Содержание

Подп. и дата
Инв. № дубл.
Взам. инв. №
Подп. и дата
Инв. № подл.

Изм.	Лист	№ докум.	Подп.	Дата	Вариант №3		
Разраб.		Белым А.А.			Пояснительная записка к лабораторной работе по курсу «Вычислительный практикум» по теме «Интегральные уравнения Фредгольма второго рода»	Лит.	Лист: 2 / Листов: 21
Пров.		Ермаков А.С.					
Н. контр.						ТулГУ гр. 220601	
Утв.							

					Вариант №3	Лист
Изм.	Лист	№ докум.	Подп.	Дата		3

Инв. № подл. | Подп. и дата | Взам. инв. № | Инв. № дубл. | Подп. и дата

ВВЕДЕНИЕ

Интегральными уравнениями называются функциональные уравнения, содержащие интегральные преобразования над неизвестной функцией y(x).

Известны несколько видов интегральных уравнений (уравнение Вольтера, уравнение Фредгольма). Решение их может возникнуть в математическом анализе.

В данной работе разбирается неоднородное интегральное уравнение Фредгольма второго рода. Рассматривается вид этого уравнения и способ его решения.

Отчёт также содержит полный текст программы на языках C и Python, описание всех функций, инструкцию пользователю и тестовый пример.

1. РЕШЕНИЕ ИНТЕГРАЛЬНЫХ УРАВНЕНИЙ ФРЕДГОЛЬМА ВТОРОГО РОДА

1.1. Содержательное описание задачи

Неоднородное уравнение Фредгольма второго рода выглядит так:

$$f(x) = \phi(x) - \lambda \int_a^b K(x,s)\phi(s)ds$$

Задача состоит в том, чтобы имея ядро $K(x,s)$ и функцию $f(x)$, найти функцию $\phi(x)$. То есть, задана некая функция $f(x)$, задано ядро, представляющее собой функцию от двух переменных $K(x,s)$. Также заданы пределы интегрирования a и b, коэффициент λ и шаг интегрирования. Требуется найти табличные значения функции $\phi(x)$, где соответствующим значениям x изменяющимся от a до b с заданным шагом сопоставлены соответствующие значения функции.

1.2. Формальная постановка задачи

Неоднородное уравнение Фредгольма второго рода выглядит так:

$$f(x) = \phi(x) - \lambda \int_a^b K(x,s)\phi(s)ds$$

Пусть функции $f(x)$ и $K(x,s)$ задаются в явном виде. Значения переменных a,b, λ и шаг интегрирования должны быть числами, причём шаг интегрирования должен быть больше 0, $a <= b$.

Ответ представить в виде таблице с тремя столбцами: в первом находится аргумент искомой функции изменяющимся от a до b с заданным шагом, а во втором столбце соответствующие аргументу значения функции, в третьем столбце представленно значение аналитического решения. Для наглядности должен быть представлен график искомой функции.

Для решения задачи требуется знать описанные ниже теоретические сведения. Интегральными уравнениями называются функциональные уравнения, содержащие интегральные преобразования над неизвестной функцией $y(x)$. Интегральное уравнение называется однородным, если $ay(x)$ есть решение уравнения для произвольного a. Линейное интегральное уравнение в общем виде может быть представлено:

$$g(x)y(x) - a\int_v k(x,s)y(s)ds = f(x)$$

где $k(x,s)$ - ядро интегрального преобразования, правая часть $f(x)$ и $g(x)$ являются заданными функциями, a - параметр уравнения. Область интегрирования V может быть фиксированной (интегральные уравнения типа фредгольмовых) или переменной (интегральные уравнения типа вольтерровых). Линейное интегральное уравнение первого рода получается при $g(x) = 0$, $a = -1$ и имеет вид:

$$\int_v k(x,s)y(s)ds = f(x)$$

Однородное линейное интегральное уравнение второго рода получается при $f(x) = 0$,$g(x) = 1$ и имеет вид:

$$y(x) - \int_v k(x,s)y(s)ds = 0$$

Неоднородное интегральное уравнение второго рода получается при $g(x) = 1$ и имеет вид:

$$y(x) - \int_v k(x,s)y(s)ds = f(x)$$

Уравнения вида

$$y(x) - \int_v k(x,s)F(y(s))ds = f(x)$$

являются неоднородными.

Линейное интегральное неоднородное уравнение Фредгольма второго рода имеет вид:

$$y(x) - \lambda \int_a^b k(x,s)F(y(s))ds = f(x)$$

где ядро определено в квадрате $V = [a,b] * [a,b]$. Кроме того, полагается, что ядро непрерывно в V. При $\lambda = 1$, используя квадратурную формулу трапеций с постоянным шагом h, получим:

$$y_i - h\sum_{j=1}^{n} A_j k_{ij} y_j = f_i$$

где $n = (b-a)/h + 1$-целое, $A_j = 1$ при j не равном 1 или n и $A_j = 0.5$ при $j = 1$ или n.

2. РАЗРАБОТКА АЛГОРИТМА

2.1. Разработка графического интерфейса пользователя

Для решения задачи требуются иметь следующие исходные данные: начало интегрирования (a), конец интегрирования (b), шаг интегрирования. Для ввода этих значений необходимо предусмотреть отдельные поля. Функции f(x) и K(x,s) будут заданы программно, поэтому нужно обеспечить вывод этих функций на экран. Кроме того, для наглядности и проверки правильности работы программы будет использоваться уравнение с уже известным аналитическим решением, и эта функция-решение также должна быть выведена на экран.

Известно, что результатом вычислений должна быть функция, следовательно, необходима таблица для вывода аргументов и значений полученной функции, и значений аналитического решения. Также решения (дискретное и аналитическое) будут визуализироваться с помощью программы Gnuplot, которая запускается в отдельном окне по нажатию специальной кнопки.

В панели меню нужно предусмотреть следующие пункты: выход из программы, решение уравнения, вывод графиков, справка по программе Gnuplot и справка по данной программе.

Итак, внешний вид разработанного интерфейса представлен на рисунке 1.

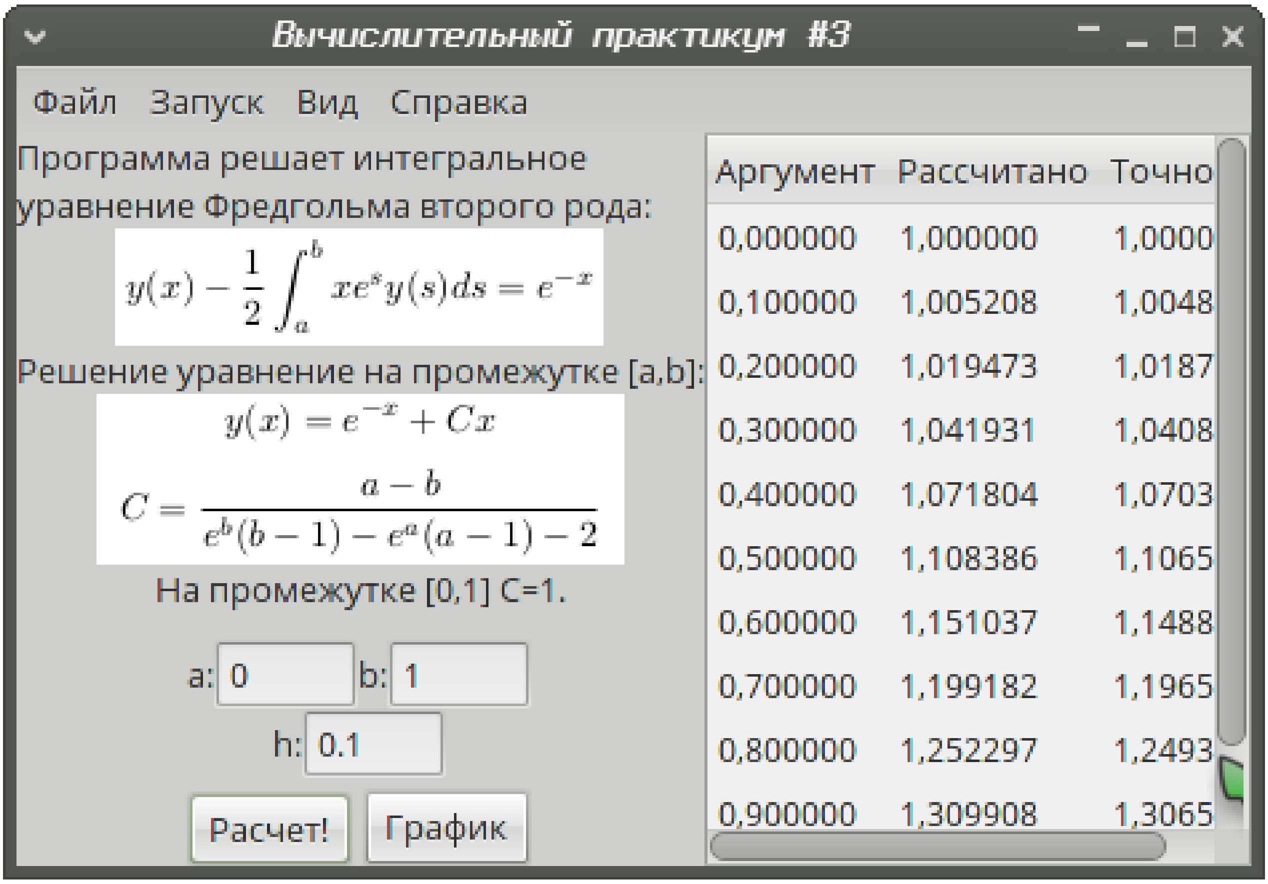

Рисунок 1 — Разработанный интерфейс программы

2.2. Разработка структур данных

Для описания исходного уравнения Фредгольма будем использовать следующие переменные

a – Начало интегрирования.

b – Конец интегрирования.

h – шаг для интегрирования.

F(x) – табличное представление функции f(x)

K(x,s) – табличное представления ядра K(x,s)

2.3. Разработка структуры алгоритма

Основную программу можно разбить на три участка: считывание значений, нахождения табличных значений искомой функции и вывод полученных результатов.

1) Для нахождения табличных значений искомой функции создадим функцию fredholm, параметры решения уравнения (см. предыдущий разел). Возвращать данная функция будет массив, представляющий собой значения искомой функции. Так как для решения уравнения Фредгольма требуется решить систему линейных алгебраических уравнений, следует создать отдельную подпрограмму для нахождения корней СЛАУ. В данной работе будет использоваться метод Гаусса для решения СЛАУ.

2) Подпрограмма ввода данных input_data.

3) Подпрограмма вывода данных output_data.

2.4. Схема алгоритма

На рисунке 2 представлена схема алгоритма создания системы алгебраических уравнений для решения интегрального уравнения Фредгольма.

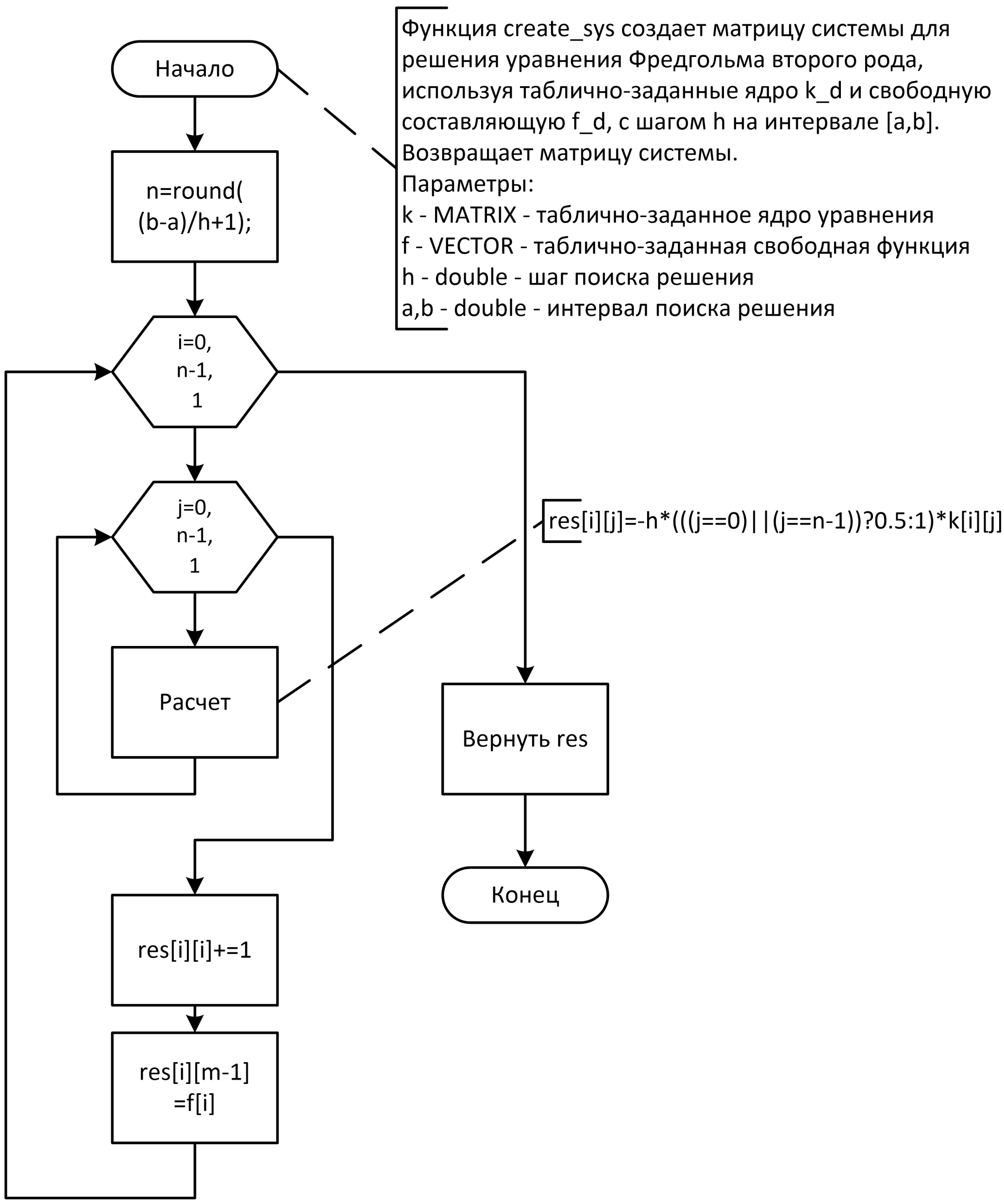

Рисунок 2 – Схема алгоритма создания системы уравнений

На рисунке 3 представлена схема алгоритма решения интегрального уравнения Фредгольма.

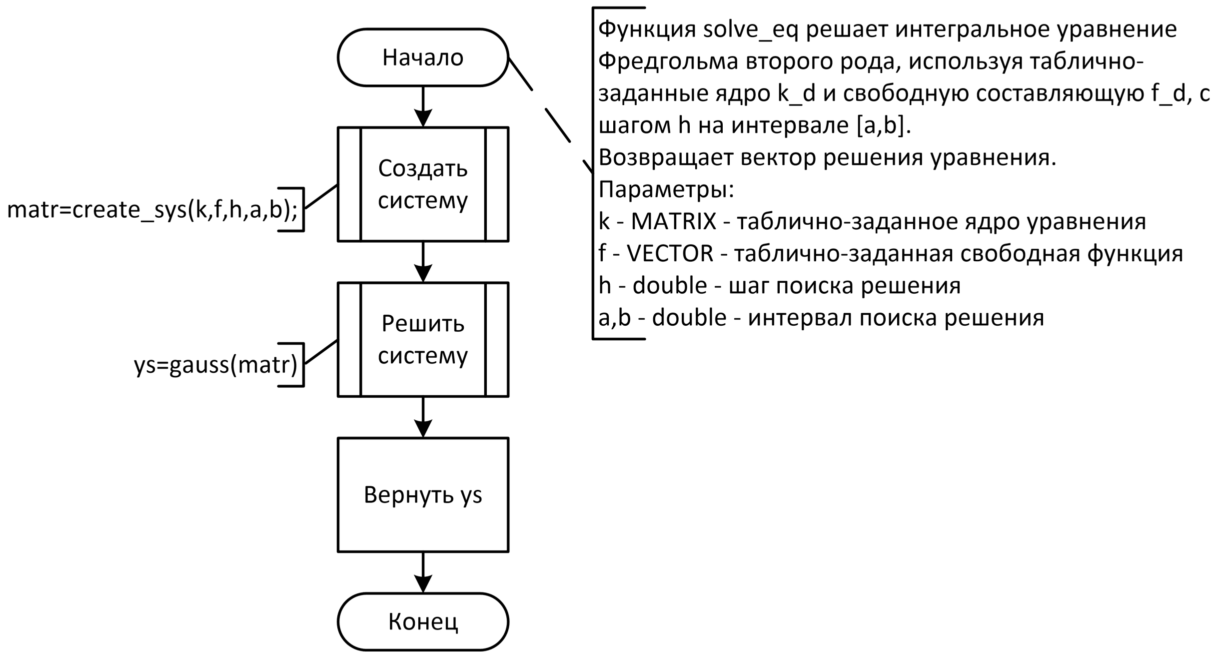

Рисунок 3 — Схема алгоритма решения уравнения Фредгольма

3. РАЗРАБОТКА ПРОГРАММЫ

3.1. Описание переменных и структур данных

В данной программе используются следующие переменные:

a – double - Начало интегрирования.

b – double - Конец интегрирования.

h – double - шаг для интегрирования.

F(x) – VECTOR - табличное представление функции f(x)

K(x,s) – MATRIX - табличное представления ядра K(x,s)

Все массивы определяются типом VECTOR, который содержит в себе:

n - int - количество элементов,

elements - массив из float - элементы массива.

Матрицы стандартного вида задаются типом MATRIX:

n,m - int - количество строк и столбцов,

elements - массив из float - элементы матрицы.

3.2. Описание функций

1. MATRIX create_sys(MATRIX k,VECTOR f,double h,double a,double b)

Функция create_sys создает матрицу системы для решения уравнения Фредгольма второго рода, используя таблично-заданные ядро k_d и свободную составляющую f_d, с шагом h на интервале [a,b].

Возвращает матрицу системы.

Параметры функции представлены в таблице 1 :

Таблица 1 – Параметры функции создания матрицы системы

имя	**тип**	**предназначение**
k	MATRIX	таблично-заданное ядро уравнения
f	VECTOR	таблично-заданная свободная функция
h	double	шаг поиска решения
a,b	double	интервал поиска решения

2. VECTOR solve_eq(MATRIX k,VECTOR f,double h, double a,double b)

Функция solve_eq решает интегральное уравнение Фредгольма второго рода, используя таблично-заданные ядро k_d и свободную составляющую f_d, с шагом h на интервале [a,b].

Возвращает вектор решения уравнения.

Параметры функции представлены в таблице 2 :

Таблица 2 – Параметры функции решения уравнения

имя	**тип**	**предназначение**
k	MATRIX	таблично-заданное ядро уравнения
f	VECTOR	таблично-заданная свободная функция
h	double	шаг поиска решения
a,b	double	интервал поиска решения

3. VECTOR gauss(MATRIX matr)

Функция решения СЛАУ методом Гаусса.

4. input_data(self)

Подпрограмма ввода исходных данных.

5. output_data(self,a,b,x,y)

Подпрограмма вывода результатов.

6. on_run_click(self,button,data=None)

Подпрограмма считывания данных, решения уравнения и вывода результатов.

7. show_chart(self):

Подпрограмма построения графика.

4. ИНСТРУКЦИЯ ПОЛЬЗОВАТЕЛЮ

Данная программа решает интегральное уравнение Фредгольма второго рода вида:

$$y(x) - \frac{1}{2}\int_a^b xe^s y(s)ds = e^{-x}$$

.

Данная программа не требует установки. Для её запуска необходимо открыть файл prac3.py. Внимание: для работы приложения на компьютере должен быть установлен Python 3, GTK+3, GObject-introspection и Gnuplot

Для работы необходимы следующие данные:

. Начало промежутка интегрирования

Конец промежутка интегрирования

Шаг интегрирования

После ввода значений для получения результата требуется нажать кнопку "Решить"либо открыть пункт меню Запуск->Решить уравнение. После этого на экран будут выведены табличные значения искомой функции и построен соответствующий график с помощью программы Gnuplot.

5. ТЕСТОВАЯ ЗАДАЧА

5.1. Аналитическое решение и умозрительные результаты

Данная программа решает интегральное уравнение Фредгольма второго рода вида:

$$y(x) - \frac{1}{2}\int_a^b xe^s y(s)ds = e^{-x}$$

.

Пусть дан участок интегрирования от 0 до 1.

Тогда аналитическое решение уравнение:

$$y(x) = e^{-x} + x$$

5.2. Решение, полученное с использованием разработанного ПО

Ниже на рисунке 4 представлен пример работы программы решения интегрального уравнения Фредгольма второго рода.

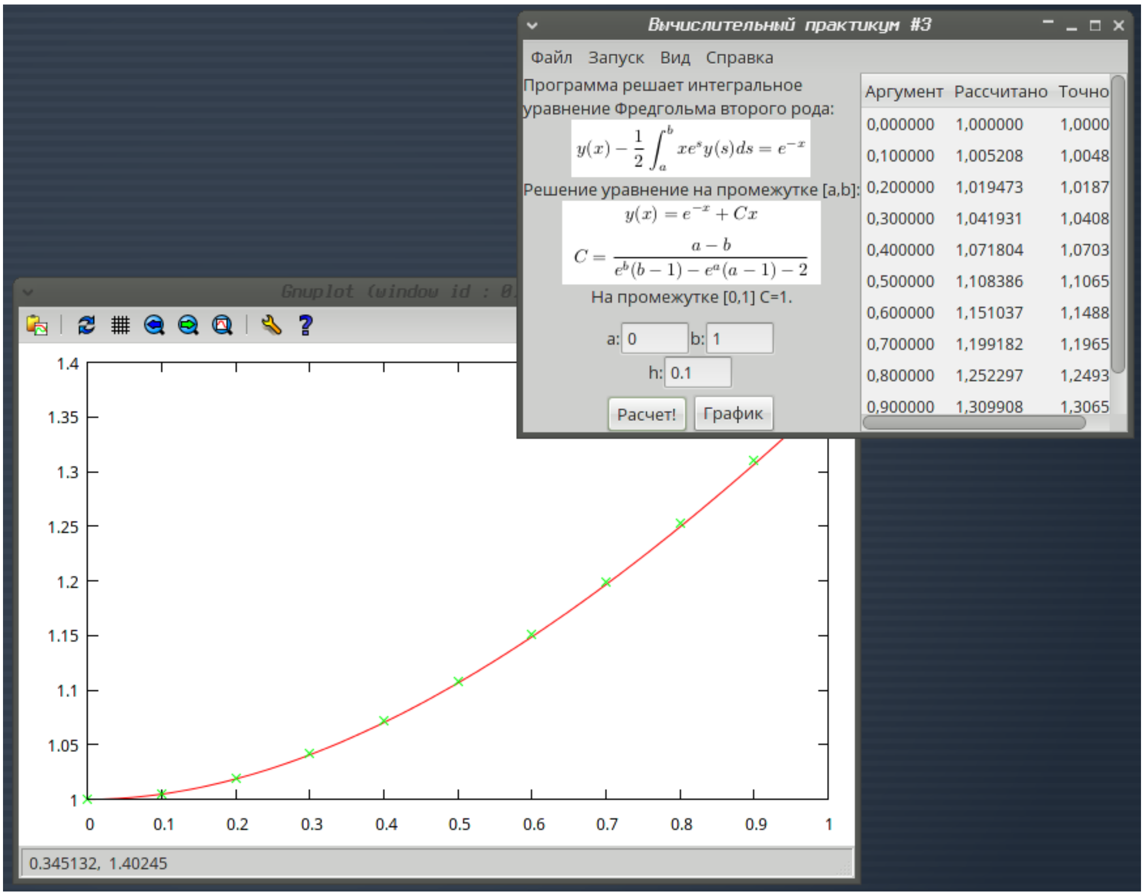

Рисунок 4 — Пример работы программы решения уравнения Фредгольма

5.3. Выводы

Данная программа решает интегральное уравнение Фредгольма второго рода. Ввод функции f(x) и K(x,s) осуществляется в явном виде. Так как решением уравнения является функция, то программы выводит табличные значения искомой функции и строит по ним график.

ЗАКЛЮЧЕНИЕ

Решение интегральных уравнений, в частности уравнений Фредгольма второго рода, представляет собой довольно сложную алгебраическую задачу. С помощью рассмотренного дискретного метода можно получить решение данного типа уравнений с любой требуемой точностью.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. http://python.org
2. http://www.gtk.org
3. http://ru.wikipedia.org
4. http://en,wikipedia.org

ПРИЛОЖЕНИЕ

Ниже приведен текст модуля расширения Python, реализующего решение интегрального уравнения Фредгольма второго рода и написанного на Си.

```
#include <Python.h>

typedef struct{
    double *elements;
    int n,m;
} MATRIX;

typedef struct{
    double *elements;
    int n;
} VECTOR;

typedef struct{
    int i,j;
} COORD;
VECTOR gauss_direct(MATRIX a){
    int i,j,k;
    VECTOR index;
    index.elements=malloc(a.n*sizeof(double));
    index.n=a.n;
    for (i=0;i<index.n;++i)
        index.elements[i]=i;

    COORD index_of_max(void){
```

```
        COORD max;
        int j,k;
        double m=abs(a.elements[i*a.m+i]);
        max.i=i,max.j=i;
        for (j=i;j<a.n;++j){
            for (k=i;k<a.n;++k){
                if (abs(a.elements[j*a.m+k])>m){
                    m=abs(a.elements[j*a.m+k]);max.i=j;max.j=k;
                }
            }
        }
        return max;
    }
    for (i=0;i<a.n;++i){
        COORD max=index_of_max();
        if (a.elements[max.i*a.m+max.j]==0){
            return index;
        }
        if (max.i!=i){
            double temp;
            for (j=i;j<a.m;++j){
                temp=a.elements[i*a.m+j];
                a.elements[i*a.m+j]=a.elements[max.i*a.m+j];
                a.elements[max.i*a.m+j]=temp;
            }
        }
        if (max.j!=i){
            int temp_i=index.elements[max.j];
            index.elements[max.j]=index.elements[i];
            index.elements[i]=temp_i;
            double temp;
            for (j=0;j<a.n;++j){
                temp=a.elements[j*a.m+i];
                a.elements[j*a.m+i]=a.elements[j*a.m+max.j];
                a.elements[j*a.m+max.j]=temp;
            }
        }
        for (j=i+1;j<a.n;++j){
            if (!a.elements[i*a.m+i])
                printf("Error!\n");
            double factor=a.elements[j*a.m+i]/a.elements[i*a.m+i];
            a.elements[j*a.m+i]=0;
            int k;
            for (k=i+1;k<a.m;++k){
                a.elements[j*a.m+k]-=a.elements[i*a.m+k]*factor;
                /*if (abs(a.elements[j*a.m+k])<1e-10)
                    a.elements[j*a.m+k]=0;*/
            }
        }

    }
    return index;
}

VECTOR gauss_reverse(MATRIX a){
    int i,j;
    VECTOR res;
    res.elements=malloc(a.n*sizeof(double));
    res.n=a.n;
```

```c
    for(i=0;i<a.n;i++)
        res.elements[i]=a.elements[i*a.m+a.m-1];

    for (i=a.n-1;i>=0;--i){
        for (j=a.n-1;j>i;--j)
            res.elements[i]-=a.elements[i*a.m+j]*res.elements[j];
        res.elements[i]/=a.elements[i*a.m+i];
    }
    return res;
}

void return_order(VECTOR *values,VECTOR index){
    int i;
    double *res;
    res=malloc(values->n*sizeof(double));
    for (i=0;i<values->n;++i)
        res[(int)index.elements[i]]=values->elements[i];
    free(values->elements);
    values->elements=res;
}
/*
Функция create_sys создает матрицу системы для решения уравнения Фредгольма
второго рода,  используя таблично-заданные
ядро k_d и свободную составляющую f_d, с шагом h на интервале [a,b].
Возвращает матрицу системы.
Параметры:
k - MATRIX - таблично-заданное ядро уравнения
f - VECTOR - таблично-заданная свободная функция
h - double - шаг поиска решения
a,b - double - интервал поиска решения
*/
MATRIX create_sys(MATRIX k,VECTOR f,double h,double a,double b){
    int i,j;
    MATRIX res;
    int n=round((b-a)/h+1);
    res.n=n;res.m=n+1;
    res.elements=malloc(res.n*res.m*sizeof(double));
    if (res.elements==NULL)
        printf("Error!\n");
    for (i=0;i<n;i++){
        for (j=0;j<n;++j)
            res.elements[i*res.m+j]=-h*(((j==0)||(j==n-1))?0.5:1)*
                    k.elements[i*k.m+j];
        res.elements[i*res.m+i]+=1;

        res.elements[i*res.m+res.m-1]=f.elements[i];
    }
    return res;
}
VECTOR gauss(MATRIX matr){
    VECTOR index=gauss_direct(matr);
    VECTOR ys=gauss_reverse(matr);
    return_order(&ys,index);
    return ys;

}

/*
Функция solve_eq решает интегральное уравнение Фредгольма второго рода,
```

```
используя таблично-заданные
ядро k_d и свободную составляющую f_d, с шагом h на интервале [a,b].
Возвращает вектор решения уравнения.
Параметры:
k - MATRIX - таблично-заданное ядро уравнения
f - VECTOR - таблично-заданная свободная функция
h - double - шаг поиска решения
a,b - double - интервал поиска решения
*/
VECTOR solve_eq(MATRIX k,VECTOR f,double h, double a,double b){
    MATRIX matr=create_sys(k,f,h,a,b);
    return gauss(matr);
}
PyObject* solve_eq_py(PyObject* self,PyObject *args){
    int i,j;
    double a,b,h;
    PyObject *k,*f;
    PyArg_ParseTuple(args,"dddOO",&a,&b,&h,&k,&f);
    MATRIX k_d;
    VECTOR f_d;
    int n=round((b-a)/h+1);
    if (n<=0)
        return NULL;
    VECTOR xs;
    xs.n=n;
    xs.elements=malloc(n*sizeof(double));
    xs.elements[0]=a;
    for (i=1;i<n;i++)
        xs.elements[i]=xs.elements[i-1]+h;
    f_d.n=n;k_d.n=n;k_d.m=n;
    f_d.elements=malloc(f_d.n*sizeof(double));
    k_d.elements=malloc(k_d.n*k_d.m*sizeof(double));
    PyObject* temp;
    for (i=0;i<n;++i){
        for (j=0;j<n;++j){
            PyObject* arg=Py_BuildValue("(dd)",xs.elements[i],xs.elements[j]);
            temp=PyObject_CallObject(k,arg);
            Py_XDECREF(arg);
            k_d.elements[i*k_d.m+j]=PyFloat_AsDouble(temp);
            Py_XDECREF(temp);
        }
        PyObject *arg=Py_BuildValue("(d)",xs.elements[i]);
        temp=PyObject_CallObject(f,arg);
        Py_XDECREF(arg);
        f_d.elements[i]=PyFloat_AsDouble(temp);
        Py_XDECREF(temp);
    }
    VECTOR ys=solve_eq(k_d,f_d,h,a,b);
    PyObject *py_xs=PyList_New(n),
             *py_ys=PyList_New(n);
    for (i=0;i<n;++i){
        PyList_SetItem(py_xs,i,PyFloat_FromDouble(xs.elements[i]));
        PyList_SetItem(py_ys,i,PyFloat_FromDouble(ys.elements[i]));
    }

    return Py_BuildValue("OO",py_xs,py_ys);
}

static PyMethodDef alg3methods[] = {
```

```
    {"solve_eq",  solve_eq_py, METH_VARARGS,""},
    {NULL, NULL, 0, NULL}        /* Sentinel */
};
static struct PyModuleDef alg3module = {
   PyModuleDef_HEAD_INIT,
   "alg3",   /* name of module */
   NULL, /* module documentation, may be NULL */
   -1,       /* size of per-interpreter state of the module,
                or -1 if the module keeps state in global variables. */
   alg3methods
};

PyMODINIT_FUNC
PyInit_alg3(void)
{
    return PyModule_Create(&alg3module);
}
```

Далее приводится текст основной программы, написанной на Python 3.

```
#!/usr/bin/env python
from gi.repository import Gtk
from math import exp
from alg3 import solve_eq
import subprocess as subp
from tempfile import NamedTemporaryFile
from os import remove
from sys import stderr
def c(a,b):
    return (a-b)/(exp(b)*(b-1)-exp(a)*(a-1)-2)
def y_solved(x,a,b):
    return exp(-x)+x*c(a,b)
def k(x,s):
    return 1/2*x*exp(s);
def f(x):
    return exp(-x);
class Application(Gtk.Builder):
    def __init__(self,ui_filename):
        Gtk.Builder.__init__(self)
        self.add_from_file(ui_filename)
        self.connect_signals(self)
        self.plot=None
        self.tempfile=NamedTemporaryFile(delete=False)
        self.tempfile.close()
        #print(help(NamedTemporaryFile))
    def show(self,form_name):
        window = self.get_object(form_name)
        window.show()
        Gtk.main()
    def on_window_destroy( self,widget, data=None):
        self.get_object('window1').hide()
        if self.plot and self.plot.poll()==None:
            self.plot.terminate()
            if self.plot.poll()==None:
                self.plot.kill()
                self.plot.wait()
        if self.tempfile:
            if not self.tempfile.closed:
                self.tempfile.close()
```

```
            remove(self.tempfile.name)
        Gtk.main_quit()
    def show_chart(self):
        a=float(self.get_object("entry1").get_text())
        b=float(self.get_object("entry2").get_text())
        h=float(self.get_object("entry3").get_text())
        table=self.get_object("treeview1").get_model()
        expr='exp(-x)+{0}*x'.format(c(a,b))
        print(expr)
        if self.plot and self.plot.poll()==None:
            self.plot.terminate()
            if self.plot.poll()==None:
                self.plot.kill()
                self.plot.wait()
        self.tempfile=open(self.tempfile.name,'wb')
        for i in table:
            self.tempfile.write("{0} {1}\n".format(i[0],i[1]).encode())
        self.tempfile.close()
        msg='plot '+expr+',"'+self.tempfile.name+'";pause -1'
        self.plot=subp.Popen(['gnuplot','-e',msg],shell=False,stdin=subp.PIPE)
    def chart_clicked(self,button,data=None):
        self.show_chart()
    def input_data(self):
        a=float(self.get_object("entry1").get_text())
        b=float(self.get_object("entry2").get_text())
        h=float(self.get_object("entry3").get_text())
        return a,b,h
    def output_data(self,a,b,x,y):
        table=self.get_object("treeview1")
        model=table.get_model()
        table.set_model(None)
        model.clear()
        for i in range(0,len(x)):
            model.append([x[i],y[i],y_solved(x[i],a,b)])
        table.set_model(model)

    def on_run_click(self,button,data=None):
        a,b,h=self.input_data()
        x,y=solve_eq(a,b,h,k,f)
        self.output_data(a,b,x,y)
        self.show_chart()

app=Application("prac3.ui")
app.show("window1")
```