Постановка задачи

Пусть на отрезке $[0;1]$ дана функция $f \in C^{\infty}[0,1]$:

$$f:[0;1] \to \mathbb{R}\text{, причём } f(0) = f(1) = 0.$$

Пусть также дана сетка из $N+1$ узлов:

$$\begin{aligned} &x_0 = -\frac{h}{2}, \\ &x_i = x_{i-1} + h, i = 1, \ldots N, \\ &h = \frac{1}{N - 1/2} \end{aligned}$$

Введём обозначение для значений функции f в узлах:

$$f_k = f(x_k), k = 0, \ldots, N.$$

Задача состоит в том, чтобы приблизить функцию f в виде

$$f(x) \approx \sum_{m=1}^{N-1} C_m \varphi^{(m)}(x)$$

так, чтобы выполнялись равенства:

$$f_k = \sum_{m=1}^{N-1} C_m \varphi_k^{(m)}\text{, где } \varphi_k^{(m)} = \varphi^{(m)}(x_k), k = 0, \ldots, N.$$

Последнее равенство в векторном виде перепишется как

$$\begin{pmatrix} f_0 \\ f_1 \\ \vdots \\ f_N \end{pmatrix} = C_1 \begin{pmatrix} \varphi_0^{(1)} \\ \varphi_1^{(1)} \\ \vdots \\ \varphi_N^{(1)} \end{pmatrix} + \ldots + C_{N-1} \begin{pmatrix} \varphi_0^{(N-1)} \\ \varphi_1^{(N-1)} \\ \vdots \\ \varphi_N^{(N-1)} \end{pmatrix}$$

Решение

I. Мотивация задачи

Для понимания задачи стоит обратиться к её истокам. Как известно, функция $f \in C^{\infty}[0,1]$ с граничными условиями $f(0) = f(1) = 0$ раскладывается в тригонометрический ряд

$$f(x) = \sum_{m=1}^{\infty} C_m \varphi^{(m)}(x).$$

Причём функции $\varphi^{(m)}$ образуют ортогональный относительно некоторого скалярного произведения базис пространства решений задачи Штурма-Лиувилля:

$$\begin{cases} u'' = -\lambda u \\ u(0) = u(1) = 0 \end{cases}$$

Таким образом, решение задачи о разложении функции f с граничными условиями в тригонометрический ряд строится следующим образом: для нахождения $\varphi^{(m)}$ решается соответствующая задача Штурма-Лиувилля, зная скалярное произведение на пространстве решений, находятся коэффициенты разложения C_m.

II. Решение поставленной задачи

Решение поставленной **дискретной** задачи будем строить **по аналогии с** описанным в мотивации **непрерывным** случаем, поэтому в первую очередь сформулируем дискретную задачу Штурма-Лиувилля.

$$\frac{u(x+h)-u(x)}{h} \approx u'(x),$$
$$\frac{[u(x+h)-u(x)]/h-[u(x)-u(x-h)]/h}{h} \approx u''(x)$$

Поэтому в обозначениях $u_k = u(x_k)$ первая часть задачи формулируется так:

$$\frac{u_{k+1}-2u_k+u_{k-1}}{h^2} = -\lambda u_k, k=1,\dots,N-1$$

Проинтерпретируем краевые условия. Понятно, что $u_N = 0$. Следующее условие получим из соотношения

$$\frac{u(h/2)+u(-h/2)}{2} = u(0)+O(h^2).$$

Поскольку $u(0)=0$, то вторым условием будет $u_1 = -u_0$. Итак, получаем систему, описывающую **дискретную задачу Штурма-Лиувилля:**

$$\begin{cases} \dfrac{u_{k+1}-2u_k+u_{k-1}}{h^2} = -\lambda u_k, k=1,\dots,N-1 \\ u_1 = -u_0, \\ u_N = 0 \end{cases}$$

Перейдем к решению. Перепишем первое уравнение:

$$u_{k+1}-2\left(1-\frac{\lambda h^2}{2}\right)u_k+u_{k-1}=0$$

Введём обозначение $p = 1-\dfrac{\lambda h^2}{2}$. Решение будем искать в виде $u_k = \mu^k$. Если найдём, то это и есть наше решение. Уравнение, указанное выше, перепишем в терминах введенных обозначений.

$$\mu^2 - 2p\mu + 1 = 0.$$

Его корни – это $\mu_{1,2} = p \pm \sqrt{p^2-1}$, причём будем считать, что $p \neq 1$, то есть имеем два различных корня, так как в противном случае уравнение имеет только тривиальные корни. Поэтому

$$u_k = C_1\mu_1^k + C_2\mu_2^k.$$

Из первого краевого условия выводится $C_1(\mu_1+1)+C_2(\mu_2+1)=0$. По теореме Виета имеем $\mu_1\mu_2=1$, так что, заменив 1 во второй скобке придём к следующему соотношению:

$$C_1=-\mu_2 C_2\,.$$

Второе краевое условие:

$$\begin{aligned}
&C_1\mu_1^N+C_2\mu_2^N=0,\\
&-\mu_2\mu_1^N+\mu_2^N=0,\\
&\mu_1^N=\mu_2^{N-1},\\
&\mu_1^{2N-1}=1
\end{aligned}$$

В последнем переходе мы снова воспользовались теоремой Виета. Таким образом,

$$\mu=e^{\frac{\pi i 2m}{2N-1}},m=1,\ldots,2N-1$$

Учитывая симметрию μ_1 и μ_2, получаем $2N-2$ различных значений μ:

$$\mu_1=e^{\pi i\frac{2m}{2N-1}},\mu_2=e^{-\pi i\frac{2m}{2N-1}},m=1,\ldots,N-1$$

Подставим в общий вид решения и найдем **действительный базис пространства решений**:

$$\begin{aligned}
u_k^{(m)}&=-C_2\mu_2\mu_1^k+C_2\mu_2^k=C_2(\mu_2^k-\mu_2\mu_1^k)\Rightarrow\\
&=C_2(exp\{-\pi i\frac{2mk}{2N-1}\}-exp\{\pi i\frac{2mk}{2N-1}-\pi i\frac{2m}{2N-1}\})=\\
&=C_2(exp\{-\pi i\frac{2mk}{2N-1}\}-exp\{\pi i\frac{2m(k-1)}{2N-1}\})=\\
&=-2iC_2\exp\{-\pi i\frac{m}{2N-1}\}[(exp\{-\pi i\frac{2mk}{2N-1}\}-exp\{\pi i\frac{2m(k-1)}{2N-1}\})/2i]=\\
&=\tilde{C}\sin(\pi m x_k)
\end{aligned}$$

Таким образом, мы получили систему из $N-1$ независимых векторов:

$$\begin{pmatrix}\varphi_0^{(m)}\\ \varphi_1^{(m)}\\ \vdots\\ \varphi_N^{(m)}\end{pmatrix}=\begin{pmatrix}\sin(\pi m x_0)\\ \sin(\pi m x_1)\\ \vdots\\ \sin(\pi m x_N)\end{pmatrix},m=1,\ldots,N-1\ .$$

Далее, для нахождения коэффициентов разложения вектора f с помощью удобного **аппарата скалярных произведений** найдём скалярное произведение, в котором полученный базис ортогонален. Поскольку полученные вектора удовлетворяют исходной задаче Штурма-Лиувилля, верно следующее матричное соотношение:

$$\begin{pmatrix} -3/h^2 & 1/h^2 & 0 & \dots & 0 \\ 1/h^2 & -2/h^2 & 1/h^2 & \dots & 0 \\ 0 & 1/h^2 & -2/h^2 & \dots & 0 \\ \vdots & \vdots & \vdots & & \vdots \\ 0 & 0 & 0 & \dots & -2/h^2 \end{pmatrix} \begin{pmatrix} \varphi_1^{(m)} \\ \varphi_2^{(m)} \\ \vdots \\ \varphi_N^{(m)} \end{pmatrix} = -\lambda \begin{pmatrix} \varphi_1^{(m)} \\ \varphi_2^{(m)} \\ \vdots \\ \varphi_N^{(m)} \end{pmatrix}$$

Матрица симметрична, а значит **система векторов ортогональна в** обычном **евклидовом скалярном произведении**. Причём, нулевая координата выражается через первую, а N -ая равна нулю у всех векторов, поэтому их можно не рассматривать.

Убедимся прямой проверкой в том, что **вектора ортогональны**.

$$(\varphi^{(m)}, \varphi^{(n)})_h = \sum_{k=1}^{N-1} h \sin[\pi m(k-1/2)h] \sin[\pi n(k-1/2)h] =$$
$$= h\sum_{k=1}^{N-1} \cos[\pi(m-n)(k-1/2)h] - \cos[\pi(m+n)(k-1/2)h]$$

Обозначим $\pi(m-n)h = \alpha$ и рассмотрим отдельно первую сумму $h\sum_{k=1}^{N-1} \cos[\alpha(k-1/2)]$. Для её подсчёта воспользуемся **трюком** из доказательства вида **ядра Дирихле**, то есть умножим и поделим на $2\sin[\alpha/2]$. Тогда имеем:

$$h\sum_{k=1}^{N-1} \cos[\alpha(k-1/2)] = \frac{h}{2sin[\alpha/2]} \sum_{k=1}^{N-1} 2\cos[\alpha(k-1/2)]\sin[\alpha/2] =$$
$$= \frac{h}{2sin[\alpha/2]} \sum_{k=1}^{N-1} \sin\alpha k - \sin\alpha(k-1) = \frac{h\sin\alpha(N-1)}{2sin[\alpha/2]} = \frac{h\sin[\alpha(N-1/2)-\alpha/2]}{2sin[\alpha/2]} =$$
$$= \frac{h\sin[\alpha(N-1/2)-\alpha/2]}{2sin[\alpha/2]} = \frac{h\sin[\pi(m-n)-\alpha/2]}{2sin[\alpha/2]} = \frac{h}{2}(-1)^{m-n+1}$$

Аналогично для второй суммы получаем $h\sum_{k=1}^{N-1} \cos[\pi(m+n)(k-1/2)h] = \frac{h}{2}(-1)^{m+n+1}$.

Следовательно,

$$(\varphi^{(m)}, \varphi^{(n)})_h = \frac{h}{2}((-1)^{m-n+1} - (-1)^{m+n+1}) = (-1)^n \frac{h}{2}((-1)^{m+1} - (-1)^{m+2n+1}) = 0 .$$

Получили, что вектора действительно ортогональны. Теперь нахождение коэффициентов разложения f реализуется очень просто. Но сначала найдём квадрат нормы базисного вектора:

$$(\varphi^{(m)}, \varphi^{(m)})_h = h\sum_{k=1}^{N-1} \sin^2(\pi m(k-1/2)h) =$$
$$= h/2(\sum_{k=1}^{N-1} 1 - \sum_{k=1}^{N-1} \cos(2\pi m(k-1/2)h) = h/2(N-1+1/2)$$

В последнем равенстве воспользовались тем же трюком из доказательства вида ядра Дирихле. Получаем:

$$(\varphi^{(m)}, \varphi^{(m)})_h = \frac{1}{2} .$$

Коэффициенты разложения вектора по базису ортогональному в скалярном произведении есть не что иное, как

$$C_m = \frac{(f, \varphi^{(m)})_h}{(\varphi^{(m)}, \varphi^{(m)})_h} = 2h\sum_{k=1}^{N-1} f_k \varphi_k^{(m)}, \text{ где } \begin{pmatrix} f_0 \\ f_1 \\ \vdots \\ f_N \end{pmatrix} = C_1 \begin{pmatrix} \varphi_0^{(1)} \\ \varphi_1^{(1)} \\ \vdots \\ \varphi_N^{(1)} \end{pmatrix} + \ldots + C_{N-1} \begin{pmatrix} \varphi_0^{(N-1)} \\ \varphi_1^{(N-1)} \\ \vdots \\ \varphi_N^{(N-1)} \end{pmatrix}.$$

Таким образом, мы **знаем векторы разложения приближаемой функции** f и **знаем**, как находить **коэффициенты разложения** по известным значениям в узлах и векторам разложения.

Структура проекта

- main.c
- Function_2B_interpolated.c, Function_2B_interpolated.h

  Реализация функции f, которую будем интерполировать

- f2c.c, f2c.h

  Функция **void f2c(double *c, double *x, double *f, int N, double h)** – реализация алгоритма построения коэффициентов разложения вектора $(f_0, ..., f_N)$ по значениям f в узлах

- c2f.c, c2f.h

  Функция **double c2f(double *c, int N, double x)** – реализация функции, интерполирующей f, по коэффициентам, полученным с помощью f2c

- Results_to_file.h, Results_to_file.c
  Реализация функции, записывающей результаты работы программы в файл results.txt
- Error_value.c, Error_value.h

  Функция **double error_value(double *x, double *c, int n)** – реализация вычисления $Err(N) = \| f - f^N \|_{C[0,1]}$. Выбирается максимально значение модуля разности истинного значения функции f и значения функции $c2f$ среди следующих точек: узловых, добавочных. Добавочные точки: между каждыми соседними узлами добавили по 2 равноудаленные от ближайших узлов точки.

- Linear_regression.ipynb

  Реализация линейной регрессии для оценки погрешности

Обзор реализованных алгоритмов

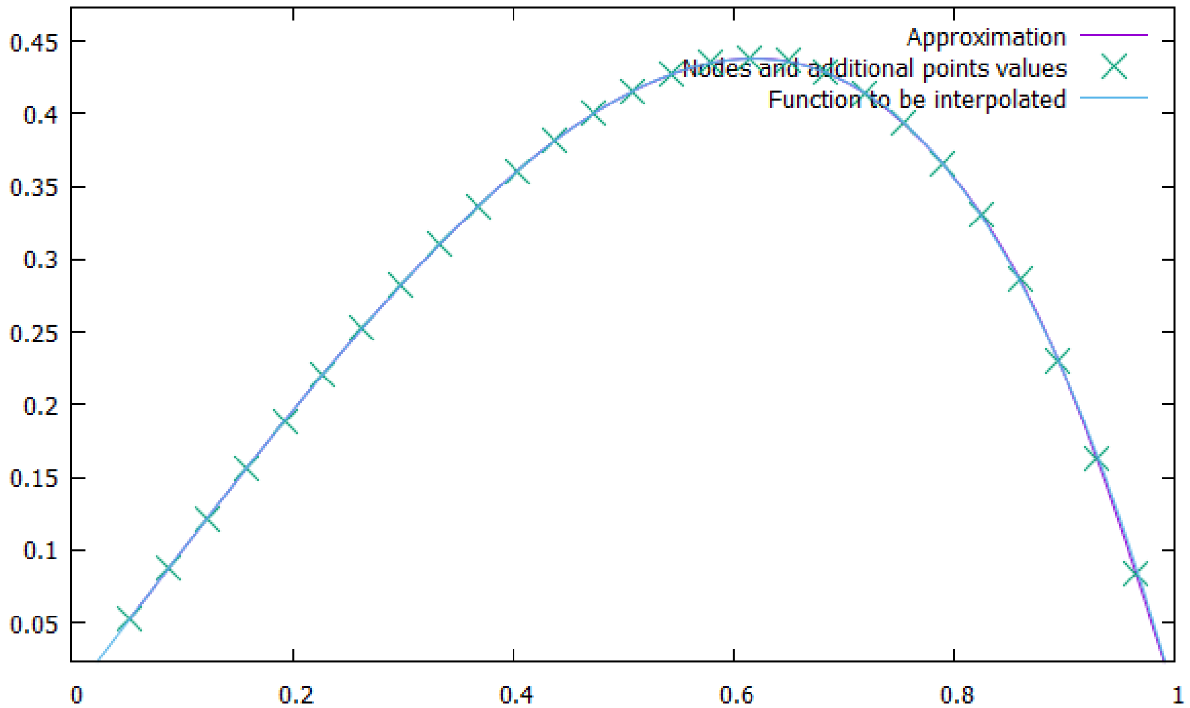
Посмотрим на работы программы при относительно небольших значениях N.

Пусть $N = 5$. Как и предполагалось, в узлах, расположенных на отрезке $[0,1]$, значения функции

x	f(x)	c2f(x)	|c2f(x)-f(x)|
0.111111	0.110372	0.110372	8.326673e-017
0.185185	0.181589	0.182134	5.444010e-004
0.259259	0.248883	0.249700	8.168857e-004
0.333333	0.310136	0.310136	1.110223e-016
0.407407	0.362844	0.361289	1.555317e-003
0.481481	0.404063	0.401971	2.091270e-003
0.555556	0.430348	0.430348	5.551115e-017
0.629630	0.437689	0.441547	3.857990e-003
0.703704	0.421435	0.426843	5.407887e-003
0.777778	0.376208	0.376208	5.551115e-017
0.851852	0.295811	0.283782	1.202903e-002
0.925926	0.173128	0.153326	1.980226e-002
1.000000	0.000000	0.000000	8.284394e-017

совпадают со значениями интерполирующей функции. Однако, замечательно то, что в промежуточных точках аппроксимация тоже относительно хорошая.

$N = 10$. Аппроксимация действительно хорошая.



Асимптотика оценки погрешности приближения

Как известно, в нашем случае $Err(N) = \| f - f^N \| \sim C\frac{1}{N^p}$. Нас интересует значение p, и для того, чтобы его найти преобразуем указанное асимптотическое соотношение:

$$-\ln Err(N) \sim \ln C^{-1} + p \ln N$$

$-\ln Err(N)$ считается с помощью функции **error_value,** как это происходит описано в разделе **Структура проекта.** $\ln N$ также считается, поскольку N задаётся нами. Таким образом, численное нахождение p (и $\ln C^{-1}$) есть задача линейной регрессии. Построим регрессию по наблюдениям, полученным при $N = 10,100,1000$. Предполагая, что при N погрешность уже выходит на асимптотику, мы получим достаточно точное значение p.

Для удобства **линейная регрессия** была реализована в отдельном файле (Linear_regression.ipynb) на языке **Python** с использованием библиотеки **sklearn**.

```
import numpy as np
import pandas as pd
from sklearn.linear_model import LinearRegression
```

```
[2] df = pd.DataFrame(columns=['1', 'lnN', 'ln(1/Err(N))'])
    lnN = []
    for N in [10, 100, 1000]:
      lnN.append(np.log(N))
    target = np.asarray([4.555523e-3, 4.181273e-5, 4.143970e-7])
    target = -np.log(target)
    df['lnN'] = lnN
    df['ln(1/Err(N))'] = target
    df['1'] = 1
    df
```

	1	lnN	ln(1/Err(N))
0	1	2.302585	5.391415
1	1	4.605170	10.082310
2	1	6.907755	14.696441

```
[3] LSM = LinearRegression()
    LSM.fit(df[df.columns[:-1]], df['ln(1/Err(N))'])
    LSM.coef_[1]
```

```
2.0205608204753096
```

Получили, что $p = 2.02$. Таким образом, $Err(N) \sim C\frac{1}{N^{2.02}}$.