**КУРСОВАЯ РАБОТА**

**Оценка параметров электрической цепи**

по дисциплине «Вычислительная математика»

Выполнил   
студент гр. 5130904/30007 Голиков В.С.

Руководитель Скуднева Е.В.

ЗАДАНИЕ

НА ВЫПОЛНЕНИЕ КУРСОВОГО ПРОЕКТА (КУРСОВОЙ РАБОТЫ)

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| студенту группы | | 5130904/30007 | |  | Голиков Валентин Сергеевич | | | | | | | | |
|  | | *(номер группы)* | |  | *(фамилия, имя, отчество)* | | | | | | | | |
| ***1. Тема проекта (работы)*** | | | | ***Оценка параметров электрической цепи*** | | | | | | | | | |
|  | | | | | | | | | | | | | |
| ***2. Срок сдачи студентом законченного проекта (работы)*** | | | | | | | | | | | | | *10.06.2025* |
| ***3. Исходные данные к проекту (работе)*** | | | | | | | | | *Система уравнений для* | | | | |
| *Нахождения параметров R, R2, e2; Уравнения для вычисления параметров* | | | | | | | | | | | | | |
| *L1, e1, пакет программ: FMIN, QUANC8, ZEROIN, DECOMP, SOLVE* | | | | | | | | | | | | | |
|  | | | | | | | | | | | | | |
|  | | | | | | | | | | | | | |
| ***4.Содержание пояснительной записки***(перечень подлежащих разработке вопросов): введение, основная часть (раскрывается структура основной части), заключение, список использованных источников, приложения)*.* | | | | | | | | | | | | | |
| ***Формулировка задания, Анализ цепи, Вычисления: -вычисление*** | | | | | | | | | | | | | |
| ***параметров цепи, -вычисление ёмкости конденсатора, -результаты*** | | | | | | | | | | | | | |
| ***вычислений, -устойчивость модели; Вывод, Приложение А (скриншоты)*** | | | | | | | | | | | | | |
| ***Приложение Б (код программы)*** | | | | | | | | | | | | | |
| Примерный объем пояснительной записки | | | | | | ***21*** | | | | страниц машинописного | | | |
| текста | | | | | | | | | | | | | |
| 5. Перечень графического материала ( с указанием обязательных чертежей и | | | | | | | | | | | | | |
| плакатов) | ---- | | | | | | | | | | | | |
|  | | | | | | | | | | | | | |
| 6. Консультанты | | | ---- | | | | | | | | | | |
|  | | | | | | | | | | | | | |
| 7. Дата получения задания: «07» марта 2025 г. | | | | | | | | | | | | | |
|  | | | | | | | | | | | | | |
| Руководитель | | | | |  | | |  | | |  | | |
|  | | | | | *(подпись)* | | |  | | | *(инициалы, фамилия)* | | |
| Задание принял к исполнению | | | | |  | |  | | | |  | | |
|  | | | | | *(подпись)* | |  | | | | *(инициалы, фамилия)* | | |
|  | | | | | | | | | | | | 07.03.2025 | |
|  | | | | | | | | | | | | *(дата)* | |

**Оглавление**

[1. Формулировка задания 4](#_Toc197336702)

[2. Анализ цепи 5](#_Toc197336703)

[3. Вычисления 6](#_Toc197336704)

[3.1 Вычисление параметров цепи 6](#_Toc197336705)

[3.2 Вычисление ёмкости конденсатора 6](#_Toc197336706)

[3.3 Результаты вычислений 7](#_Toc197336707)

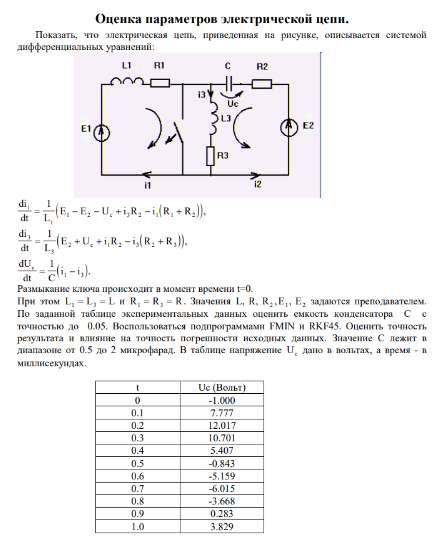
[3.4 Устойчивость модели 8](#_Toc197336708)

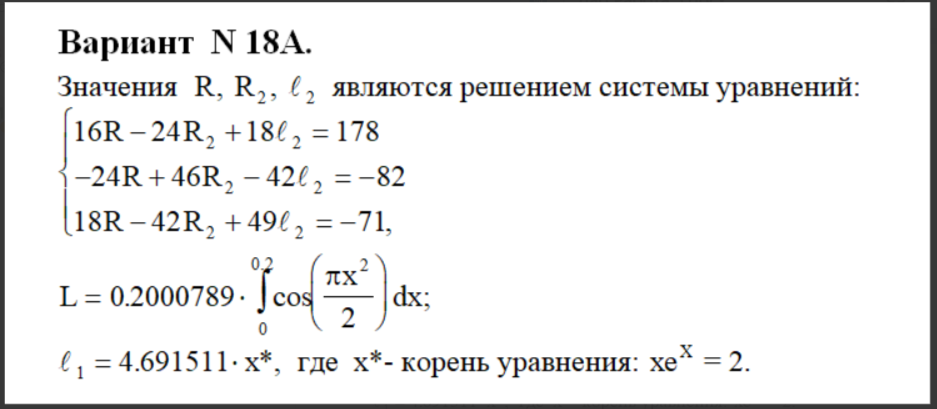
[4. Вывод 9](#_Toc197336709)

[Приложение А. Скриншоты выполнения программы 10](#_Toc197336710)

[Приложение Б. Код программы 11](#_Toc197336711)

# Формулировка задания





# Анализ цепи

Чтобы составить систему уравнений, описывающих цепь, воспользуемся вторым законом Кирхгофа, который гласит, что алгебраическая сумма падений напряжений в контуре равна алгебраической сумме ЭДС, циркулирующих в нем. После размыкания ключа выделим два контура в цепи: один – внешний, не содержащий L3 и R3, другой –внутренний, содержащий L3, R3, R2, C, E2.

* Для внешнего контура справедливо:

Подставляя все значения в верхнее уравнение, получим первое уравнение данной системы:

* Для внутреннего контура справедливо:

Аналогично выполняя подстановку получим второе уравнение данной системы:

* Ток на конденсаторе рассчитывается из соотношения

Отсюда получим третье и последнее уравнение данной системы:

# Вычисления

## 3.1 Вычисление параметров цепи

Для начала нам потребуется вычислить значения для R, R2, E1, E2, L.

Из условия задачи, R, R2 и E2 найдем через программное решение системы (пункт 1. Задание)

Используя подпрограммы DECOMP и SOLVE найдем решение данной системы. В итоге получаем **R = 40, R2 = 20, E2 = 1.**

Значение L найдем с помощью подпрограммы QUANC8. Результат ее выполнения: **L = 0.04**

Значение E1 найдем с использованием подпрограммы ZEROIN. Она позволяет вычислить корень уравнения на каком-то интервале. Ее результатом стало **E1 = 4**

(с кодом программы можно ознакомиться в Приложении Б)

## 3.2 Вычисление ёмкости конденсатора

Для расчета ёмкости конденсатора потребуется минимизировать функцию

Здесь F(C) – отклонение вычисленных значений от экспериментальных данных

Для определения ёмкости конденсатора на каждом этапе должны вызываться подпрограмма RKF45. В нее передаются следующие параметры:

- i1 = E1 / R1  
 - i2 = 0- UC = -1  
 - hprint = 0.0001  
 - ABSerr, RELerr = 1e-11 (максимально возможная погрешность сетки)

Для вызова RKF45 создана вспомогательная функция rkf45Driver – она позволяет задать все параметры, перечисленные выше, инициализировать RKF45 и выполнить расчеты. С кодом программы так же можно ознакомиться в Приложении Б.

## 3.3 Результаты вычислений

В результате мы получили C = 9,99989e-07 фарад = 0.999989 микрофарад, что удовлетворяет интервалу [0.5; 2] мкф.

|  |  |  |
| --- | --- | --- |
| t | UC(эксп) | UC |
| 0 | -1 | -1 |
| 0.1 | 7.777 | 7.77676 |
| 0.2 | 12.017 | 12.01725 |
| 0.3 | 10.701 | 10.70075 |
| 0.4 | 5.407 | 5.40675 |
| 0.5 | -0.843 | -0.84286 |
| 0.6 | -5.159 | -5.15867 |
| 0.7 | -6.015 | -6.01547 |
| 0.8 | -3.668 | -3.66789 |
| 0.9 | 0.283 | 0.2829 |
| 1 | 3.829 | 3.82921 |

Для оценки погрешности всех вычислений, требуется оценить погрешность на каждом этапе. Для этого достаточно пройтись вручную по каждой вызванной функции и оценить погрешность на каждом вычислении.

В связи с этим погрешность вычисления начальных параметров R1, R2, E2 через программ DECOMP и SOLVE составила 10-13

Погрешность параметра L, посчитанная в подпрограмме QUANC8 (параметр errest) равен 1.08526e-19, то есть погрешность составила примерно 10-19

В подпрограмме ZEROIN погрешность была установлена 10-10 и при проверке сходимости внутри программы все было в порядке, таким образом точность полученного значения (E1) составляет 10-10

Как показано в предыдущей таблице, вычисленные значения совпадают с экспериментальными данными полностью в двух разрядах. Однако, при округлении значений модели до трех знаков после запятой, совпадение становится полным. Это указывает на высокую точность вычисленных значений модели.

## 3.4 Устойчивость модели

Обратимся теперь к устойчивости модели относительно погрешности начальных данных. Для начала изменим C на 1% и решим данную систему. Полученные значения запишем в столбик под названием UC1

После этого изменим начальные данные на 1%, а именно параметры E1, E2, L, R и так же решим систему. Полученные значения запишем в столбик UC2

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| t | UC(эксп) | UC | UC1 | UC2 |
| 0 | -1 | -1 | -1 | -1 |
| 0.1 | 7.777 | 7.77676 | 7.69707 | 7.78595 |
| 0.2 | 12.017 | 12.01725 | 11,9346 | 12.068 |
| 0.3 | 10.701 | 10.70075 | 10,6967 | 10,8146 |
| 0.4 | 5.407 | 5.40675 | 5.50876 | 5.56454 |
| 0.5 | -0.843 | -0.84286 | -0.679727 | -0.699754 |
| 0.6 | -5.159 | -5.15867 | -5.02332 | -5.09721 |
| 0.7 | -6.015 | -6.01547 | -5.98654 | -6.07072 |
| 0.8 | -3.668 | -3.66789 | -3.76748 | -3.81887 |
| 0.9 | 0.283 | 0.2829 | 0.101713 | 0.106412 |
| 1 | 3.829 | 3.82921 | 3.65625 | 3.71253 |

Из вышеприведенной таблице видно, что значения UC1 немного отклонились и только значение при t = 0.9 мс изменилось сильно. Аналогичные результаты получены в значениях UС2. В случае изменения параметров элементов цепи снова сильно изменилось UС при t = 0.9 мс.

 Таким образом, можно смело утверждать, что полученная модель достаточно устойчива и что матрица данной системы хорошо обусловлена (cond = 106.008)

# Вывод

В проделанной работе с помощью законов физики было показано, что данная электрическая цепь может быть описана системой из трех уравнений. Использовав знания, ранее полученные при выполнении лабораторных работ по дисциплине «Вычислительная математика», я успешно применил подпрограммы QUANC8, DECOMP, SOLVE, RKF45. В процессе выполнения курсовой работы так же были применены подпрограммы ZEROIN и FMIN. Первая – для вычисления параметра E1 для системы, вторая – для минимизации функции (нахождение минимального значения между вычисленными и экспериментальными).

С высокой точностью (10-7) вычислена ёмкость конденсатора. При оценке полученных значений модели, я убедился, что от экспериментальных они отличаются во 2-3 знаках, что свидетельствует о хорошем построении модели.

При отклонении различных параметров системы, я так же убедился, что модель устойчива к погрешности начальных данных и при вычислении решения системы с изначально искаженными данными отклонение составляет 1-2%

Поставленную задачу считаю полностью выполненной.

# Приложение А. Скриншоты выполнения программы

Начальные параметры:

Изображение выглядит как текст, Шрифт, снимок экрана, Графика

Контент, сгенерированный ИИ, может содержать ошибки.Изображение выглядит как текст, снимок экрана, Шрифт, дизайн

Контент, сгенерированный ИИ, может содержать ошибки.

Изображение выглядит как текст, снимок экрана

Контент, сгенерированный ИИ, может содержать ошибки.Изображение выглядит как текст, снимок экрана

Контент, сгенерированный ИИ, может содержать ошибки.Изображение выглядит как снимок экрана, текст

Контент, сгенерированный ИИ, может содержать ошибки.Вычисление ёмкости конденсатора и получение значений модели

# Приложение Б. Код программы

#include <iostream>

#include "corecrt\_math\_defines.h"

#include "cmathsrc/cmath.h"

#include "rkf45.c"

**using** **namespace** std;

**double** pi = M\_PI;

//Некоторые полезные макросы

#define ABS(x) (((x) >= 0) ? (x) : -(x))

#define MAX(x, y) (((x) >= (y)) ? (x) : (y))

#define MIN(x, y) (((x) < (y)) ? (x) : (y))

#define SIGN0(x) (((x) > 0) ? 1 : -1)

#define SIGN(x) (((x) == 0) ? 0 : SIGN0(x))

#define SIGN2(a, b) (SIGN(b)\*ABS(a))

#define MAXIMUM DBL\_MAX

#define EPSILON DBL\_EPSILON

**double** E1, E2, R, R2, L = **0.0**;

**double** U[] = { -**1**, **7.777**, **12.017**, **10.701**, **5.407**, -**0.843**, -**5.159**, -**6.015**, -**3.668**, **0.283**, **3.829** };

**double** Ui[**11**] = { **0.0** };

**double** C = **0.0**;

**void** **solve**(**unsigned** **int** n, **double**\* A, **double**\* b, **int**\* ipvt)

{

**unsigned** **int** k, i;

**int** m;

**double** t;

**if** (!n) **return**;

//Прямой ход

**if** (n == **1**)

{

b[**0**] /= \*A;

**return**;

}

**for** (k = **0**; k < (n - **1**); k++)

{

m = ipvt[k];

t = b[m];

b[m] = b[k];

b[k] = t;

**for** (i = k + **1**; i < n; i++) b[i] += A[i \* n + k] \* t;

}

//Обратная подстановка

**for** (k = n - **1**; k > **0**; k--)

{

b[k] /= A[k \* (n + **1**)];

t = -b[k];

**for** (i = **0**; i < k; i++) b[i] += A[i \* n + k] \* t;

}

b[**0**] /= \*A;

}

**void** **decomp**(**unsigned** **int** n, **double**\* A, **double**\* cond, **int**\* ipvt)

{

**unsigned** **int** i, j, k;

**int** kp1, m;

**double** ek, t, anorm, ynorm, znorm, \* work;

**if** (!n) **return**;//размерность матрицы не может быть нулевой

ipvt[n - **1**] = **1**;

**if** (n == **1**)//Случай матрицы 1x1

{

\*cond = **1.0**;

**if** (\*A == **0.0**) \*cond = MAXIMUM;//Точная вырожденность

**return**;

}

//Вычислить 1-норму матрицы A

anorm = **0.0**;

**for** (j = **0**; j < n; j++)

{

t = **0.0**;

**for** (i = j; i < n \* n; i += n) t += ABS(A[i]);

**if** (t > anorm) anorm = t;

}

//Гауссово исключение с частичным выбором ведущего элемента

**for** (k = **0**; k < (n - **1**); k++)

{

kp1 = k + **1**;

//Найти ведущий элемент

m = k \* n + k;

**for** (i = (kp1 \* n + k); i < n \* n; i += n) **if** (ABS(A[i]) > ABS(A[m])) m = i;

m /= n;

ipvt[k] = m;

**if** (m != k) ipvt[n - **1**] = -ipvt[n - **1**];

t = A[m \* n + k];

A[m \* n + k] = A[k \* (n + **1**)];

A[k \* (n + **1**)] = t;

//Пропустить этот шаг, если ведущий элемент равен нулю

**if** (t != **0.0**)

{

//Вычислить множители

**for** (i = (kp1 \* n + k); i < n \* n; i += n) A[i] = -A[i] / t;

//Переставлять и исключать по столбцам

**for** (j = kp1; j < n; j++)

{

**int** temp = k - j;

t = A[m \* n + j];

A[m \* n + j] = A[k \* n + j];

A[k \* n + j] = t;

**if** (t != **0.0**) **for** (i = (kp1 \* n + j); i < n \* n; i += n) A[i] += A[i + temp] \* t;

}

}

}

work = **new** **double**[n];

//Решить систему (транспонированная A)\*Y = I

**for** (k = **0**; k < n; k++)

{

t = **0.0**;

**if** (k != **0**) **for** (i = k, j = **0**; j < (k - **1**); i += n, j++) t += A[i] \* work[j];

**if** (t < **0.0**) ek = -**1.0**;

**else** ek = **1.0**;

**double** temp = A[k \* (n + **1**)];

**if** (temp == **0.0**)

{

\*cond = MAXIMUM;

**delete**[] work;

**return**;

}

work[k] = -(ek + t) / temp;

}

k = n - **2**;

**do**

{

t = work[k];

kp1 = k + **1**;

**for** (i = kp1 \* n + k, j = kp1; j < n; i += n, j++) t += A[i] \* work[j];

work[k] = t;

m = ipvt[k];

**if** (m != k)

{

t = work[m];

work[m] = work[k];

work[k] = t;

}

} **while** (k--);

ynorm = **0.0**;

**for** (i = **0**; i < n; i++) ynorm += ABS(work[i]);

//Решить систему A\*Z = Y

solve(n, A, work, ipvt);

znorm = **0.0**;

**for** (i = **0**; i < n; i++) znorm += ABS(work[i]);

//Оценить обусловленность

\*cond = anorm \* znorm / ynorm;

**if** (\*cond < **1.0**) \*cond = **1.0**;

**delete**[] work;

}

**double** **quanc8**(**double**(\*F)(**double**), **double** a, **double** b, **double** ae, **double** re, **double**\* errest, **int**\* nofun, **double**\* flag)

{

**const** **double** w0 = (**double**)**3956** / (**double**)**14175**,

w1 = (**double**)**23552** / (**double**)**14175**,

w2 = (**double**)(-**3712**) / (**double**)**14175**,

w3 = (**double**)**41984** / (**double**)**14175**,

w4 = (**double**)(-**18160**) / (**double**)**14175**;

**double** result = **0.0**, area = **0.0**, x0 = a, f0, stone = (b - a) / (**double**)**16**, step, cor11 = **0.0**, qprev = **0.0**, qnow, qdiff, qleft, esterr, tolerr,

qright[**31**], f[**16**], x[**16**], fsave[**8**][**30**], xsave[**8**][**30**];

//\*\*\* ЭТАП 1 \*\*\* Присвоение начальных значений переменным, не зависящим от интервала. Генерирование констант.

#define LEVMIN 1

#define LEVMAX 30

#define LEVOUT 6

#define NOMAX 5000

**int** levmax = LEVMAX, nofin = NOMAX - **8** \* (LEVMAX - LEVOUT + (**1** << (LEVOUT + **1**))),//Если '\*nofun' достигает значения 'nofin', то тревога

lev = **0**, nim = **1**, i, j;

//Присвоить нулевые значения переменным суммам

\*flag = **0.0**;

\*errest = **0.0**;

\*nofun = **0**;

**if** (a == b) **return** **0.0**;

//\*\*\* ЭТАП 2 \*\*\* Присвоение начальных значений переменным, зависящим от интервала, в соответствии с первым интервалом

x[**15**] = b;

f0 = F(x0);

x[**7**] = (x0 + b) / (**double**)**2**;

x[**3**] = (x0 + x[**7**]) / (**double**)**2**;

x[**11**] = (x[**7**] + b) / (**double**)**2**;

x[**1**] = (x0 + x[**3**]) / (**double**)**2**;

x[**5**] = (x[**3**] + x[**7**]) / (**double**)**2**;

x[**9**] = (x[**7**] + x[**11**]) / (**double**)**2**;

x[**13**] = (x[**11**] + b) / (**double**)**2**;

**for** (j = **1**; j <= **15**; j += **2**) f[j] = F(x[j]);

\*nofun = **9**;

//\*\*\* ЭТАП 3 \*\*\* Основные вычисления

**while** (**1**)

{

x[**0**] = (x0 + x[**1**]) / (**double**)**2**;

f[**0**] = F(x[**0**]);

**for** (j = **2**; j <= **14**; j += **2**)

{

x[j] = (x[j - **1**] + x[j + **1**]) / (**double**)**2**;

f[j] = F(x[j]);

}

nofun += **8**;

step = (x[**15**] - x0) / (**double**)**16**;

qleft = (w0 \* (f0 + f[**7**]) + w1 \* (f[**0**] + f[**6**]) + w2 \* (f[**1**] + f[**5**]) + w3 \* (f[**2**] + f[**4**]) + w4 \* f[**3**]) \* step;

qright[lev] = (w0 \* (f[**7**] + f[**15**]) + w1 \* (f[**8**] + f[**14**]) + w2 \* (f[**9**] + f[**13**]) + w3 \* (f[**10**] + f[**12**]) + w4 \* f[**11**]) \* step;

qnow = qleft + qright[lev];

qdiff = qnow - qprev;

area += qdiff;

//\*\*\* ЭTAП 4 \*\*\* Проверка сходимости для интервала

esterr = ABS(qdiff) / (**double**)**1023**;

tolerr = MAX(ae, re \* ABS(area)) \* (step / stone);

**if** (lev < LEVMIN || ((lev < levmax) && (\*nofun <= nofin) && (esterr > tolerr)))

{

//\*\*\* ЭTAП 5 \*\*\* Сходимости нет. Установить следующий интервал

nim <<= **1**;

lev++;

//Запомнить элементы, относящиеся к правой половине интервала, для будущего использования

**for** (i = **0**; i < **8**; i++)

{

fsave[i][lev - **1**] = f[i + **8**];

xsave[i][lev - **1**] = x[i + **8**];

}

//Собрать элементы, относящиеся к левой половине интервала для немедленного использования

qprev = qleft;

**for** (i = **7**; i >= **0**; i--)

{

f[i \* **2** + **1**] = f[i];

x[i \* **2** + **1**] = x[i];

}

**continue**;

}

**if** (lev >= levmax) \*flag += (**double**)**1.0**;

**else** **if** (\*nofun > nofin)

{

nofin <<= **1**;

levmax = LEVOUT;

\*flag += (b - x0) / (b - a);

//Текущее предельное значение глубины деления пополам равно 'levmax'

}

//\*\*\* ЭTAП 7 \*\*\* Сходимость для интервала имеет место. Прибавить очередные слагаемые к переменным суммам

result += qnow;

\*errest += esterr;

cor11 += qdiff / (**double**)**1023**;

//Установить следующий интервал

**while** (nim & **1**)//если nim - нечётное

{

nim /= **2**;

lev--;

}

nim++;

**if** (lev <= **0**) **break**;

//Собрать элементы, необходимые для следующего интервала

qprev = qright[lev - **1**];

x0 = x[**15**];

f0 = f[**15**];

**for** (i = **0**; i < **8**; i++)

{

f[i \* **2** + **1**] = fsave[i][lev - **1**];

x[i \* **2** + **1**] = xsave[i][lev - **1**];

}

}

//\*\*\* ЭTAП 8 \*\*\* Заключительные операции и выход

result += cor11;

//Обеспечить, чтобы значение переменной '\*errest' было не меньше уровня округлений

**if** (\*errest == (**double**)**0.0**) **return** result;

**double** temp;

**while** (**1**)

{

temp = ABS(result) + \*errest;

**if** (temp != ABS(result)) **return** result;

\*errest \*= (**double**)**2**;

}

}

**double** **zeroin**(**double**(\*F)(**double**), **double** a, **double** b, **double** tol)

{

**double** c, d, e, fa = F(a), fb = F(b), fc, tol1, xm, p, q, r, s;

**while** (**1**)

{

c = a;

fc = fa;

e = d = b - a;

**do**

{

**if** (ABS(fc) < ABS(fb))

{

a = b;

b = c;

c = a;

fa = fb;

fb = fc;

fc = fa;

}

//Проверка сходимости

tol1 = (**double**)**2** \* EPSILON \* ABS(b) + tol / (**double**)**2**;

xm = (c - b) / (**double**)**2**;

**if** (ABS(xm) <= tol1 || fb == **0**) **return** b;

//Необходима ли бисекция

**if** (ABS(e) >= tol1 && ABS(fa) > ABS(fb))

{

//Возможна ли квадратичная интерполяция

**if** (a != c)

{

//Обратная квадратичная интерполяция

q = fa / fc;

r = fb / fc;

s = fb / fa;

p = s \* ((**double**)**2** \* xm \* q \* (q - r) - (b - a) \* (r - (**double**)**1**));

q = (q - (**double**)**1**) \* (r - (**double**)**1**) \* (s - (**double**)**1**);

}

**else**

{

//Линейная интерполяция

s = fb / fa;

p = (**double**)**2** \* xm \* s;

q = (**double**)**1** - s;

}

//Выбрать знаки

**if** (p > **0**) q = -q;

**else** p = -p;//p = ABS(p)

//Приемлема ли интерполяция

**if** (((**double**)**2** \* p) >= ((**double**)**3** \* xm \* q - ABS(tol1 \* q)) || p >= ABS(e \* q / (**double**)**2**)) e = d = xm;//Бисекция

**else**

{

e = d;

d = p / q;

}

}

**else** e = d = xm;//Бисекция

//Завершить шаг

a = b;

fa = fb;

**if** (ABS(d) > tol1) b += d;

**else** b += SIGN2(tol1, xm);

fb = F(b);

} **while** (fb \* SIGN(fc) <= **0**);

}

}

**double** **FMin**(**double**(\*F)(**double**), **double** a, **double** b, **double** tol)

{

**double** c = (**double**)((**double**)**3.0** - (**double**)sqrt(**5.0**)) / (**double**)**2**,//с - это возведённая в квадрат величина, обратная к золотому сечению

d, e = (**double**)**0**, eps = (**double**)sqrt(EPSILON),//eps приблизительно равно квадратному корню из относительной машинной точности

xm, p, q, r, tol1, tol2, u, v = a + c \* (b - a), w = v, x = v, fx = F(v), fu, fv = fx, fw = fx;

**while** (**1**)

{

xm = (a + b) / (**double**)**2**;

tol1 = eps \* ABS(x) + tol / (**double**)**3**;

tol2 = tol1 \* (**double**)**2**;

//Проверить критерий окончания

**if** (ABS(x - xm) <= (tol2 - (b - a) / (**double**)**2**)) **return** x;

**if** (ABS(e) > tol1)//Если золотое сечение не требуется

{

//Построить параболу

r = (x - w) \* (fx - fv);

q = (x - v) \* (fx - fw);

p = (x - v) \* q - (x - w) \* r;

q = (**double**)**2** \* (q - r);

**if** (q > **0**) p = -p;

**else** q = -q;//q = ABS(q)

r = e;

e = d;

//Приемлема ли парабола

**if** (ABS(p) >= ABS(q \* r / (**double**)**2**) || p <= q \* (a - x) || p >= q \* (b - x))

{

//Шаг золотого сечения

**if** (x >= xm) e = a - x;

**else** e = b - x;

d = c \* e;

}

**else**

{

//Шаг параболической интерполяции

d = p / q;

u = x + d;

//F не следует вычислять слишком близко к 'a' или 'b'

**if** ((u - a) < tol2 || (b - u) < tol2) d = SIGN2(tol1, xm - x);

}

}

**else**

{

//Шаг золотого сечения

**if** (x >= xm) e = a - x;

**else** e = b - x;

d = c \* e;

}

//F не следует вычислять слишком близко к 'x'

**if** (ABS(d) >= tol1) u = x + d;

**else** u = x + SIGN2(tol1, d);

fu = F(u);

//Присвоить новые значения параметрам 'a', 'b', 'v', 'w' и 'x'

**if** (fu <= fx)

{

**if** (u >= x) a = x;

**else** b = x;

v = w;

fv = fw;

w = x;

fw = fx;

x = u;

fx = fu;

**continue**;

}

**if** (u < x) a = u;

**else** b = u;

**if** (fu <= fw || w == x)

{

v = w;

fv = fw;

w = u;

fw = fu;

}

**else** **if** (fu <= fv || v == x || v == w)

{

v = u;

fv = fu;

}

}

**return** x;

}

**void** **recombination** (**double**\*\* A, **double**\* a, **int** n) {

**for** (**int** i = **0**; i < n; i++) {

**for** (**int** j = **0**; j < n; j++) {

a[i \* n + j] = A[i][j];

}

}

}

**double** **l**(**double** x) {

**return** cos((pi \* pow(x, **2**)) / **2**);

}

**double** **z**(**double** x) {

**return** x \* exp(x) - **2**;

}

**int** **f**(**int** n, **double** t, **double** x[], **double** dxdt[]) {

dxdt[**0**] = **1** / L \* (E1 - E2 - x[**2**] + x[**1**] \* R2 - x[**0**] \* (R + R2));

dxdt[**1**] = **1** / L \* (E2 + x[**2**] + x[**0**] \* R2 - x[**1**] \* (R2 + R));

dxdt[**2**] = **1** / C \* (x[**0**] - x[**1**]);

**return** **0**;

}

**void** **rkf45Driver**() {

**double** hprint = **1e-4**;

**int** n = **3**, nfe, fail, stepCount,

flag = **1**,

maxfe = **5000**;

**double** relerr = **1.0e-11**,

abserr = relerr,

t1, t2, x[**3**], xp[**3**], h;

rkfinit(**3**, &fail);

**if** (fail == **0**) {

x[**0**] = E1 / R;

x[**1**] = **0**;

x[**2**] = U[**0**];

**for** (**int** i = **1**; i < **11**; i++) {

t2 = hprint \* i; //TOUT

t1 = t2 - hprint; //T

rkf45(f, **3**, x, xp, &t1, t2, &relerr, abserr, &h, &nfe, maxfe, &flag);

**if** (flag != **2**)

cout << "**\t**flag ! " << flag;

Ui[i] = x[**2**];

}

rkfend();

}

}

**double** **F**(**double** c) {

**static** **int** iter = **1**;

C = c;

cout << "**\t**C " << C << endl;

rkf45Driver();

cout << endl;

**for** (**int** i = **0**; i < **11**; i++) {

cout << "**\t**Ui " << Ui[i] << endl;

}

cout << endl;

**double** sum = **0**;

**for** (**int** i = **0**; i < **11**; i++) {

sum += pow((U[i] - Ui[i]), **2**);

}

cout << "**\t**sum " << sum << " iter " << iter << endl;

iter++;

**return** sum;

}

**void** **main**() {

//DECOMP & SOLVE

**double**\*\* A = **new** **double**\* [**3**];

**for** (**int** i = **0**; i < **3**; i++) {

A[i] = **new** **double**[**3**];

}

A[**0**][**0**] = **16**, A[**0**][**1**] = -**24**, A[**0**][**2**] = **18**;

A[**1**][**0**] = -**24**, A[**1**][**1**] = **46**, A[**1**][**2**] = -**42**;

A[**2**][**0**] = **18**, A[**2**][**1**] = -**42**, A[**2**][**2**] = **49**;

**int**\* pivot = **new** **int**[**3**];

**for** (**int** i = **0**; i < **3**; i++)

pivot[i] = **0**;

**double** flag = **0**;

**double** cond = **0**;

**double**\* a = **new** **double**[**9**];

**for** (**int** i = **0**; i < **9**; i++)

a[i] = **0**;

recombination (A, a, **3**);

decomp(**3**, a, &cond, pivot);

cout << "**\t**decomp called." << endl << "**\t**cond=" << cond << endl << endl;

**double**\* b = **new** **double**[**3**];

b[**0**] = **178**, b[**1**] = -**82**, b[**2**] = -**71**;

solve(**3**, a, b, pivot);

R = b[**0**], R2 = b[**1**], E2 = b[**2**];

cout << "**\t**solve called." << endl << "**\t**R=" << b[**0**] << endl;

cout << "**\t**R2=" << b[**1**] << endl;

cout << "**\t**E2=" << b[**2**] << endl << endl;

//QUANC8

**int** nofunR = **0**;

**double** errestR = **0**, posnR = **0**;

**double** abserr = **1e-12**;

**double** relerr = **1e-12**;

L = quanc8(l, **0**, **0.2**, abserr, relerr, &errestR, &nofunR, &flag);

L \*= **0.2000789**;

cout << "**\t**quanc8 called." << endl << "**\t**L=" << L << " flag=" << flag << endl << endl;

//ZEROIN

**double** x1, x2, toler;

x1 = **0**;

x2 = **1**;

toler = **1.0e-10**;

E1 = zeroin(z, x1, x2, toler);

E1 \*= **4.691511**;

cout << "**\t**zeroin called." << endl << "**\t**E1=" << E1 << endl;

**double** cMin = FMin(F, **5e-7**, **2e-6**, **1e-12**);

cout << "**\n\n\t**cMin = " << cMin << endl;

}