Министерство науки и высшего образования Российской Федерации
Федеральное государственное бюджетное образовательное учреждение высшего образования
«Московский государственный технический университет имени Н.Э. Баумана
(национальный исследовательский университет)»
(МГТУ им. Н.Э. Баумана)

ФАКУЛЬТЕТ *Робототехники и комплексной автоматизации*

КАФЕДРА *Системы автоматизированного проектирования (РК-6)*

ОТЧЕТ О ВЫПОЛНЕНИИ ЛАБОРАТОРНОЙ РАБОТЫ

по дисциплине: «Разработка программных систем»

Студент	Израелян Ева Арамовна
Группа	РК6-64Б
Тип задания	Лабораторная работа №2
Тема лабораторной работы	Многопоточное программирование

Студент _____ **Израелян Е.А.**
подпись, дата *фамилия, и.о.*

Преподаватель _____ **Федорук В.Г.**
подпись, дата *фамилия, и.о.*

Оценка _____

Москва, 2022 г.

Оглавление

Задание на лабораторную работу

Разработать многопотоковый вариант программы моделирования распространения электрических сигналов в двухмерной прямоугольной сетке RC-элементов. Метод формирования математической модели - узловой. Метод численного интегрирования - **явный** Эйлера. Внешнее воздействие - источники тока и напряжения. Количество потоков, временной интервал моделирования и количество элементов в сетке - параметры программы. Программа должна демонстрировать ускорение по сравнению с последовательным вариантом. Предусмотреть визуализацию результатов посредством утилиты gnuplot. При этом утилита gnuplot должна вызываться отдельной командой после окончания расчёта.

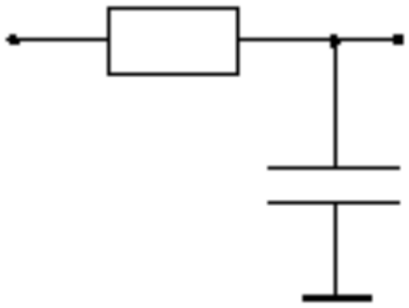

Рисунок 1. RC-элемент

Описание структуры программы и реализованных способов взаимодействия потоков управления

Входные параметры: количество строк, количество столбцов, количество потоков, количество итераций.

В начале программы задаются все глобальные константы, необходимые для расчётов, объявляются переменные двух барьеров и структура, обеспечивающая контроль выхода за границы массива ленты со значениями напряжений в узлах ThreadRecord.

В начале функции main() производится проверка на количество поступивших аргументов, затем производится присвоение объявленным переменным соответствующих аргументов. Происходит выделение памяти под «текущий» и

«предыдущий» массив со значениями напряжений, заполнение их нулевыми значениями. Затем выполняется инициализация атрибутов потока, указание области конкуренции потоков, состояния отсоединённости. Выделяется память под структуры ThreadRecord размером nthread (количество потоков), инициализируются границы лент матрицы сетки RC-элементов в каждой из соответствующих структур. С помощью функций pthread_barrier_init() создаётся два барьера для синхронизации работы потоков управления с шириной барьера равной nthread+1. Функцией pthread_create() создаются потоки управления в количестве nthread для функции my_solver(). Узлы сетки делятся между потоками равномерно.

После создания потоков управления они приостанавливаются барьером1 до тех пор, пока функция main() не дойдёт до этапа «Старт расчётов» (см. комментарии в коде программы). После того как барьер1 будет пройден, выполнится расчёт значений напряжений в узлах сетки первого временного интервала каждым из потоков и каждый из них будет приостановлен барьером2 до того момента, пока все потоки управления не досчитают свои узлы. Как только барьер2 будет пройден, функция main() выполнит копирование значений «текущего» массива значений напряжений в узлах сетки в «предыдущий», после чего остановится на барьере1. Когда он будет пройден, потоки возобновят расчёты, относящиеся к следующему временному интервалу.

Описанные взаимодействия расчётов и копирования массивов значений напряжений в узлах будут продолжены, пока не будут рассчитаны значения напряжений в сетке для каждого интервала времени.

Описание основных используемых структур данных

– double *cur, *prev – массивы, хранящие значения напряжений в узлах сетки текущий и предыдущий момент времени;

– pthread_barrier_t barr1, barr2 – для синхронизации работы потоков;

– структура ThreadRecord:

```
typedef struct {
      pthread_t tid; // Идентификатор потока управления
      int first;
      int last;
} ThreadRecord;
```

Блок-схема программы

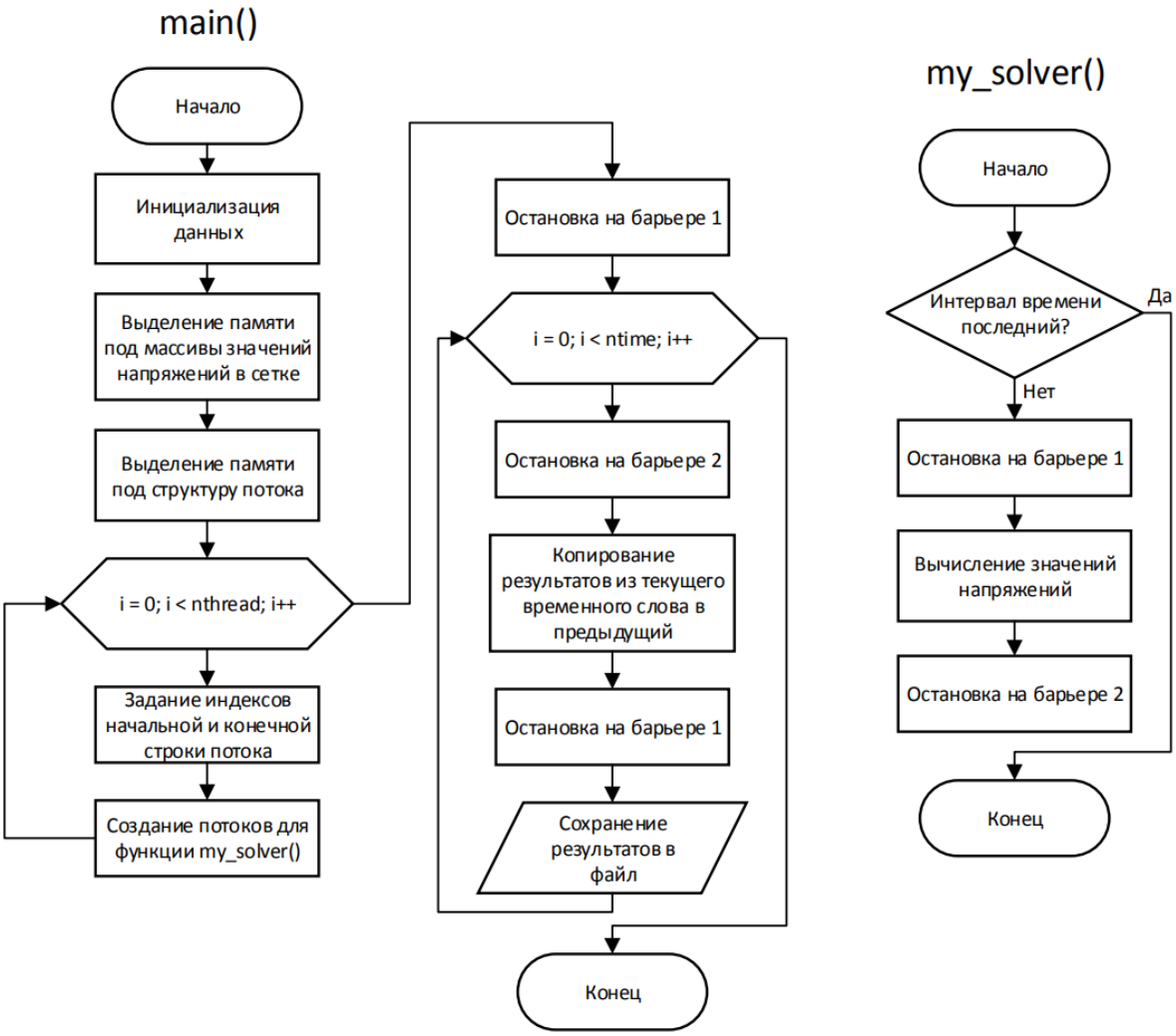

Рисунок 2. Блок-схема к программе

Примеры результатов работы программы

Запуск программы: ./a.out 8 8 1 100

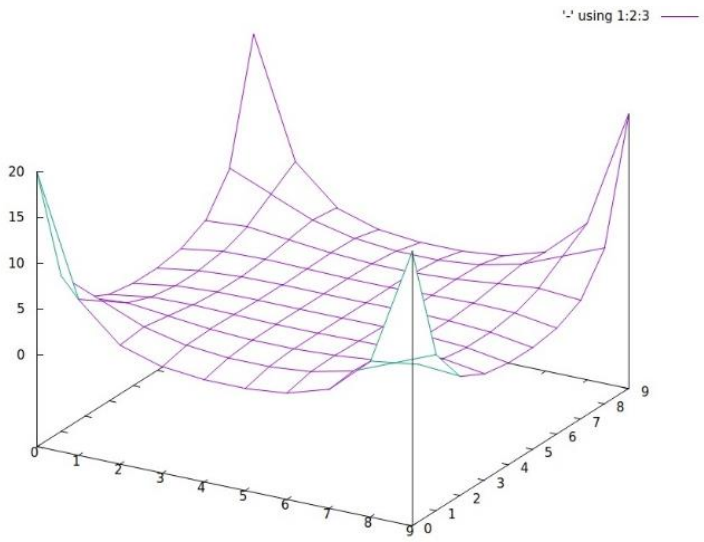

Рисунок 3. Визуализация результатов посредством утилиты gnuplot

Время выполнения программы в неграфическом режиме для размерности сетки 2048x2048 для 2048 временных слоёв:

Количество потоков	Время, ms
1	233680
2	117769
4	64264
8	56192

Текст программы

```
#include <stdlib.h>
#include <unistd.h>
#include <stdio.h>
#include <string.h>
#include <signal.h>
#include <pthread.h>
#include <sched.h>
#include <sys/time.h>

typedef struct {
	pthread_t tid;
	int first;
	int last;
} ThreadRecord;

int M, N; // Размерность сетки
double* cur, * prev; // Массивы значений напряжений в текущий и предыдущий момент времени
double U = 20.0;
double C = 1.0;
double R = 5.0;
double h = 1.0;
double I = 10.0;
int done = 0;
ThreadRecord* threads; // Массив индексов границ лент матрицы
pthread_barrier_t barr1, barr2;
```

```
void* my_solver(void* arg_p) {
	ThreadRecord* thr;
	int i, j;
	double tmp = 0.0;
	thr = (ThreadRecord*)arg_p;
	while (!done) {
		pthread_barrier_wait(&barr1);
		for (i = thr->first; i <= thr->last; i++) {
			for (j = 0; j < N; j++) {
				// Условия вычислений для четырех концов сетки RC-элементов
				if (j == 0 && i == 0) {
					cur[i * N + j] = U;
					continue;
				}
				else if (j == N - 1 && i == 0) {
					cur[i * N + j] = U;
					continue;
				}
				else if (j == 0 && i == M - 1) {
					cur[i * N + j] = U;
					continue;
				}
				else if (j == N - 1 && i == M - 1) {
					cur[i * N + j] = U;
					continue;
				} // Условия для вычисления остальных границ сетки RC-элементов
				else if (j == 0)
					tmp = prev[(i - 1) * N + j] + prev[i * N + j + 1] +
					prev[(i + 1) * N + j];
				else if (j == N - 1)
					tmp = prev[i * N + j - 1] + prev[(i - 1) * N + j] +
					prev[(i + 1) * N + j];
				else if (i == 0)
					tmp = prev[i * N + j - 1] + prev[i * N + j + 1] +
					prev[(i + 1) * N + j];
				else if (i == M - 1)
					tmp = prev[i * N + j - 1] + prev[(i - 1) * N + j] +
					prev[i * N + j + 1];
				// Условие для части расчета внутренних узлов сетки RC-элементов
				else
					tmp = prev[i * N + j - 1] + prev[(i - 1) * N + j] +
					prev[i * N + j + 1] + prev[(i + 1) * N + j];
				// Окончательный подсчет значения напряжения в узле
				cur[i * N + j] = h * tmp / C / R + prev[i * N + j] * (1 - 4 *
					h / C / R);
			}
		}
		pthread_barrier_wait(&barr2);
	}
}

int main(int argc, char* argv[]) {
	int nthread; // Кол-во расчётных потоков
	int ntime;   // Кол-во циклов
	int i, j, k;

	pthread_attr_t pattr;

	if (argc != 5)
		exit(1);

	M = atoi(argv[1]);
	N = atoi(argv[2]);
	nthread = atoi(argv[3]);
	ntime = atoi(argv[4]);
```

```
	if (M % nthread)
		exit(2);

	// Выделение памяти
	cur = (double*)malloc(sizeof(double) * M * N);
	prev = (double*)malloc(sizeof(double) * M * N);

	memset(cur, 0, sizeof(double) * M * N);
	memset(prev, 0, sizeof(double) * M * N);

	pthread_attr_init(&pattr);
	pthread_attr_setscope(&pattr, PTHREAD_SCOPE_SYSTEM);
	pthread_attr_setdetachstate(&pattr, PTHREAD_CREATE_JOINABLE);

	threads = (ThreadRecord*)calloc(nthread, sizeof(ThreadRecord));

	pthread_barrier_init(&barr1, NULL, nthread + 1);
	pthread_barrier_init(&barr2, NULL, nthread + 1);

	j = M / nthread;
	for (i = 0; i < nthread; i++) {
		threads[i].first = j * i; // Индекс начальной строки для потока
		threads[i].last = j * (i + 1) - 1; // Индекс конечной строки для потока
		if (pthread_create(&(threads[i].tid), &pattr, my_solver, (void*)
			&(threads[i]))) // Создание потоков управления
			perror("pthread_create");
	}
	struct timeval tv1, tv2, dtv;
	struct timezone tz;
	gettimeofday(&tv1, &tz);

	pthread_barrier_wait(&barr1); // Старт расчётов

	FILE* fd = fopen("output.txt", "w");
	FILE* fp = fopen("vg.dat", "w");
	fprintf(fp, "set dgrid3d\n");
	fprintf(fp, "set hidden3d\n");
	fprintf(fp, "set xrange [0:%d]\n", M);
	fprintf(fp, "set yrange [0:%d]\n", N);
	fprintf(fp, "set zrange [0:%d]\n", (int)U);
	fprintf(fp, "do for [i=0:%d]{\n", ntime - 1);
	fprintf(fp, "splot 'output.txt' index i using 1:2:3 with lines\npause(0.1)}\n");

	for (k = 1; k < ntime; k++) {
		pthread_barrier_wait(&barr2);
		memcpy(prev, cur, sizeof(double) * M * N);
		memset(cur, 0, M * N);
		pthread_barrier_wait(&barr1);
		for (i = 0; i < M; i++)
			for (j = 0; j < N; j++) {
				fprintf(fd, "%d\t%d\t%f\n", i, j, cur[i * N + j]);
			}
		fprintf(fd, "\n\n");
	}
	fclose(fd);
	pclose(fp);

	done = 1;
	gettimeofday(&tv2, &tz);
	dtv.tv_sec = tv2.tv_sec - tv1.tv_sec;
	dtv.tv_usec = tv2.tv_usec - tv1.tv_usec;
	printf("%d s %ld ms\n", (int)dtv.tv_sec, dtv.tv_usec);
	return 0;
}
```