\_device\_\_ void kernel\_down(double\* A, double\* B, int n,  
   int number\_row, int number\_column) {  
   int bx = blockIdx.x;  
   int by = blockIdx.y;  
   int tx = threadIdx.x;  
   int ty = threadIdx.y;  
   int row = by \* n + ty;  
   int column = bx \* n + tx;  
    
   if (number\_column < column && number\_row < row) {  
   double glav = A[row \* n + number\_column] / A[number\_row \* n + number\_column];  
    
   if (number\_column == column) B[row] -= B[number\_row] \* glav;  
   A[row \* n + column] -= glav \* A[number\_row \* n + column];  
   }  
  }
* Функция main в которой проходит инициализация CUDA элементов и цикл по расчету матриц порядка от 100 до 3480
* int main() {  
   FILE\* f1, \* f2;  
     
     
   //main cycle  
   for (int n = 100; n < 3500; n += 20) {  
   //matrix creation  
   double\* data = (double\*)malloc(sizeof(double) \* (n \* n));  
   double\* data1 = (double\*)malloc(n \* sizeof(double));  
    
   srand(time(0));  
   for (int i = 0; i < n \* n; i++) {  
   data[i] = (double)(rand() % 101);  
   }  
    
   double determinant = 1;  
   //write matrix order to file  
   f1 = fopen("Matrix Orders.txt", "a");  
   printf("matrix order = %d\n", n);  
   fprintf(f1, "%d\n", n);  
   fclose(f1);  
    
   double\* dev\_data;  
   double\* dev\_data1;  
   //init events to get time of calculation  
   cudaEvent\_t begin, end;  
   cudaEventCreate(&begin);  
   cudaEventCreate(&end);  
   cudaEventRecord(begin, 0);  
    
   //allocate memory on device  
   cudaMalloc((void\*\*)&dev\_data, sizeof(double) \* (n \* n));  
   cudaMalloc((void\*\*)&dev\_data1, sizeof(double) \* (n));  
   cudaMemcpy(dev\_data, data, sizeof(double) \* n \* n, cudaMemcpyHostToDevice);  
   cudaMemcpy(dev\_data1, data1, sizeof(double) \* n , cudaMemcpyHostToDevice);  
   float timeToCalculate = 0.0;  
     
    
   gaussDeterminant <<<n, 1 >>> (data, dev\_data,dev\_data1, n);  
   cudaMemcpy(data, dev\_data, n \* n \* sizeof(double), cudaMemcpyDeviceToHost);  
   cudaEventRecord(end, 0);  
   cudaEventSynchronize(end);  
   cudaEventElapsedTime(&timeToCalculate, begin, end);  
    
   //write time to calculate in file  
   f2 = fopen("Time in milli seconds GPU.txt", "a");  
   fprintf(f2, "%.2f\n", timeToCalculate);  
   fclose(f2);  
    
   printf("time to calculate = %.2f milliseconds\n", timeToCalculate);  
    
   //printf("determinant %f\n", determinant);  
    
   //destroy events  
   cudaEventDestroy(begin);  
   cudaEventDestroy(end);  
   //free memory  
   cudaFree(dev\_data);  
   free(data);  
   }  
   return 0;  
  }

**Результаты**

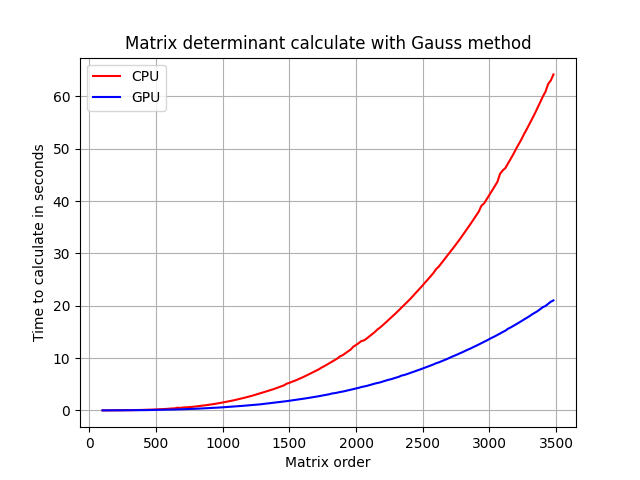
****

Рис 1. Графики зависимости времени расчета от порядка матрицы

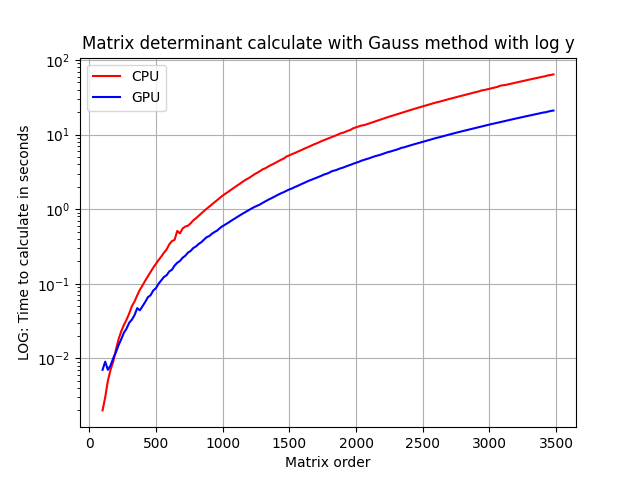


Рис 2. Графики зависимости времени расчета от порядка матрицы с логарифмической осью y

**Вывод**

Написали программу для расчета определителя матрицы методом Гаусса на CUDA. Нашли зависимость времени расчета определителя матрицы методом гаусса от порядка матрицы с использованием CUDA. Сравнили графики CPU и GPU. Из графиков видно, что GPU перегоняет CPU начиная с определителя 220.