БГУИР

Кафедра информатики

ОТЧЕТ

по лабораторной работе № 1

Решение краевых задач.

Методы коллокаций, наименьших квадратов и Галеркина

Выполнил: Проверил:

ст. гр. 553504 Гербик А. И.

Криницин А. В.

Минск 2017

ОГЛАВЛЕНИЕ

[1. Введение 3](#_Toc465628474)

[1.1. Постановка задачи 3](#_Toc465628475)

[1.2. Краткие теоретические сведения 3](#_Toc465628476)

[1.3 Способы решения краевой задачи 4](#_Toc465628480)

[2. Метод коллокаций 4](#_Toc465628481)

[3. Метод наименьших квадратов 5](#_Toc465628482)

[4. Метод Галеркина 6](#_Toc465628483)

[5. Исходный код программы 6](#_Toc465628484)

[6. Анализ полученных результатов 8](#_Toc465628485)

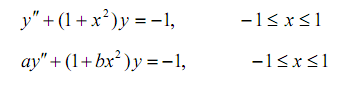
[6.1 Результат работы программы 8](#_Toc465628489)

[6.2 Вывод 8](#_Toc465628490)

# **1.Введение**

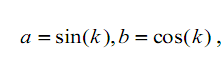
## **Постановка задачи**

Методами коллокаций, интегральным и дискретным методами наименьших квадратов и Галеркина получить численное решение краевой задачи:

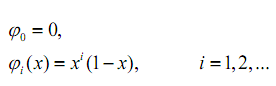


Вариант 13

Исходные данные:

k=13

Базисную систему выбрать в виде:



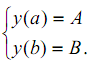
## **Краткие теоретические сведения**

Будем рассматривать дифференциальное уравнение второго порядка.

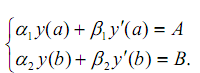
,

где , , − заданные непрерывные на отрезке *[a, b]* функции.

Краевой задачей называется задача нахождения решения, удовлетворяющего граничным условиям:



Часто вместо граничных условий используют обобщенные граничные условия:



Граничные условия называются однородными, если A = B = 0. Соответственно, краевая задача называется однородной, если у нее однородные граничные условия и правая часть уравнения.

**Теорема.** Краевая задача имеет решение, причем единственное тогда и только тогда, когда соответствующая ей однородная краевая имеет только нулевое решение (тривиальное решение однородной краевой задачи).



## **Способы решения краевой задачи**

Поскольку достаточно хороших аналитических методов нет, то используются приближенные методы.

Система дважды непрерывно дифференцируемых функций называется *базисной системой*, если выполняется:

1) удовлетворяет граничному условию,

2) функции − линейно независимы на *[a, b]* и удовлетворяют однородным граничным условиям.

Тогда по базисным функциям строят приближенное решение в виде линейной комбинации базисных функций:

.

Задача сводится к выбору коэффициентов таких, чтобы функция

*yn(x)* удовлетворяла граничному условию и была в некотором смысле близкой к точному решению.

Поступают следующим образом. Выражение

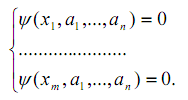
называют невязкой.

Легко видеть, что, если бы , то *yn(x)* было бы точным решением. К сожалению, так бывает очень редко. Следовательно, необходимо выбрать коэффициенты таким образом, чтобы невязка была в некотором смысле минимальной.

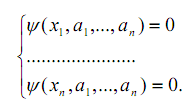
# 

# **2. Метод коллокаций**

На отрезке *[a, b]* выбираются точки, которые называются точками коллокации. Точки коллокации последовательно подставляются в невязку. Считая, что невязка должна быть равна нулю в точках коллокации, в итоге получаем систему уравнений для определения коэффициентов .



Обычно *m=n*. Получается система из *n* линейных уравнений с *n* неизвестными (коэффициентами ):



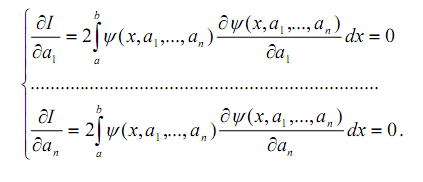
Решая эту систему найдем приближенное решение *yn(x).* Для повышения точности расширяем систему базисных функций. В значительной степени успех в применении метода зависит от удачного выбора базисной системы.

# **3. Метод наименьших квадратов**

**Интегральный МНК.** Как и в методе коллокаций приближенное решение строится по базисной системе. Для нахождения коэффициентов при базисных функциях минимизируется интеграл от квадрата невязки

.

Для нахождения минимума интеграла вычисляем первые производные от интеграла по параметрам и приравнивая их нулю строим систему нормальных уравнений



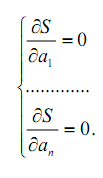
Решая ее, находим .

**Дискретный МНК.**

Выбирают *N>n* точек и решают задачу

.

Для ее решения строится система:

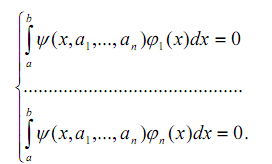


# **4.Метод Галеркина**

По базисной системе строим приближенное решение

.

Рассматриваем невязку и для определения коэффициентов при базисных функциях строим систему



# **Исходный код программы**

|  |
| --- |
| # coding: utf-8  # In[1]:  from sympy import \*  # In[2]:  x = Symbol('x')  a1 = Symbol('a1')  a2 = Symbol('a2')  y = Symbol('y')  value = Symbol('value')  # In[3]:  #метод коллокаций  # y'' + (1+x^2)y = -1  # In[34]:  def collocations(y2, psi):  s1 = psi.subs(value, -1/2.0)  s2 = psi.subs(value, 0)  s3 = psi.subs(value, 1/2.0)  print s1  print s2  print s3  r = linsolve([s1, s2, s3], a1, a2)  r1, r2 = next(iter(r))  print '\ny = {} \* (1 - x\*\*2) + {} \* (x\*\*2 - x\*\*4)'.format(r1.evalf(4), r2.evalf(4))  # In[37]:  y2 = (1 - x\*\*2) \* a1 + (x\*\*2 - x\*\*4) \* a2  psi = diff(y2, x, 2).subs(x, value) + (1 + value\*\*2) \* y2.subs(x, value) + 1  collocations(y2, psi)  # In[36]:  #2  y2 = (1 - x\*\*2) \* a1 + (x\*\*2 - x\*\*4) \* a2  psi = sin(13.0) \* diff(y2, x, 2).subs(x, value) + (1 + cos(13.0) \* value\*\*2) \* y2.subs(x, value) + 1  collocations(y2, psi)  # In[ ]:  # In[110]:  #метод наим квадратов интегральный  # In[18]:  def integr\_mnk(y2, psi):  s1 = 2 \* integrate(psi \* diff(psi, a1), (x, -1, 1) )  s2 = 2 \* integrate(psi \* diff(psi, a2), (x, -1, 1) )  print s1  print s2    r = linsolve([s1, s2], a1, a2)  r1, r2 = next(iter(r))  print '\ny = {} \* (1 - x\*\*2) + {} \* (x\*\*2 - x\*\*4)'.format(r1.evalf(4), r2.evalf(4))  # In[19]:  y2 = (1.0 - x\*\*2) \* a1 + (x\*\*2 - x\*\*4) \* a2  psi = diff(y2, x, 2) + (1 + x\*\*2) \* y2 + 1  integr\_mnk(y2, psi)  # In[20]:  #2  y2 = (1.0 - x\*\*2) \* a1 + (x\*\*2 - x\*\*4) \* a2  psi = sin(13.0) \* diff(y2, x, 2) + (1 + cos(13.0) \* x\*\*2) \* y2 + 1  integr\_mnk(y2, psi)  # In[118]:  # In[120]:  #мнк дискретный  # In[27]:  def mnk\_discr(y2, psi):  s = (psi\*\*2).subs(x, -1/2.0) + (psi\*\*2).subs(x, 0) + (psi\*\*2).subs(x, 1/2.0)  s1 = diff(s, a1)  s2 = diff(s, a2)  print s1  print s2  r = linsolve([s1, s2], a1, a2)  r1, r2 = next(iter(r))  print '\ny = {} \* (1 - x\*\*2) + {} \* (x\*\*2 - x\*\*4)'.format(r1.evalf(4), r2.evalf(4))  # In[24]:  y2 = (1 - x\*\*2) \* a1 + (x\*\*2 - x\*\*4) \* a2  psi = diff(y2, x, 2) + (1 + x\*\*2) \* y2 + 1  mnk\_discr(y2, psi)  # In[132]:  # In[28]:  #2  y2 = (1 - x\*\*2) \* a1 + (x\*\*2 - x\*\*4) \* a2  psi = sin(13.0) \* diff(y2, x, 2) + (1 + cos(13.0) \* x\*\*2) \* y2 + 1  mnk\_discr(y2, psi)  # In[135]:  # In[152]:  #Галеркина  # In[29]:  def galerkin(y2, psi):  s1 = 2 \* integrate(psi \* (1 - x\*\*2), (x, -1, 1) )  s2 = 2 \* integrate(psi \* (x\*\*2 - x\*\*4), (x, -1, 1) )  print s1  print s2  r = linsolve([s1, s2], a1, a2)  r1, r2 = next(iter(r))  print '\ny = {} \* (1 - x\*\*2) + {} \* (x\*\*2 - x\*\*4)'.format(r1.evalf(4), r2.evalf(4))  # In[30]:  y2 = (1 - x\*\*2) \* a1 + (x\*\*2 - x\*\*4) \* a2  psi = diff(y2, x, 2) + (1 + x\*\*2) \* y2 + 1  galerkin(y2, psi)  # In[31]:  #2  y2 = (1 - x\*\*2) \* a1 + (x\*\*2 - x\*\*4) \* a2  psi = sin(13.0) \* diff(y2, x, 2) + (1 + cos(13.0) \* x\*\*2) \* y2 + 1  galerkin(y2, psi) |

# **6Анализ полученных результатов**



## ***6.1 Результат работы программы***

Метод коллокаций:

*y = 0.9568 \* (1 - x\*\*2) + -0.02162 \* (x\*\*2 — x\*\*4)*

Интегральный метод наименьших квадратов:

*y = 0.9327 \* (1 - x\*\*2) + -0.06818 \* (x\*\*2 - x\*\*4)*

Дискретный метод наименьших квадратов:

*y = 0.9568 \* (1 - x\*\*2) + -0.02162 \* (x\*\*2 - x\*\*4)*

Метод Галеркина

*y = 0.9334 \* (1 - x\*\*2) + -0.05433 \* (x\*\*2 - x\*\*4)*

## ***Вывод***

Описываемые методы относятся к приближенно-аналитическим и входят в группу методов минимизации невязки.

*Метод коллокаций* наиболее простой. Однако точность во многом зависит от выбранных узлов коллокации и базисной системы. Для повышения точности систему базисных функций необходимо расширять.

*Метод наименьших квадратов* и *метод Галеркина* являются более сложным из-за вычисления скалярных произведений, требующих интегрирования функции, однако обеспечивают большую точность.