МОСКОВСКИЙ ГОСУДАРАСТВЕННЫЙ УНИВЕРСИТЕТ

имени М. В. Ломоносова

Факультет Вычислительной Математики и Кибернетики

Компьютерный практикум по курсу

«ВВЕДЕНИЕ В ЧИСЛЕННЫЕ МЕТОДЫ»

ЗАДАНИЕ № 1

ОТЧЕТ

о выполнение задания

студента 205 учебной группы факультета ВМК МГУ

Швецова Дениса Андреевича

Москва. 2014 г.

Практическая работа №1.1.

Вариант

Приложение 1-9, приоложение 2(п.2-2);

Цель работы

Изучить классический метод Гаусса решения системы линейных алгебраических уравнений.

Постановка задачи

Дана система уравнений $Ax=f$ порядка nxn с невырожденной матрицей A. Написать программу, решающую систему линейных алгебраических уравнений заданного пользователем размера (n — параметр программы) методом Гаусса и методом Гаусса с выбором главного элемента.

Предусмотреть возможность задания элементов матрицы системы и ее правой части как во входном файле, так и путем задания специальных формул.

Цели и задачи практической работы.

1) Решить заданную СЛАУ методом Гаусса и методом Гаусса с выбором главного элемента;

2) Вычислить определитель матрицы *det(A);*

3) Вычислеть обратную матрицу A^{-1};

4) Определить вопрос вычислительной устойчивости метода Гаусса (при больших значениях праматреа n);

5) Правильность решения СЛАУ подтвердить системой тестов;

Алгоритм решения

Метод Гаусса.

Не ограничивая, общности будем считать, что коэффициент a_{11}, который называется ведущим элементом первого шага, отличен от нуля (в случае $a_{11} = 0$ поменяем местами строки в матрице A с номерами 1 и i, где a_{i1} не равен нулю). Разделим все члены первого уравнения на a_{11}. Вычтем из каждого i - го уравнения системы ($i = 2,\ldots, n$), преобразованное первое уравнение умноженное на a_{i1}. Все коэффициенты кроме первого в первом столбце матрицы A обнулятся. Далее рассмотрим укороченную систему из ($n - 1$) последних уравнений. И проделаем ту же операцию с ведущим элементом a_{22}. Продолжая такой процесс еще (n - 2) раза, приводим систему к виду $Cx = f$, *где* C – верхняя треугольная матрица. Это был прямой ход Гаусса. Обратный ход Гаусса состоит из последовательно определения, начиная с x_n, всех элементов вектора x.

Метод Гаусса с выбором главного элемента.

Полностью повторяет метод Гаусса, за исключением того, что ведущий элемент всегда наибольший элемент этой строки матрицы A, если это не так, то меняем столбцы местами.

Описание прграммы

На вход в командой строке программы подается два аргумента:

- число t (1 или 2).

- второй аргумент это имя файла, в котором находится матрица, если t = 1, или число из интервала (0,1] (используется для задания матрицы формулой), если t = 2.

Матрица в файле имеет следующий вид. Сначала записано число n размер матрицы. Затем записаны элементы первой строки матрицы коэффициентов через пробел, потом первый элемент столбца значений. Далее записана вторая строка матрицы коэффициентов и второй элемент столбца значений, затем третья строка… и т. д.

Программа выводит ответ в следующем виде:

«Решение СЛАУ методом Гаусса без выбора главного элемента:

X1 = …; X2 = …; X3 = …; … Xn = …;

Решение СЛАУ методом Гаусса с выбором главного элемента:
X1 = …; X2 = …; X3 = …; …Xn = ;

Определитель матрицы:
det(A) = …

Обратная матрица:
…
…
…»

Файл «Gaussian_elimination.c» содержит код программы.

Вычислительная устойчивость метода подтверждается системой тестов, которые были проверенны с помощью специализированного программного обеспечения (http://www.wolframalpha.com).

Выводы

Мной были изучен классический метод Гаусса решения системы линейных алгебраических уравнений.

Приложение 1

Текст программы

Файл «Gaussian_elimination.c»

```
#include <stdio.h>
#include <stdlib.h>
#include <math.h>
#include <string.h>

#define EPSILON 0.0000001
#define qM 1.001 - 2 * M * 0.001

typedef double **matrix;
typedef double * vector;

enum {
        M = 2,
        N = 40
};

enum {
        EAS = 0, //Gaussian elimination without pivoting
        PVT = 1  //Gaussuan elimination with pivoting
};

typedef int (*find_func)(matrix, int, int);
typedef void(*stdtion_func)(matrix, int, int);
typedef void(*subtract_func)(matrix, int, int, int);
typedef void(*calculate_func)(matrix, int, int *, vector, int);

matrix read_matrix(int *n, FILE *f)
{
        int i, j;
        fscanf(f,"%d\n", n);
        matrix a = calloc (*n, sizeof(vector));
        for (i = 0; i < *n; i++)
        {
                a[i] = calloc (*n + 1, sizeof(double));
                for (j = 0;  j < *n + 1; j++)
                        fscanf(f,"%lf", &a[i][j]);
        }
        return a;
}

matrix conversion_matrix(matrix src, int n)
{
        /*conversion matrix from a[string][column] to a[column][string]*/
        int i, j;
        matrix mtrx = calloc(n + 1, sizeof(vector));
        for (i = 0; i < n + 1; i++)
                mtrx[i] = calloc(n, sizeof(double));
        for (i = 0; i < n; i++)
                for (j = 0; j < n + 1; j++)
```

```
                        mtrx[j][i] = src[i][j];
        return mtrx;
}

matrix creat_matrix(int *n, double x)
{
        /*create matrix from of the formula*/
        int i, j;
        *n = N;
        matrix mtrx = calloc (N, sizeof(vector));
        for (i = 0; i < N; i++)
                mtrx[i] = calloc (N + 1, sizeof(double));
        for (i = 0; i < N; i++)
                for (j = 0; j < N; j++)
                        mtrx[i][j] = i == j ? pow(qM - 1, i + j) : pow(qM, i + j) + 0.1 * (j - i);
        for (i = 0; i < N; i++)
                mtrx[i][N] = fabs(x - (double)N / 10.0) * (i + 1) * sin(x);
        return mtrx;
}

matrix copy_matrix(matrix src, int n, int m)
{
        int i;
        matrix dst = calloc(n, sizeof(vector));
        for (i = 0; i < n; i++)
        {
                dst[i] = calloc(m, sizeof(double));
                memcpy(dst[i], src[i], m * sizeof(double));
        }
        return dst;
}

matrix attached_matrix(matrix mtrx, int n)
{
        /*creat attached matrix : [A|I]*/
        int i, j;
        matrix mtrx2 = calloc(n, sizeof(vector));
        for (i = 0; i < n; i++)
        {
                mtrx2[i] = calloc(2 * n, sizeof(double));
                memcpy(mtrx2[i], mtrx[i], n * sizeof(double));
                for (j = n; j < 2 * n; j++)
                        mtrx2[i][j] = (double)(i == j - n);
        }
        return mtrx2;
}

void free_matrix(matrix mtrx, int n)
{
        int i;
        for (i = 0; i < n; i++)
                free(mtrx[i]);
```

```
	free(mtrx);
}

int find_max(matrix mtrx, int n, int numstr)
{
	/*find maximal element in numstr'st string */
	/*need for Gaussian elimination with pivoting */
	/*matrix[column][string] */
	int i, tmp = numstr;
	double max = fabs(mtrx[numstr][numstr]);
	for (i = numstr; i < n; i++)
	{
		if (max < fabs(mtrx[i][numstr]))
		{
			max = fabs(mtrx[i][numstr]);
			tmp = i;
		}
	}
	if (max < EPSILON)
		puts("matrix is degenerate");
	return tmp;
}

int find_not_null_clm(matrix mtrx, int n, int num)
{
	/*find not null element in num'st column*/
	int i;
	for (i = num; i < n; i++)
	{
		if (fabs(mtrx[num][i]) > EPSILON)
			return i;
	}
	puts("matrix is degenerate");
	return -1;
}

int my_swap(matrix mtrx, int *form, int new, int old)
{
	/*swap new'st and old'st columns (or strings) in matrix*/
	/*if necessary describe swap variables*/
	/*returned value: 1 if wasn't swap, -1 if was swap*/
	if (new == old)
		return 1;
	vector hlp = mtrx[new];
	mtrx[new] = mtrx[old];
	mtrx[old] = hlp;
	if( form != NULL)
	{
		int hlp2 = form[new];
		form[new] = form[old];
		form[old] = hlp2;
	}
```

```
	return -1;
}

void stdtion_of_string(matrix mtrx, int numstr, int n)
{
	/*divide by the dioganal element of string*/
	/*for Gaussian elimination with pivoting*/
	/*matrix[column][string]*/
	double a = mtrx[numstr][numstr];
	int i;
	mtrx[numstr][numstr] = 1.0;
	for (i = numstr + 1; i < n; i++)
	{
		mtrx[i][numstr] /= a;
	}
}

void stdtion_of_string2(matrix mtrx, int numstr, int n)
{
	/*divide by the dioganal element of string*/
	int i;
	double a = mtrx[numstr][numstr];
	for (i = numstr; i < n; i++)
	{
		mtrx[numstr][i] /= a;
	}
}

void subtract_string(matrix mtrx, int num1, int num2, int n)
{
	/*subtract num1'st string of num2'st string*/
	/*for Gaussian elimination with pivoting*/
	/*matrix[column][string]*/
	double factor = mtrx[num1][num2];
	int i;
	mtrx[num1][num2] = 0.0;
	for (i = num1 + 1; i < n; i++)
	{
		mtrx[i][num2] -= mtrx[i][num1] * factor;
	}
}

void subtract_string2(matrix mtrx, int num1, int num2, int n)
{
	/*subtract num1'st string of num2'st string*/
	double factor = mtrx[num2][num1];
	int i;
	mtrx[num2][num1] = 0.0;
	for (i = num1 + 1; i < n; i++)
	{
		mtrx[num2][i] -= mtrx[num1][i] * factor;
	}
```

```
}

void calculate_ans2(matrix mtrx, int numstr, int *a, vector ans, int n)
{
	/*calculate answer for Gaussian elimination*/
	double tmp = mtrx[numstr][n];
	int i;
	for (i = numstr + 1; i < n; i++)
	{
		tmp -= (mtrx[numstr][i] * ans[i]);
	}
	ans[numstr] = tmp;
}

void calculate_ans(matrix mtrx, int numstr, int *form, vector ans, int n)
{
	/*calculate answer for Gaussian elimination with pivoting*/
	/*matrix[column][sting]*/
	double tmp = mtrx[n][numstr];
	int i;
	for (i = numstr + 1; i < n; i++)
	{
		tmp -= (mtrx[i][numstr] * ans[form[i]]);
	}
	ans[form[numstr]] = tmp;
}

vector Gauss(matrix src, int n, double *determ, int mode)
{
	matrix mtrx;
	int *form, sign = 1;
	find_func find;
	stdtion_func stdtion;
	subtract_func subtract;
	calculate_func calculate;
	//two methods differ in set of functions
	//choose set of function
	if (mode == PVT)
	{
	/*Gaussian elimination with pivoting*/
	/*matrix is represented as matrix[column][string]*/
		int i;
		form = calloc (n, sizeof(int));
		for (i = 0; i < n; i++)
			form[i] = i;
		find = find_max;
		stdtion = stdtion_of_string;
		subtract = subtract_string;
		calculate = calculate_ans;
		mtrx = conversion_matrix(src, n);
	}
	else
```

```
	{
	/*Gaussian elimination*/
		form = NULL;
		find = find_not_null_clm;
		stdtion = stdtion_of_string2;
		subtract = subtract_string2;
		calculate = calculate_ans2;
		mtrx = copy_matrix(src, n, n + 1);
	}
	vector ans;
	double det = 1;
	int i, hlp, j;
	for (i = 0; i < n; i++) //forward stroke Gaussian elimination
	{
		hlp = find(mtrx, n, i);
		sign *= my_swap(mtrx, form, hlp, i);
		if (determ != NULL)
			det *= mtrx[i][i]; //if necessary calculate deterinate
		stdtion(mtrx, i, n + 1);
		for (j = i + 1; j < n; j++)
		{
			subtract(mtrx, i, j, n + 1);
		}
	}
	ans = calloc(n, sizeof(double));
	for (i = n - 1; i >= 0; i--)
	{
		calculate(mtrx, i, form, ans, n); //reversal Gaussian elimination
	}
	if (determ != NULL)
		*determ = sign * det;
	if (form != NULL)
		free(form);
	mode == PVT ? free_matrix(mtrx, n + 1) : free_matrix(mtrx, n);
	return ans;
}


matrix inverse(matrix src, int n)
{
	/*calculate inverse matrix*/
	matrix mtrx = attached_matrix(src, n);
	int tmp, i, j;
	for (i = 0; i < n; i++)
	{
		tmp = find_not_null_clm(mtrx, n, i);
		my_swap(mtrx, NULL, tmp, i);
		stdtion_of_string2(mtrx, i, 2 * n);
		for (j = i + 1; j < n; j++)
		{
			subtract_string2(mtrx, i, j, 2 * n);
		}
```

```
        }
        for (i = n - 2; i >= 0; i--)
        {
                for (j = i + 1; j < n; j++)
                {
                        subtract_string2(mtrx, j, i, 2 * n);
                }
        }
        matrix answer = calloc(n, sizeof(vector));
        for (i = 0; i < n; i++)
        {
                answer[i] = calloc(n, sizeof(double));
                for (j = 0; j < n; j++)
                {
                        answer[i][j]  = mtrx[i][n + j];
                }
        }
        free_matrix(mtrx, n);
        return answer;
}

void report_answer(vector answ1, vector answ2, double det, matrix mtrx, int n)
{
        int i, j;
        puts("Решение СЛАУ методом Гаусса без выбора главного элемента:");
        for (i = 0; i < n; i++)
                printf("X%d = %.3lf; ", i + 1, answ1[i]);
        puts("\n");
        puts("Решение СЛАУ методом Гаусса с выбором главного элемента:");
        for (i = 0; i < n; i++)
                printf("X%d = %.3lf; ", i+1, answ2[i]);
        puts("\n");
        printf("Определитель матрицы:\ndet(A) = %.3lf\n", det);
        printf("\n");
        puts("Обратная матрица:");
        for (i = 0; i < n; i++)
        {
                for (j = 0; j < n; j++)
                        printf("%.1lf ", mtrx[i][j]);
                printf("\n");
        }
}

int main (int argc, char **argv)
{
        int n;
        if(argc < 2)
        {
                puts("Too few parameters");
                puts("Write type of submission matrix (1 -- from file, 2 -- formula)");
                return 0;
        }
```

```
	int submission;
	sscanf(argv[1],"%d", &submission);
	double determ;
	vector answer1, answer2;
	matrix mtrx, reverse;
	double x = 1;
	FILE *f;
	if (submission == 1)
	{
		if (argc >=2)
			{if ((f = fopen(argv[2], "r")) == NULL)
			{
				puts("error with file");
				return -1;
			}}
		else
			if( (f = fopen("input.txt", "r")) == NULL)
			{
				puts("please try again and se file");
				return -1;
			}
		mtrx = read_matrix(&n, f);
	}
	else
	{
		if (argc >= 2)
			sscanf(argv[2], "%lf", &x);
		mtrx = creat_matrix(&n, x);
	}
		answer2 = Gauss(mtrx, n, NULL, EAS);
	answer1 = Gauss(mtrx, n, &determ, PVT);
	reverse = inverse(mtrx, n);
	report_answer(answer1, answer2, determ, reverse, n);
	free_matrix(reverse, n);
	free_matrix(mtrx, n);
	return 0;
}
```

Приложение 2

Система тестов, подтверждающая решения СЛАУ

I. Задание элементов матрицы системы и ее правой части во входном файле

Приложение 1-9

1) Система уравнений
2x1 - 5x2 + 3x3 + x4 = 5
3x1 – 7x2 + 3x3 –x4 = -1
5x1 – 9x2 + 6x3 +2x4 = 7
4x1 – 6x2 + 3x3 + x4 = 8

Ответ программы:
Решение СЛАУ методом Гаусса без выбора главного элемента:
X1 = 0.000000; X2 = -3.000000; X3 = -5.333333; X4 = 6.000000;

Решение СЛАУ методом Гаусса с выбором главного элемента:
X1 = 0.000000; X2 = -3.000000; X3 = -5.333333; X4 = 6.000000;

Определитель матрицы:
det(A) = 18.000000

Обратная матрица:
-1.000000 0.000000 0.333333 0.333333
-1.000000 -0.000000 0.666667 -0.333333
-1.000000 0.166667 1.055556 -0.944444
1.000000 -0.500000 -0.500000 0.500000

Ответ ресурса http://www.wolframalpha.com совпадает с ответом программы.

2) Система уравнений
4x1 + 3x2 -9x3 = 9
2x1 + 5x2 - 8x3 = 8
2x1 + 16x2 - 14x3 = 24

Ответ программы:
Решение СЛАУ методом Гаусса без выбора главного элемента:
X1 = 3.000000; X2 = 2.000000; X3 = 1.000000;

Решение СЛАУ методом Гаусса с выбором главного элемента:
X1 = 3.000000; X2 = 2.000000; X3 = 1.000000;

Определитель матрицы:
det(A) = 70.000000

Обратная матрица:
0.828571 -1.457143 0.300000
0.171429 -0.542857 0.200000
0.314286 -0.828571 0.200000

Ответ ресурса http://www.wolframalpha.com совпадает с ответом программы.

3) Система уравнений:

12x1 + 14x2 - 15x3 + 24x4 + 27x5 = 5
16x1 - 18x2 — 22x3 + 29x4 + 37x5 = 8
18x1 + 20x2 — 21x3 + 32x4 + 41x5 = 9
10x1 + 12x2 — 16x3 + 20x4 + 23x5 = 4
46x1 — 2x2 + 16x3 — 33x4 + 1x5 = 27

Ответ программы:

Решение СЛАУ методом Гаусса без выбора главного элемента:
X1 = 0.516764; X2 = -1.183990; X3 = 0.017603; X4 = -0.000000; X5 = 0.579212;

Решение СЛАУ методом Гаусса с выбором главного элемента:
X1 = 0.516764; X2 = -1.183990; X3 = 0.017603; X4 = 0.000000; X5 = 0.579212;

Определитель матрицы:
det(A) = -4772.000000

Обратная матрица:
0.600168 0.620704 -0.659681 -0.528080 0.022213
-2.386840 -4.024728 3.215004 3.545683 -0.006287
0.030176 -0.273261 0.257334 -0.054484 -0.001676
1.000000 1.000000 -1.000000 -1.000000 0.000000
0.135792 0.770327 -0.341995 -0.745180 -0.007544

Ответ http://www.wolframalpha.com совпадает с ответом программы.

II. **Задание элементов матрицы системы и ее правой части путем задания специальных формул.**

Приложение 2(п 2-2)

Элементы матрицы вычисляются по формуле:

$$A_{ij}=\begin{cases} q_M^{i+j}, i\neq j \\ (q_M-1)^{i+j}, i=j, \end{cases}$$

$$где\ q_M=1.001-2\cdot 10^{-3}, i, j=1,\dots,n$$

Элементы вектора f (вектора правой части) задаются формулами

$$b_i=\left|x-\frac{n}{10}\right|\cdot i\cdot\sin(x), i=1,\dots,n$$

M = 2, n = 40

Ответ программы для x = 0.7

Решение СЛАУ методом Гаусса без выбора главного элемента:
X1 = -29.772487; X2 = 0.012367; X3 = 0.024688; X4 = 0.036961; X5 = 0.049185; X6 = 0.061359; X7 = 0.073479; X8 = 0.085546; X9 = 0.097557; X10 = 0.109510; X11 = 0.121404; X12 = 0.133237; X13 = 0.145007; X14 = 0.156713; X15 = 0.168352; X16 = 0.179923; X17 = 0.191423; X18 = 0.202852; X19 = 0.214207; X20 = 0.225486; X21 = 0.236688; X22 = 0.247809; X23 = 0.258849; X24 = 0.269805; X25 = 0.280676; X26 = 0.291459; X27 = 0.302152; X28 = 0.312752; X29 = 0.323259; X30 = 0.333670; X31 = 0.343982; X32 = 0.354193; X33 = 0.364301; X34 = 0.374305; X35 = 0.384201; X36 = 0.393987; X37 = 0.403661; X38 = 0.413221; X39 = 0.422663; X40 = 0.431987;

Решение СЛАУ методом Гаусса с выбором главного элемента:
X1 = -29.772487; X2 = 0.012367; X3 = 0.024688; X4 = 0.036961; X5 = 0.049185; X6 = 0.061359; X7 = 0.073479; X8 = 0.085546; X9 = 0.097557; X10 = 0.109510; X11 = 0.121404; X12 = 0.133237; X13 = 0.145007; X14 = 0.156713; X15 = 0.168352; X16 = 0.179923; X17 = 0.191423; X18 = 0.202852; X19 = 0.214207; X20 = 0.225486; X21 = 0.236688; X22 = 0.247809; X23 = 0.258849; X24 = 0.269805; X25 = 0.280676; X26 = 0.291459; X27 = 0.302152; X28 = 0.312752; X29 = 0.323259; X30 = 0.333670; X31 = 0.343982; X32 = 0.354193; X33 = 0.364301; X34 = 0.374305; X35 = 0.384201; X36 = 0.393987; X37 = 0.403661; X38 = 0.413221; X39 = 0.422663; X40 = 0.431987;

Определитель матрицы:
det(A) = 2.254305

Обратная матрица:
-9.876931 0.968049 0.935656 0.902842 0.869596 0.835913 0.801789 0.767219 0.732201 0.696729 0.660800 0.624409 0.587552 0.550224 0.512422 0.474141 0.435376 0.396123 0.356378 0.316136 0.275392 0.234142 0.192382 0.150106 0.107310 0.063989 0.020139 -0.024246 -0.069171 -0.114639 -0.160657 -0.207228 -0.254359 -0.302055 -0.350319 -0.399158 -0.448577 -0.498580 -0.549173 -0.600362

0.968794 -1.005995 0.000083 0.000126 0.000168 0.000212 0.000256 0.000300 0.000345 0.000391 0.000437 0.000483 0.000530 0.000578 0.000626 0.000675 0.000725 0.000775 0.000825 0.000876

0.000928 0.000980 0.001033 0.001087 0.001141 0.001196 0.001251 0.001307 0.001364 0.001421
0.001479 0.001538 0.001597 0.001657 0.001717 0.001779 0.001841 0.001903 0.001967 0.002031

0.937160 0.000083 -1.011923 0.000253 0.000339 0.000426 0.000515 0.000604 0.000694 0.000786
0.000879 0.000972 0.001067 0.001163 0.001260 0.001359 0.001458 0.001558 0.001660 0.001763
0.001867 0.001972 0.002079 0.002187 0.002296 0.002406 0.002517 0.002630 0.002744 0.002859
0.002976 0.003094 0.003213 0.003334 0.003455 0.003579 0.003703 0.003829 0.003957 0.004086

0.905122 0.000126 0.000253 -1.017809 0.000511 0.000643 0.000776 0.000911 0.001048 0.001186
0.001326 0.001467 0.001610 0.001755 0.001902 0.002050 0.002200 0.002352 0.002505 0.002660
0.002818 0.002976 0.003137 0.003300 0.003464 0.003630 0.003798 0.003969 0.004141 0.004315
0.004490 0.004668 0.004848 0.005030 0.005214 0.005400 0.005588 0.005778 0.005971 0.006165

0.872666 0.000168 0.000339 0.000511 -1.023641 0.000863 0.001041 0.001222 0.001406 0.001591
0.001778 0.001968 0.002160 0.002354 0.002551 0.002750 0.002951 0.003154 0.003360 0.003569
0.003779 0.003992 0.004208 0.004426 0.004646 0.004869 0.005095 0.005323 0.005554 0.005787
0.006023 0.006262 0.006503 0.006747 0.006994 0.007243 0.007496 0.007751 0.008008 0.008269

0.839790 0.000212 0.000426 0.000643 0.000863 -1.029416 0.001310 0.001537 0.001767 0.002001
0.002236 0.002475 0.002716 0.002961 0.003208 0.003458 0.003711 0.003967 0.004225 0.004487
0.004752 0.005020 0.005291 0.005565 0.005843 0.006123 0.006407 0.006694 0.006984 0.007277
0.007574 0.007874 0.008177 0.008484 0.008795 0.009108 0.009425 0.009746 0.010070 0.010398

0.806488 0.000256 0.000514 0.000776 0.001041 0.001310 -1.035131 0.001856 0.002134 0.002415
0.002700 0.002988 0.003279 0.003574 0.003872 0.004174 0.004479 0.004788 0.005101 0.005417
0.005737 0.006060 0.006388 0.006718 0.007053 0.007392 0.007734 0.008081 0.008431 0.008785
0.009143 0.009505 0.009872 0.010242 0.010617 0.010995 0.011378 0.011765 0.012157 0.012553

0.772758 0.000300 0.000604 0.000911 0.001222 0.001537 0.001856 -1.040782 0.002504 0.002834
0.003168 0.003507 0.003849 0.004195 0.004545 0.004899 0.005257 0.005620 0.005987 0.006358
0.006733 0.007113 0.007497 0.007885 0.008278 0.008675 0.009077 0.009484 0.009895 0.010310
0.010731 0.011156 0.011586 0.012020 0.012460 0.012904 0.013354 0.013808 0.014268 0.014732

0.738595 0.000345 0.000694 0.001047 0.001405 0.001767 0.002133 0.002504 -1.046367 0.003259
0.003643 0.004031 0.004425 0.004823 0.005225 0.005632 0.006044 0.006461 0.006883 0.007309
0.007741 0.008177 0.008619 0.009066 0.009517 0.009974 0.010436 0.010903 0.011376 0.011854
0.012337 0.012826 0.013320 0.013820 0.014325 0.014836 0.015353 0.015875 0.016403 0.016937

0.703995 0.000390 0.000785 0.001185 0.001590 0.002000 0.002414 0.002834 0.003258 -1.051882
0.004123 0.004563 0.005008 0.005458 0.005913 0.006374 0.006841 0.007312 0.007790 0.008272
0.008761 0.009255 0.009754 0.010260 0.010771 0.011288 0.011811 0.012340 0.012874 0.013415
0.013962 0.014515 0.015075 0.015640 0.016212 0.016791 0.017375 0.017966 0.018564 0.019169

0.668954 0.000436 0.000878 0.001325 0.001777 0.002235 0.002699 0.003168 0.003642 0.004122
-1.057324 0.005100 0.005597 0.006101 0.006610 0.007125 0.007646 0.008174 0.008707 0.009247
0.009792 0.010345 0.010903 0.011468 0.012039 0.012617 0.013202 0.013793 0.014391 0.014995
0.015606 0.016225 0.016850 0.017482 0.018121 0.018768 0.019421 0.020082 0.020750 0.021426

0.633468 0.000483 0.000972 0.001466 0.001967 0.002474 0.002986 0.003505 0.004030 0.004562
0.005099 -1.062689 0.006194 0.006751 0.007314 0.007885 0.008461 0.009045 0.009635 0.010232
0.010836 0.011447 0.012065 0.012690 0.013323 0.013962 0.014609 0.015263 0.015924 0.016593
0.017270 0.017954 0.018646 0.019345 0.020053 0.020768 0.021491 0.022222 0.022962 0.023709

0.597534 0.000530 0.001066 0.001609 0.002159 0.002715 0.003277 0.003847 0.004423 0.005006
0.005596 0.006193 -1.067974 0.007409 0.008027 0.008653 0.009286 0.009926 0.010574 0.011229
0.011892 0.012563 0.013241 0.013927 0.014621 0.015322 0.016032 0.016750 0.017476 0.018210
0.018953 0.019703 0.020463 0.021230 0.022007 0.022791 0.023585 0.024388 0.025199 0.026019

0.561146 0.000578 0.001162 0.001754 0.002352 0.002958 0.003572 0.004192 0.004820 0.005456

0.006099 0.006750 0.007408 -1.073175 0.008748 0.009430 0.010120 0.010818 0.011524 0.012238
0.012960 0.013691 0.014430 0.015178 0.015934 0.016699 0.017472 0.018254 0.019046 0.019846
0.020655 0.021473 0.022300 0.023137 0.023983 0.024838 0.025703 0.026578 0.027462 0.028356

0.524301 0.000626 0.001259 0.001900 0.002549 0.003205 0.003870 0.004542 0.005222 0.005911
0.006608 0.007312 0.008026 0.008747 -1.078289 0.010216 0.010964 0.011720 0.012484 0.013258
0.014041 0.014832 0.015633 0.016443 0.017262 0.018091 0.018929 0.019776 0.020633 0.021500
0.022377 0.023263 0.024159 0.025066 0.025982 0.026909 0.027846 0.028793 0.029751 0.030720

0.486994 0.000674 0.001357 0.002048 0.002747 0.003455 0.004171 0.004896 0.005629 0.006371
0.007122 0.007882 0.008651 0.009428 0.010215 -1.083311 0.011817 0.012632 0.013456 0.014290
0.015134 0.015987 0.016850 0.017723 0.018606 0.019499 0.020402 0.021316 0.022239 0.023174
0.024118 0.025074 0.026040 0.027017 0.028005 0.029004 0.030013 0.031035 0.032067 0.033111

0.449222 0.000724 0.001456 0.002197 0.002948 0.003707 0.004476 0.005253 0.006040 0.006836
0.007642 0.008458 0.009283 0.010117 0.010962 0.011816 -1.088238 0.013555 0.014439 0.015334
0.016239 0.017155 0.018081 0.019018 0.019965 0.020923 0.021893 0.022873 0.023864 0.024866
0.025880 0.026905 0.027942 0.028990 0.030050 0.031122 0.032206 0.033301 0.034409 0.035529

0.410980 0.000774 0.001556 0.002349 0.003151 0.003962 0.004784 0.005615 0.006456 0.007307
0.008169 0.009040 0.009922 0.010814 0.011717 0.012630 0.013554 -1.093065 0.015434 0.016390
0.017358 0.018336 0.019326 0.020327 0.021340 0.022364 0.023400 0.024448 0.025507 0.026579
0.027662 0.028758 0.029866 0.030986 0.032119 0.033265 0.034423 0.035594 0.036779 0.037976

0.372264 0.000824 0.001658 0.002502 0.003356 0.004221 0.005096 0.005981 0.006877 0.007783
0.008701 0.009629 0.010568 0.011519 0.012480 0.013453 0.014437 0.015432 -1.097789 0.017458
0.018489 0.019531 0.020585 0.021652 0.022730 0.023821 0.024925 0.026041 0.027169 0.028311
0.029465 0.030632 0.031812 0.033005 0.034212 0.035432 0.036666 0.037914 0.039175 0.040450

0.333070 0.000875 0.001760 0.002657 0.003564 0.004482 0.005411 0.006351 0.007302 0.008265
0.009239 0.010225 0.011222 0.012231 0.013252 0.014285 0.015330 0.016387 0.017457 -1.102406
0.019633 0.020739 0.021859 0.022992 0.024137 0.025295 0.026467 0.027652 0.028850 0.030062
0.031288 0.032527 0.033780 0.035048 0.036329 0.037625 0.038935 0.040259 0.041598 0.042952

0.293392 0.000926 0.001864 0.002813 0.003774 0.004746 0.005730 0.006725 0.007733 0.008752
0.009784 0.010828 0.011884 0.012952 0.014033 0.015127 0.016234 0.017353 0.018485 0.019631
-1.106911 0.021962 0.023147 0.024346 0.025559 0.026786 0.028027 0.029282 0.030550 0.031834
0.033132 0.034444 0.035771 0.037113 0.038470 0.039842 0.041229 0.042632 0.044050 0.045483

0.253228 0.000979 0.001969 0.002972 0.003986 0.005013 0.006052 0.007104 0.008168 0.009245
0.010335 0.011437 0.012553 0.013681 0.014823 0.015979 0.017147 0.018330 0.019526 0.020736
0.021960 -1.111299 0.024450 0.025717 0.026998 0.028294 0.029604 0.030930 0.032270 0.033625
0.034996 0.036382 0.037784 0.039202 0.040635 0.042084 0.043549 0.045031 0.046529 0.048043

0.212572 0.001031 0.002075 0.003132 0.004201 0.005283 0.006379 0.007487 0.008608 0.009743
0.010892 0.012054 0.013229 0.014419 0.015622 0.016840 0.018072 0.019318 0.020578 0.021853
0.023143 0.024448 -1.115567 0.027103 0.028453 0.029818 0.031200 0.032596 0.034009 0.035437
0.036882 0.038343 0.039820 0.041314 0.042824 0.044352 0.045896 0.047457 0.049036 0.050632

0.171419 0.001085 0.002183 0.003294 0.004418 0.005557 0.006708 0.007874 0.009054 0.010247
0.011455 0.012677 0.013913 0.015164 0.016430 0.017711 0.019006 0.020317 0.021642 0.022984
0.024340 0.025712 0.027100 -1.119709 0.029924 0.031360 0.032813 0.034282 0.035768 0.037270
0.038789 0.040326 0.041879 0.043450 0.045039 0.046645 0.048269 0.049911 0.051571 0.053250

0.129766 0.001139 0.002291 0.003458 0.004638 0.005833 0.007042 0.008266 0.009504 0.010757
0.012025 0.013307 0.014605 0.015919 0.017247 0.018591 0.019951 0.021327 0.022719 0.024126
0.025551 0.026991 0.028448 0.029922 -1.123722 0.032920 0.034445 0.035987 0.037546 0.039123
0.040718 0.042331 0.043962 0.045611 0.047278 0.048964 0.050669 0.052393 0.054135 0.055897

0.087609 0.001193 0.002401 0.003623 0.004860 0.006112 0.007379 0.008662 0.009959 0.011272
0.012601 0.013945 0.015305 0.016681 0.018073 0.019482 0.020907 0.022349 0.023807 0.025282
0.026775 0.028284 0.029811 0.031355 0.032917 -1.127599 0.036095 0.037710 0.039345 0.040997
0.042668 0.044358 0.046067 0.047795 0.049543 0.051310 0.053096 0.054902 0.056728 0.058575

0.044941 0.001248 0.002512 0.003791 0.005085 0.006395 0.007721 0.009062 0.010420 0.011793
0.013183 0.014590 0.016013 0.017452 0.018909 0.020383 0.021874 0.023382 0.024908 0.026451
0.028012 0.029591 0.031189 0.032805 0.034439 0.036091 -1.131337 0.039454 0.041163 0.042892
0.044641 0.046409 0.048197 0.050005 0.051833 0.053681 0.055550 0.057440 0.059350 0.061282

0.001759 0.001304 0.002624 0.003960 0.005312 0.006681 0.008066 0.009467 0.010885 0.012320
0.013772 0.015242 0.016728 0.018232 0.019754 0.021293 0.022851 0.024427 0.026021 0.027633
0.029264 0.030914 0.032582 0.034270 0.035977 0.037704 0.039450 -1.134930 0.043002 0.044808
0.046635 0.048482 0.050350 0.052238 0.054148 0.056079 0.058032 0.060006 0.062002 0.064019

-0.041942 0.001361 0.002738 0.004132 0.005542 0.006970 0.008415 0.009877 0.011356 0.012853
0.014368 0.015901 0.017452 0.019021 0.020608 0.022214 0.023839 0.025483 0.027146 0.028828
0.030529 0.032251 0.033991 0.035752 0.037533 0.039334 0.041156 0.042999 -1.138374 0.046746
0.048652 0.050579 0.052527 0.054497 0.056490 0.058504 0.060541 0.062600 0.064682 0.066788

-0.086166 0.001418 0.002853 0.004305 0.005775 0.007262 0.008767 0.010291 0.011832 0.013392
0.014970 0.016567 0.018183 0.019818 0.021472 0.023146 0.024839 0.026551 0.028284 0.030036
0.031809 0.033602 0.035416 0.037251 0.039106 0.040983 0.042881 0.044801 0.046742 -1.141661
0.050691 0.052698 0.054729 0.056781 0.058857 0.060956 0.063078 0.065224 0.067393 0.069587

-0.130919 0.001475 0.002969 0.004480 0.006009 0.007557 0.009124 0.010709 0.012314 0.013937
0.015579 0.017241 0.018923 0.020624 0.022346 0.024087 0.025849 0.027631 0.029434 0.031258
0.033103 0.034969 0.036857 0.038766 0.040697 0.042650 0.044625 0.046623 0.048643 0.050686
-1.144789 0.054842 0.056954 0.059091 0.061251 0.063435 0.065644 0.067876 0.070134 0.072417

-0.176205 0.001534 0.003086 0.004657 0.006247 0.007856 0.009485 0.011133 0.012800 0.014488
0.016195 0.017923 0.019671 0.021440 0.023229 0.025039 0.026871 0.028723 0.030598 0.032494
0.034411 0.036351 0.038313 0.040298 0.042305 0.044336 0.046389 0.048466 0.050566 0.052689
0.054837 -1.147750 0.059205 0.061426 0.063671 0.065942 0.068237 0.070559 0.072905 0.075278

-0.222029 0.001593 0.003205 0.004836 0.006487 0.008158 0.009849 0.011561 0.013292 0.015045
0.016818 0.018612 0.020427 0.022264 0.024122 0.026002 0.027904 0.029828 0.031774 0.033743
0.035734 0.037749 0.039786 0.041847 0.043932 0.046040 0.048172 0.050328 0.052509 0.054715
0.056945 0.059200 -1.150540 0.063787 0.066118 0.068476 0.070860 0.073270 0.075707 0.078171

-0.268395 0.001652 0.003325 0.005017 0.006730 0.008464 0.010218 0.011993 0.013790 0.015608
0.017447 0.019309 0.021192 0.023097 0.025025 0.026975 0.028948 0.030944 0.032963 0.035006
0.037072 0.039161 0.041275 0.043413 0.045576 0.047763 0.049975 0.052212 0.054474 0.056762
0.059076 0.061415 0.063781 -1.153152 0.068593 0.071038 0.073511 0.076012 0.078540 0.081096

-0.315309 0.001713 0.003446 0.005200 0.006976 0.008772 0.010591 0.012431 0.014293 0.016177
0.018084 0.020013 0.021965 0.023940 0.025938 0.027959 0.030004 0.032073 0.034165 0.036282
0.038424 0.040590 0.042781 0.044997 0.047238 0.049505 0.051797 0.054116 0.056461 0.058832
0.061230 0.063655 0.066107 0.068587 -1.155580 0.073629 0.076192 0.078784 0.081404 0.084053

-0.362775 0.001773 0.003568 0.005385 0.007224 0.009085 0.010968 0.012873 0.014802 0.016753
0.018727 0.020725 0.022746 0.024791 0.026860 0.028954 0.031071 0.033214 0.035381 0.037573
0.039791 0.042034 0.044303 0.046597 0.048918 0.051266 0.053640 0.056041 0.058469 0.060925
0.063408 0.065919 0.068459 0.071026 0.073623 -1.157820 0.078902 0.081586 0.084299 0.087043

-0.410799 0.001835 0.003692 0.005572 0.007475 0.009400 0.011349 0.013320 0.015316 0.017335
0.019378 0.021445 0.023536 0.025653 0.027793 0.029959 0.032151 0.034367 0.036610 0.038878
0.041173 0.043494 0.045841 0.048216 0.050617 0.053046 0.055503 0.057987 0.060500 0.063041
0.065610 0.068209 0.070836 0.073493 0.076179 0.078896 -1.159863 0.084419 0.087227 0.090065

-0.459385 0.001897 0.003818 0.005761 0.007728 0.009719 0.011734 0.013773 0.015836 0.017923
0.020035 0.022173 0.024335 0.026523 0.028737 0.030976 0.033242 0.035534 0.037852 0.040197
0.042570 0.044970 0.047397 0.049852 0.052335 0.054846 0.057386 0.059955 0.062553 0.065180
0.067836 0.070523 0.073239 0.075986 0.078764 0.081572 0.084412 -1.161705 0.090186 0.093121

-0.508539 0.001960 0.003944 0.005953 0.007985 0.010042 0.012123 0.014230 0.016361 0.018518
0.020700 0.022908 0.025143 0.027403 0.029690 0.032004 0.034345 0.036713 0.039108 0.041531
0.043982 0.046461 0.048969 0.051506 0.054071 0.056666 0.059290 0.061944 0.064628 0.067342
0.070087 0.072862 0.075669 0.078507 0.081377 0.084278 0.087212 0.090178 -1.163339 0.096209

-0.558265 0.002024 0.004073 0.006146 0.008244 0.010368 0.012517 0.014692 0.016892 0.019119
0.021372 0.023652 0.025959 0.028293 0.030654 0.033043 0.035460 0.037904 0.040378 0.042879
0.045410 0.047970 0.050559 0.053178 0.055826 0.058505 0.061215 0.063955 0.066726 0.069528
0.072362 0.075227 0.078125 0.081055 0.084018 0.087014 0.090043 0.093105 0.096201 -1.164757