МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ «ВОРОНЕЖСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ» (ФГБОУ ВО «ВГУ»)

Факультет прикладной математики, информатики и механики

Кафедра прикладной механики и информатики

Отчет по лабораторной работе №3

на тему:

**«Численные исследования способов приближения функции методом наименьших квадратов при увеличении степени многочлена»**

Выполнил: студент 3 к. 1 гр. ПМИ в.о.

Бедарев Анатолий Андреевич

Проверил: к.ф-м.н, доц.

Гудович Николай Николаевич

Воронеж – 2017

**Содержание**

|  |  |
| --- | --- |
| Постановка задачи ....................................................................................................... | 3 |
| Указания к выполнению лабораторной работы ....................................................... | 4 |
| Ход выполнения работы ............................................................................................. | 6 |
| Выводы по работе ........................................................................................................ | 15 |
| Список литературы ...................................................................................................... | 16 |
| Приложение (листинг) ................................................................................................ | 17 |
|  |  |
|  |  |
|  |  |

**Постановка задачи**

**1**. Составить и отладить программу приближенного нахождения значения функции с использованием полинома рассчитанного методом наименьших квадратов. Запрограммировать вычисление аппроксимирующего многочлена в произвольной точке x\* отрезка [a, b] следующими тремя способами:

- непрерывным способом;

- дискретным способом;

- полудискретным способом (с использованием квадратурных формул приближенного вычисления – формулы трапеций и формулы Симпсона).

*Входные данные*:

- отрезок [a, b];

- функция F(x), по которой производится расчет значений в узлах интерполяции (значения которой приближаются интерполяционным многочленом)

, ; (1)

- произвольная точка x\* отрезка [a, b], для которой считается значение интерполяционного многочлена.

В программе предусмотреть вычисление набора узловых точек *x0, x1, …, xn* (считать равноотстоящими друг от друга на отрезке [a, b]).

*Выходные данные*:

- значение многочлена в точке x\* (приближенное значение функции, Pn(x\*)).

**2**. Составить и отладить программу построения графиков исходной функции F(x) и ее аппроксимирующего полинома Pn(x), построенного по равноотстоящим узлам аппроксимации на отрезке [a, b] тремя указанными способами.

**3**. Провести численный эксперимент для выяснению вопроса о сходимости графика аппроксимирующего полинома к графику исходной функции и влиянии степени аппроксимирующего полинома и количества узловых точек на точность интерполяции. Выяснить, какой из способов применения метода наименьших квадратов более устойчив к влиянию ошибок округления.

**Указания к выполнению лабораторной работы**

***Метод наименьших квадратов (МНК)*** – это другой способ приближения функции *f*, заданной на отрезке [a, b]. Он отличен от метода интерполяции. Общим у этих двух методов является то, что приближающая функция является многочленом степени не выше *n*, т.е. многочленом вида:

 (1)

степени не выше *n*.

В МНК в качестве меры близости приближающего многочлена *Pn* к приближаемой функции *f* используется среднеквадратичное уклонение:

 (2)

Далее ставится задача о таком подборе многочлена (1), т.е. о таком подборе его коэффициентов *ci*, чтобы среднеквадратичное уклонение (2) было бы минимальным. Таким образом, осуществляется переход к оптимизационной задаче (в данном случае, к задаче нахождения минимума функции):

 (3)

От квадратного корня в задаче (3) можно избавиться, это не поменяет результатов оптимизации:

 (4)

Чтобы решить оптимизационную задачу (4), необходимо вычислить частные производные этой функции по переменным *ci* и приравнять эти производные к нулю. В результате получится система уравнений для нахождения коэффициентов многочлена наилучшего среднеквадратичного приближения для функции *f* на отрезке [a, b].

Описанный способ относится к ***непрерывному способу МНК*** вычисления аппроксимирующего многочлена.

Кроме описанного выше существует также ***дискретный вариант МНК***. Дискретность в этом наименовании означает, что этот вариант применяется не к функции, заданной на отрезке приближения [a, b], а к функции, заданной таблицей своих значений в точках *x0, x1, …, xn*.

В дискретном варианте ищут минимум функции:

 (5)

Т.е. интеграл в ф.(4) заменяем суммой и ищем такой набор коэффициентов, который минимизирует величину F.

После этого подставляем найденные значения в выражение (1) и получаем искомый полином.

Кроме описанных выше, непрерывного и дискретного вариантов МНК имеется и ***полудискретный вариант***. Суть его заключается в том, что скалярные произведения в левой части системы линейных алгебраических уравнений для нахождения коэффициентов *ci* приближающего многочлена *Pn* определяются как и в непрерывном случае через определенные интегралы по отрезку [a, b], а скалярные произведения в правой части системы, выражающиеся через определенные интегралы от произведения приближаемой функции *f* на степени независимой переменной, находятся с помощью формул для приближенного вычисления определенных интегралов (квадратурные формулы).

Чаще всего используются следующие интерполяционные квадратурные формулы:

- локально-интерполяционная формула трапеций;

- формула Симпсона (формула парабол).

**Ход выполнения работы**

Выполнение лабораторной работы проводилось только на языке технического моделирования MatLab R2017a. Отказ от применения традиционных систем разработки приложений в пользу системы научно-технических расчетов MatLab обусловлен широкими возможностями последней в плане реализации работы с матрицами и символьными вычислениями.

Для выполнения поставленных задач было разработано несколько программных модулей (см. Приложение).

**На первом этапе** в среде MatLab реализованы три описанных выше алгоритма МНК для нахождения коэффициентов аппроксимирующего полинома; алгоритмы оформлены отдельными программными модулями в следующих файлах:

* *coefMNKSolid.m* – модуль, реализующий вычисление коэффициентов непрерывным способом;
* *coefMNKDiscrete.m* – модуль, реализующий вычисление коэффициентов дискретным способом;
* *coefMNKQuasiDiscrete.m* – модуль, реализующий вычисление коэффициентов полудискретным способом.

Для вычисления значений функции F(x) для узла x использовался реализованный ранее в первых двух лабораторных работах модуль *f.m*.

Кроме того, поскольку в модулях непрерывного и полудискретного способов используются символьные вычисления, в отдельный модуль *f\_sym.m*. была вынесена функция, возвращающая своим результатом символьное представление исходной функции f.

Модуль *coefMNKSolid* получает на вход в качестве параметров концы отрезка [a, b] и степень аппроксимирующего полинома n, возвращая массив коэффициентов полинома *ci*. Внутри модуля задействованы символьные вычисления, которые осуществляют символьное интегрирование и дифференцирование с последующим решением системы алгебраических уравнений.

Модули *coefMNKDiscrete* и *coefMNKQuasiDiscrete* получают на вход в качестве параметровмассивы координат узлов, например, следующего вида:

*X = [1; 2; 3],*

*Y = [-2; 1; 6],*

а также степень аппроксимирующего полинома n, возвращая массив коэффициентов полинома *ci*. Кроме того, дополнительным параметром *coefMNKQuasiDiscrete* принимает *typeQuadra* – номер квадратурной формулы, применяемой для приближенного вычисления интегралов в правой части уравнений (1 – формула трапеций, 2 – формула Симпсона).

Вызов модулей осуществляется командами:

*coefMNKSolid(a, b, n)*

*coefMNKDiscrete(X, Y, n)*

*coefMNKQuasiDiscrete(X, Y, n, typeQuadra)*

В свою очередь, чтобы однозначно определить интерфейс, обеспечивающий доступ ко всем трем модулям, реализован промежуточный объединяющий модуль *coefMNKBase.m* с командой для его вызова следующего вида:

*coefMNKBase(a, b, n, N, typeMNK, typeQuadra)*

где передаваемые параметры a, b, n – соответственно границы отрезка [a, b] и степень интерполяционного полинома, а *typeMNK* – номер способа, которым рассчитываются коэффициенты аппроксимирующего полинома (1 - непрерывный, 2 - дискретный, 3 - полудискретный), *typeQuadra* – номер квадратурной формулы, применяемой для приближенного вычисления интегралов в правой части уравнений (1 – формула трапеций, 2 – формула Симпсона). Часть параметров опциональна и задействуется только в определенном методе.

Модуль возвращает матрицу коэффициентов С, чтобы затем на ее основе можно было рассчитать значение аппроксимирующего полинома в любой точке.

**На втором этапе** в среде MatLab разработан модуль *pointMNK.m*, реализующий вычисление значения интерполяционного многочлена в точке x, а также дополнительные модули для формирования графиков.

Первый модуль (*pointMNK.m*) получает в качестве передаваемых параметров матрицу коэффициентов C и значение x, для которого надо посчитать P(x):

*Px = pointMNK(C, x)*

Построение графиков выполняют следующие модули:

* *plotMNK.m* – выводит одиночный график полинома, совмещенный с графиком функции и узловыми точками;
* *plotMNKFull.m* – выводит совмещенные графики полиномов, коэффициенты которых посчитаны всеми тремя способами (для сравнения);
* *plotDeltaMNK.m* – выводит совмещенные графики зависимости средней абсолютной ошибки аппроксимации от степени аппроксимирующего полинома n (для всех трех способов для сравнения);
* *plotDelta2MNK.m* – выводит совмещенные графики зависимости средней абсолютной ошибки аппроксимации от количества узловых точек N (для всех трех способов для сравнения);
* *plot3DeltaMNK.m* – выводит 3D графики зависимость точности аппроксимации МНК от n и N;
* *deltaMNK.m* – впомогательный модуль для расчета средней абсолютной ошибки аппроксимации.

Вызываются эти модули следующим образом:

*plotMNK(a, b, n, N, typeMNK, typeQuadra)*

*plotMNKFull(a, b, n, N, typeQuadra)*

*plotDeltaMNK(a, b, N, n\_start, n\_end, typeQuadra)*

*plotDelta2MNK(a, b, n, N\_start, N\_end, typeQuadra)*

*plot3DeltaMNK(a, b, n\_start, n\_end , N\_start, N\_end, typeMNK, typeQuadra)*

*deltaMNK(Y1, Y2)*

где a и b - границы отрезка, n - степень полинома, *typeMNK* – номер способа, которым рассчитываются коэффициенты аппроксимирующего полинома (1 - непрерывный, 2 - дискретный, 3 - полудискретный), *typeQuadra* – номер квадратурной формулы, применяемой для приближенного вычисления интегралов в правой части уравнений (1 – формула трапеций, 2 – формула Симпсона),   
*n\_start* и *n\_end* – начальное и конечное значение диапазона перебора степени аппрокисмирующего полинома, *N\_start, N\_end* – начальное и конечное значение диапазона перебора числа узловых точек.

При построении графиков равноотстоящие узлы интерполяции дополняются промежуточными точками, поскольку узловых точек недостаточно для демонстрации расхождения графиков (в узловых точках они совпадают).

**На третьем этапе** для выяснения вопроса о сходимости графика аппроксимирующего полинома к графику исходной функции, влиянии степени аппроксимирующего полинома и способа расчета его коэффициентов, а также количества узловых точек на точность интерполяции выполнен численный эксперимент. Результаты представлены на рис. 1 – 10.

|  |  |
| --- | --- |
| а) | б) |
|  |  |
| при n = 2, N = 3; | при n = 3, N = 3; |
|  |  |
| в) | г) |
|  |  |
| при n = 4, N = 3; | при n = 5, N = 3; |
|  |  |
| д) | е) |
|  |  |
| при n = 6, N = 3; | при n = 7, N = 3. |

Рисунок 1 – Сравнение точности приближения при различных значениях степени аппроксимирующего полинома для всех трех способов

|  |  |
| --- | --- |
| а) | б) |
| C:\Users\hroniko\AppData\Local\Microsoft\Windows\INetCache\Content.Word\n8_N3_Tr.png |  |
| при n = 8, N = 3; | при n = 9, N = 3; |
|  |  |
| в) | г) |
|  |  |
| при n = 10, N = 3; | при n = 4, N = 5; |
|  |  |
| д) | е) |
|  |  |
| при n = 4, N = 10; | при n = 4, N = 20. |

Рисунок 2 – Сравнение точности приближения при различных значениях степени аппроксимирующего полинома для всех трех способов

Изменяя степень аппроксимирующего полинома от 2 до 10 и количество узловых точек от 3 до 20, были получены сравнительные графики (рис. 1 и 2). Их анализ показал, прежде всего, что ***аппроксимирующий многочлен для нашей функции обладает сходимостью с точностью до некоторой постоянной ошибки***.

В отношении зависимости от степени аппроксимирующего полинома дискретный и полудискретный способы расчета коэффициентов полинома МНК проявляют ***схожие свойства***: при небольших значениях степени полинома с увеличением степени наблюдается постепенное улучшение приближения к функции. Затем этот рост сходимости замедляется, и, начиная с некоторого номера, увеличение степени полинома уже не приводит к существенному росту приближения, достигая устойчивой ошибки аппроксимации (см. рис. 3).

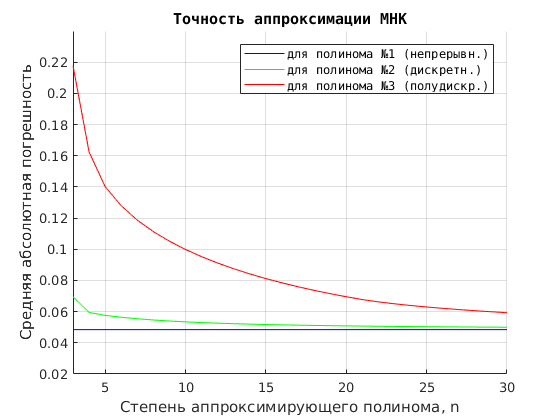


Рисунок 3 – Зависимость средней ошибки аппроксимации от степени аппроксимирующего полинома: из-за различий в алгоритмах расчета влияние ошибки округления начинает сказываться при разных значениях N

В то же время для ***непрерывного способа*** уже после достижения значения степени полинома n = 4 дальнейшее ее увеличение не дает никаких положительных результатов; это связано с особенностями непрерывного метода (следует заметить, однако, что при вычислении ошибки аппроксимации использовались дискретные методы, что явилось причиной накопления ошибок округления и в итоге установления некоторого значения ошибки аппроксимации на рис. 3 на отметке 0.05).

***Для дискретного метода*** характерно более быстрое достижение точности приближения, чем для полудискретного метода. Это связано, очевидно, с большими ошибками округления у полудискретного метода, наличие которых объясняет использование приближенных формул вычисления интеграла ( квадратурные формулы).

Следует отметить также, ***что применение формул трапеций оказалось предпочтительнее для нашей функции, чем формулы Симпсона, поскольку для полудискретного способа это позволяет существенно увеличить сходимость*** (рис. 4).

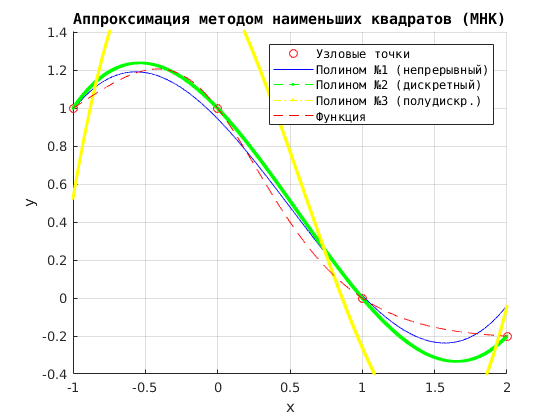
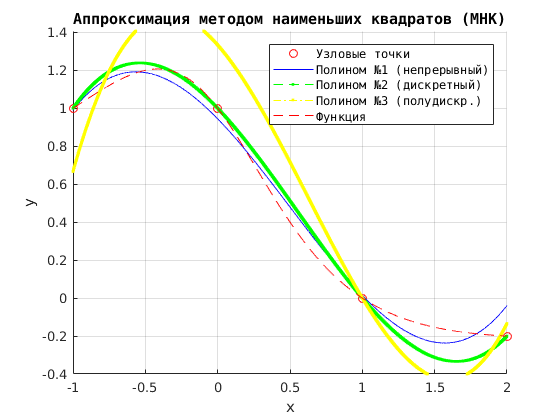


Рисунок 4 – Сравнение точности аппроксимации при различных значениях степени аппроксимирующего полинома для всех трех способов (желтой линией нарисован график аппроксимирующего полинома): метод трапеций в полудискретном способе (слева) дает меньшую ошибку по сравнению с методом Симпсона (справа)

Особо интересно поведение аппроксимирующего полинома на пространстве переменных n и N. Для непрерывного метода зависимость ошибки аппроксимации от n и N представляет собой линии уровня, неизменные вдоль N (рис. 5).

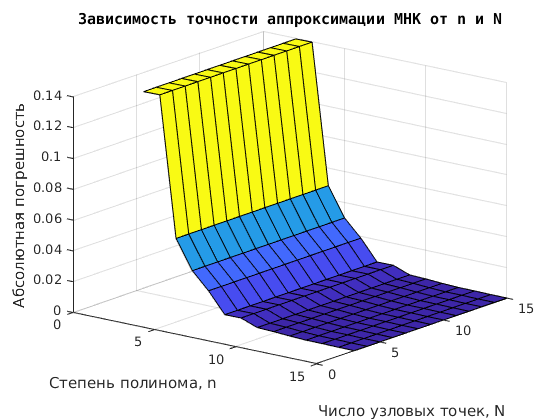


Рисунок 5 – Зависимость ошибки аппроксимации от степени полинома n и количества узловых точек N: n =[3…15]; N =[3…15]

Таким образом, N совершенно точно не влияет на точность аппроксимации непрерывного метода (хотя это и ожидаемо, поскольку N не участвует в расчете коэффициентов полинома!).

Что касается дискретного и полудискретного методов, то для них зависимости точности аппроксимации от n и N носят более сложный характер. В частности, установлено, что увеличение n и N способствуют увеличению точности аппроксимации, но для участков, где N изменяется от 4 до 7, а n – от 10 и выше, наблюдаются резкие скачки с резким увеличение ошибки аппроксимации. Данную особенность хорошо иллюстрирует рис. 6.

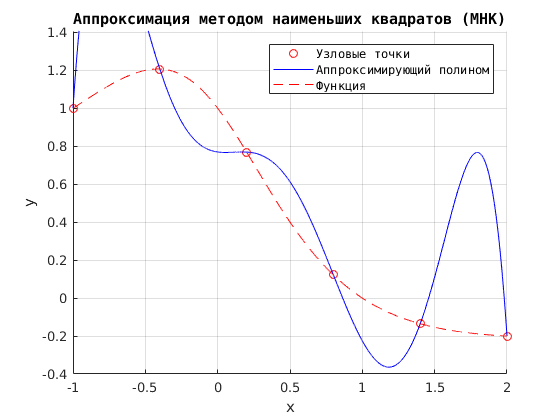
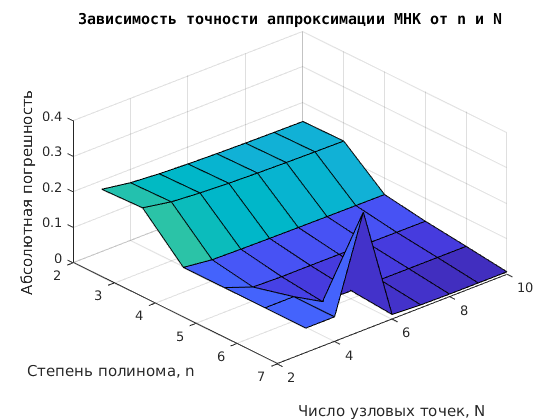


Рисунок 6 – Характерные «всплески» роста ошибки   
аппроксимации (слева) при больших значениях n   
и малых значениях N и объясняющее их поведение  
 графика аппроксимирующего полинома (справа)

Из рис. 6 видно, что «всплески» обусловлены колебательным поведением аппроксимирующего полинома, который начинает вести себя как интерполяционный полином Лагранжа (пересекает все узловые точки) при n превышающих N.

На рис. 7 и 8 показаны зависимости ошибки аппроксимации от n и N для дискретного и полудискретного методов расчета коэффициентов.

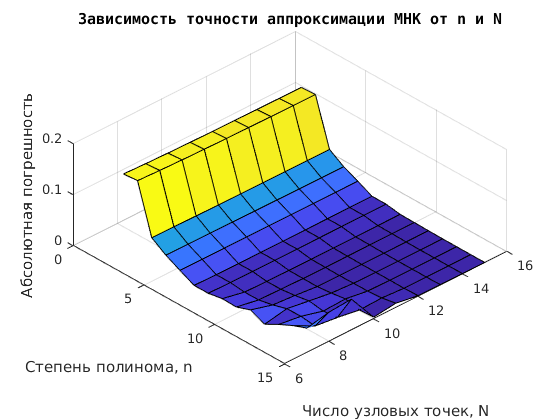


Рисунок 7 – Зависимость ошибки аппроксимации от n и N   
для дискретного метода расчета коэффициентов   
полинома: n =[2…15]; N =[7…15]

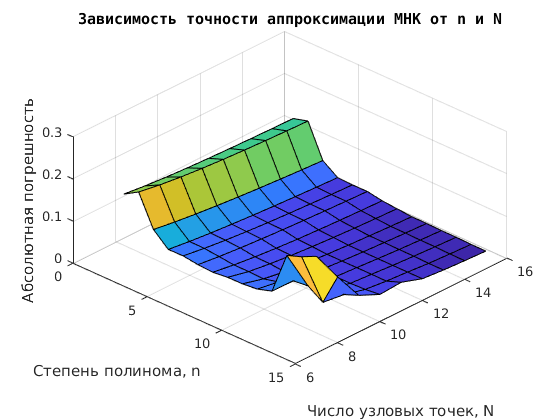


Рисунок 8 – Зависимость ошибки аппроксимации от n и N   
для полудискретного метода расчета коэффициентов   
полинома: n =[2…15]; N =[7…15]

Таким образом, для непрерывного метода в качестве достаточной степени можно рекомендовать n = 4; в отношении полудискретного и дискретного способов, где влияние также оказывает количество узловых точек N, достаточно ограничится значениями n = 8 и N = 12 (для дискретного метода) и n = 12 и N = 16 (для полудискретного метода).

**Выводы по работе**

1. В ходе лабораторной работе рассмотрен способ нахождения значения функции с использованием полинома, рассчитанного методом наименьших квадратов, а также разработаны функциональные модули для системы MatLab, с помощью которых можно проводить аппроксимацию функций.

2. В результате численного эксперимента установлено, что аппроксимирующий полином для заданной функции обладает сходимостью с точностью до некоторой постоянной ошибки, при этом она тем меньше, чем выше степень полинома – для непрерывного метода, и чем выше степень полинома и количество узловых   
точек – для дискретного и полудискретного способов. Ошибка аппроксимации обусловлена как использованием приближенным методов расчета, так и ошибками округления.

3. Наилучшим в плане сходимости способом расчета является непрерывный; в силу особенностей расчета он не зависит от количества узловых точек N и дает хорошую сходимость при меньших значениях n, чем его дискретный и полудискретный аналоги. Однако непрерывный способ требует реализации на ЭВМ символьных вычислений, что в ряде случаев затруднено, т.к. налагает дополнительные требования к системе и используемым ресурсам.

4. Полудискретный метод расчета уступает по точности дискретному в силу использования приближенных методов вычисления интегралов; из-за этой особенности полудискретный метод сильнее зависит от количества узловых точек и ему требуется большее их количество для эффективного приближения полинома к исходной функции.

5. В полудискретном и дискретном способах существует особое соотношение количества узловых точек и степени полинома, для которого есть риск возникновения колебательного поведения аппроксимирующего полинома. В этом случае последний начинает вести себя как интерполяционный полином Лагранжа (пересекает все узловые точки). Поэтому при назначении n и N требуется тщательный анализ поведения аппроксимирующего полинома.

**Список литературы**

1. *Гудович А.Н., Гудович Н.Н.* Элементы численных методов: учебное пособие. Вып 3. Метод наименьших квадратов. Воронеж: Издательско-полиграфический центр Воронежского государственного университета, 2016 г. – 32 с.
2. *Кетков, Ю.Л. и др.* Matlab 7: программирование, численные методы / Ю.Л. Кетков, А.Ю. Кетков, М.М. Шульц, - СпБ.: БХВ-Петербург, 2005 г. – 752 с.

**Приложение (листинг)**

|  |
| --- |
| Модуль MatLab **f.m** |

function Y=f(X)

Y = (1-X.)/(1+X.^2);

end

|  |
| --- |
| Модуль MatLab **f\_sym.m** |

function y=f(X)

syms x; %Определение символьной переменной

y = sym ((1-x)/(1+x^2));

end

|  |
| --- |
| Модуль MatLab coefMNKSolid.m |

function cc = coefMNKSolid(a, b, n)

syms x; %Определение символьной переменной

c = sym('c', [1 n]); % Определяем массив символьных переменных с1, ..., сn

% 1 Собираем символьный многочлен p:

tmp = 0;

for i = 1:n % от c1 до сn-1

tmp = tmp + c(i)\*x^(i-1); %

end

p = sym(tmp);

% 2 Определяем нашу символьную функцию (из внешнего файла):

f = f\_sym;

% 3 Формируем подынтегральную функцию:

g = sym((f-p)^2);

% 4 Вычисляем определенный интеграл на отрезке [a, b] по указанной

% переменной, отнормировав на длину ab:

F = 1/(b-a)\* int(g, x, a, b);

% 5 Производим символьное дифференцирование:

c\_diff = sym('c\_diff', [1 n]);

for i = 1:n % от c1 до сn-1

c\_diff(i) = diff(F, c(i)); % символьное дифференцирование

end

% 6 Символьно решаем систему линейных алгебраических уравнений

c = solve (c\_diff);

% 7 Получаем массив строк с именами всех полей структуры "c":

cname = fieldnames(c);

% 8 Получаем кодержимое каждого поля структуры, обращаясь по имени поля, и

% конвертируем в double, сохраняя в итоговый массив

for i = 1:n % от c1 до сn-1

%getfield(S, 'field') — возвращает содержимое поля структуры S, что эквивалентно S. field;

cname\_cell = cname(i);

cc(i) = double(getfield(c, cname\_cell{1})); % Фигурные скобки для обращения к содержимому ячейки

end

end

|  |
| --- |
| Модуль MatLab coefMNKDiscrete.m |

function cc = coefMNKDiscrete(X, Y, n)

% Дискретный метод вычисления коэфиициентов аппроксимирующего полинома

N = length(X); % Количество узловых точек

syms x; %Определение символьной переменной

c = sym('c', [1 n]); % Определяем массив символьных переменных с1, ..., сn

% 1 Собираем матрицу уравнений:

u = sym('u', [1 n]); % Определяем массив символьных переменных u1, ..., un

% Для каждой символьной переменной формируем содержимое:

for k = 1:n % от u1 до u(n-1)

% 1.1 Считаем левую сумму:

summ\_L = 0;

% Организуем цикл по переменным ci

for i = 1:n % от c1 до c(n-1)

% 1.1 Для каждой переменной ci организуем цикл по фj (считаем <фi,фk>):

for j = 1:N

summ\_L = summ\_L + ( X(j)^(i-1) ) \* ( X(j)^(k-1) ) \* c(i);

end

end

% 1.2 Нормируем

summ\_L = 1/N \* summ\_L;

% 1.3 Считаем правую сумму:

summ\_R = 0;

% Организуем цикл по по fj (считаем <f,фk>):

for j = 1:N

summ\_R = summ\_R + Y(j) \* ( X(j)^(k-1) );

end

% 1.4 Нормируем

summ\_R = 1/N \* summ\_R;

% 1.5 Вычитаем из левой суммы правую и окончательно формируем уравнение:

summ = summ\_L - summ\_R;

% 1.6 Переводим к символьному виду и записываем в соответсвующий

% элемент символьной матрицы уравнений

u(k) = sym(summ);

end

% 2 Символьно решаем систему линейных алгебраических уравнений

c = solve (u);

% 3 Получаем массив строк с именами всех полей структуры "c":

cname = fieldnames(c);

% 4 Получаем содержимое каждого поля структуры, обращаясь по имени поля, и

% конвертируем в double, сохраняя в итоговый массив

for i = 1:n % от c1 до сn-1

%getfield(S, 'field') — возвращает содержимое поля структуры S, что эквивалентно S. field;

cname\_cell = cname(i);

cc(i) = double(getfield(c, cname\_cell{1})); % Фигурные скобки для обращения к содержимому ячейки

end

end

|  |
| --- |
| Модуль MatLab coefMNKQuasiDiscrete.m |

function cc = coefMNKQuasiDiscrete(X, Y, n, typeQuadra)

% Полудискретный метод вычисления коэфиициентов аппроксимирующего полинома

% X - матрица узловых точек (иксы)

% Y - матрица узловых точек (игреки)

% n - желаемая степень аппроксимирующего полинома

% N - вытаскиваем из размерности X

% type - тип квадратурной формулы для приближенного вычисления интеграла,

% если 1, то по формуле трапеций, если 2 - то по формуле Симпсона

N = length(X); % Количество узловых точек

syms x; %Определение символьной переменной

c = sym('c', [1 n]); % Определяем массив символьных переменных с1, ..., сn

% 1 Собираем матрицу уравнений:

u = sym('u', [1 n]); % Определяем массив символьных переменных u1, ..., un

% Для каждой символьной переменной формируем содержимое:

for k = 1:n % от u1 до u(n-1)

% 1.1 Считаем левую сумму (как в дискретном варианте):

summ\_L = 0;

% Организуем цикл по переменным ci

for i = 1:n % от c1 до c(n-1)

% 1.1 Для каждой переменной ci организуем цикл по фj (считаем <фi,фk>):

for j = 1:N

summ\_L = summ\_L + ( X(j)^(i-1) ) \* ( X(j)^(k-1) ) \* c(i);

end

end

% 1.2 Нормируем

summ\_L = 1/N \* summ\_L;

% 1.3 Считаем правую сумму:

summ\_R = 0;

if (typeQuadra == 1) % считаем по формуле трапеций

% 1.3.1 Собираем полудискретную сумму методом трапеций

for i = 1:N-1

summ\_R = summ\_R + ( Y(i) \* X(i)^(k-1) + Y(i+1) \* X(i+1)^(k-1) ) \* (X(i+1)-X(i))/2;

end

% 1.3.2 Нормируем

summ\_R = 1/(X(N)-X(1)) \* summ\_R;

end

if (typeQuadra == 2) % считаем по формуле Симпсона

% 1.3.3 Собираем полудискретную сумму методом трапеций

for i = 1:N-1

summ\_R = summ\_R + ( Y(i) \* X(i)^(k-1) + 4 \* f( (X(i)+X(i+1))/2 ) \* ((X(i)+X(i+1))/2)^(k-1) + Y(i+1) \* X(i+1)^(k-1) ) \* (X(i+1)-X(i))/6;

end

% 1.3.4 Нормируем

summ\_R = 1/(X(N)-X(1)) \* summ\_R;

end

% 1.5 Вычитаем из левой суммы правую и окончательно формируем уравнение:

summ = summ\_L - summ\_R;

% 1.6 Переводим к символьному виду и записываем в соответсвующий

% элемент символьной матрицы уравнений

u(k) = sym(summ);

end

% 2 Символьно решаем систему линейных алгебраических уравнений

c = solve (u);

% 3 Получаем массив строк с именами всех полей структуры "c":

cname = fieldnames(c);

% 4 Получаем содержимое каждого поля структуры, обращаясь по имени поля, и

% конвертируем в double, сохраняя в итоговый массив

for i = 1:n % от c1 до сn-1

%getfield(S, 'field') — возвращает содержимое поля структуры S, что эквивалентно S. field;

cname\_cell = cname(i);

cc(i) = double(getfield(c, cname\_cell{1})); % Фигурные скобки для обращения к содержимому ячейки

end

end

|  |
| --- |
| Модуль MatLab coefMNKBase.m |

function cc = coefMNKBase(a, b, n, N, typeMNK, typeQuadra)

% Интерфейсная функция для вызова всех других функций

% a - начало отрезка

% b - конец отрезка

% n - степень полинома

% N - количество точек разбиения

% typeMNK - тип МНК, 1 - непрерывный, 2 - дискретный, 3 - полудискретный

% typeQuadra (только для 3 типа МНК) - тип квадратурной формулы, 1 -

% трапеций, 2 - Симпсона

if (typeMNK == 1) % Если непрерывный способ, то

cc = coefMNKSolid(a, b, n); % вычисляем

end

if (typeMNK == 2) % Если дискретный способ, то

% Считаем значения X и Y в равноотстоящих точках на отрезке [a, b]

X = linspace(a, b, N+1); % N+1 узловых точек

Y = f(X);

cc = coefMNKDiscrete(X, Y, n); % вычисляем

end

if (typeMNK == 3) % Если полудискретный способ, то

% Считаем значения X и Y в равноотстоящих точках на отрезке [a, b]

X = linspace(a, b, N+1); % N+1 узловых точек

Y = f(X);

cc = coefMNKQuasiDiscrete(X, Y, n, typeQuadra); % вычисляем

end

% Во всех остальных случаях вернем как по первому случаю

if (typeMNK < 1)

cc = coefMNKSolid(a, b, n); % вычисляем

end

if (typeMNK > 3)

cc = coefMNKSolid(a, b, n); % вычисляем

end

end

|  |
| --- |
| Модуль MatLab pointMNK.m |

function p = pointMNK(C, xx)

% Функция рассчета значения аппроксимирующего полинома в точке xx

% На вход получает матрицу коэффициентов C, матрицу узловых точек X и

% значение xx, для которого надо посчитать P(xx)

n = length(C); % Количество узловых точек

p = C(1); % Начальное значение полинома в точке

for i = 2 : n % Внешний цикл по коэффициентам C(i)

p = p + C(i)\*xx^(i-1);

end

end

|  |
| --- |
| Модуль MatLab plotMNK.m |

function plotMNK(a, b, n, N, typeMNK, typeQuadra)

% Основная функция, вычисляет коэффициенты аппроксимирующего полинома по одному из

% методов рассчета и строит сравнительные графики для исходной функции и

% аппроксимирующего полинома, с нанесением узловых точек

% 1 Создаем новое окно для графика и подписываем оси

figure;

xlabel('x');

ylabel('y');

hold on;

grid on;

% 2 Вычисляем коэффициенты аппроксимирующего полинома и узловые точки через базовую

% функцию:

C = coefMNKBase(a, b, n, N, typeMNK, typeQuadra);

% 3 Печатаем узловые точки:

X = linspace(a, b, N+1); % N+1 узловых точек

Y = f(X);

plot (X, Y, 'ro');

% 3 Печатаем график аппроксимирующего полинома:

XX = linspace(a, b, 10000);

LX = XX \* 0;

for i = 1:length(XX)

LX(i) = pointMNK(C, XX(i));

end

plot(XX, LX, 'b');

% 4 Печатаем график функции:

X = XX;

Y = f(X);

plot(X, Y, 'r--');

% 5 Подписываем легенду

title('Аппроксимация методом наименьших квадратов (МНК)', 'FontName', 'Courier');

name = strcat('Аппроксимирующий полином');

h1 = legend('Узловые точки', 'Аппроксимирующий полином', 'Функция');

set(h1, 'FontName', 'Courier');

% 6 Выставляем более-менее приемлемый масштаб:

axis([a b min(Y)-0.2 max(Y)+0.2])

end

|  |
| --- |
| Модуль MatLab plotMNKFull.m |

function plotMNKFull(a, b, n, N, typeQuadra)

% Основная функция, вычисляет коэффициенты аппроксимирующего полинома по каждому из

% методов рассчета и строит сравнительные графики для исходной функции и

% трех аппроксимирующих полиномов, с нанесением узловых точек

% 1 Создаем новое окно для графика и подписываем оси

figure;

xlabel('x');

ylabel('y');

hold on;

grid on;

% 2 Вычисляем коэффициенты аппроксимирующего полинома №1 (непрерывный метод)

% и узловые точки через базовую функцию:

C1 = coefMNKBase(a, b, n, N, 1, typeQuadra);

% 3 Печатаем узловые точки:

X = linspace(a, b, N+1); % N+1 узловых точек

Y = f(X);

plot (X, Y, 'ro');

% 4 Вычисляем коэффициенты аппроксимирующего полинома №2 (дискретный метод)

% через вспомогательную функцию:

C2 = coefMNKBase(a, b, n, N, 2, typeQuadra);

% 5 Вычисляем коэффициенты аппроксимирующего полинома №3 (полудискретный)

% через вспомогательную функцию:

C3 = coefMNKBase(a, b, n, N, 3, typeQuadra);

% 6 Печатаем график полинома №1:

XX = linspace(a, b, 10000);

LX = XX \* 0;

for i = 1:1:length(XX)

LX(i) = pointMNK(C1, XX(i));

end

plot(XX, LX, 'b');

% 7 Печатаем график полинома №2:

for i = 1:1:length(XX)

LX(i) = pointMNK(C2, XX(i));

end

plot(XX, LX, 'g--.');

% 8 Печатаем график полинома №3:

for i = 1:1:length(XX)

LX(i) = pointMNK(C3, XX(i));

end

plot(XX, LX, 'y-..');

% 9 Печатаем график функции:

Y = f(XX);

plot(XX, Y, 'r--');

% 10 Подписываем легенду

title('Аппроксимация методом наименьших квадратов (МНК)', 'FontName', 'Courier');

h1 = legend('Узловые точки', 'Полином №1 (непрерывный)', 'Полином №2 (дискретный)', 'Полином №3 (полудискр.)', 'Функция');

set(h1, 'FontName', 'Courier');

% 11 Выставляем более-менее приемлемый масштаб:

axis([a b min(Y)-0.2 max(Y)+0.2])

end

|  |
| --- |
| Модуль MatLab deltaMNK.m |

function d = deltaMNK(Y1, Y2)

% Функция вычисления погрешности между значениями функции и значениями

% аппроксимирующего полинома

n = length(Y1); % Количество узловых точек

D = Y1 \* 0; % Матрица разности

for i = 1 : n

D(i) = Y1(i) - Y2(i);

end

d = 0;

for i = 1 : n

d = d + abs(D(i));

end

d = d / n;

end

|  |
| --- |
| Модуль MatLab plotDeltaMNK.m |

function plotDeltaMNK(a, b, N, size\_start, size\_end, typeQuadra)

color = ['b'; 'g'; 'r']; % Матрица цветов

% 1 Создаем новое окно для графика и подписываем оси

figure;

xlabel('Степень аппроксимирующего полинома, n');

ylabel('Средняя абсолютная погрешность');

grid on; hold on;

NN = N;

for typeMNK = 1:3 % Для все трех способов расчета

EN = size\_start : size\_end;

E = EN \* 0; % Создаем матрицу ошибок

k = 1;

for n = size\_start : size\_end

try

C = coefMNKBase(a, b, n, NN, typeMNK, typeQuadra);

% Считаем значения X и Y в равноотстоящих точках на отрезке [a, b]

X1 = linspace(a, b, 10000); % 10000 узловых точек

Y1 = f(X1);

% Считаем значения полинома в этих 10000 точках:

Y2 = Y1 \* 0;

for i = 1:length(X1)

Y2(i) = pointMNK(C, X1(i));

end

% Считаем значения ошибок

E(k) = deltaMNK(Y1, Y2);

catch ME

E(k) = E(k-1);

end

k = k+1;

end

% 3 Печатаем график:

plot (EN, E, color(typeMNK));

end

% 5 Подписываем легенду

title('Точность аппроксимации МНК', 'FontName', 'Courier');

h1 = legend('для полинома №1 (непрерывн.)', 'для полинома №2 (дискретн.)', 'для полинома №3 (полудискр.)');

set(h1, 'FontName', 'Courier');

end

|  |
| --- |
| Модуль MatLab plotDelta2MNK.m |

function plotDelta2MNK(a, b, n, N\_start, N\_end, typeQuadra)

color = ['b'; 'g'; 'r']; % Матрица цветов

% 1 Создаем новое окно для графика и подписываем оси

figure;

xlabel('Степень аппроксимирующего полинома, n');

ylabel('Средняя абсолютная погрешность');

grid on; hold on;

for typeMNK = 1:3 % Для все трех способов расчета

EN = N\_start : N\_end;

E = EN \* 0; % Создаем матрицу ошибок

k = 1;

for N = N\_start : N\_end

try

C = coefMNKBase(a, b, n, N, typeMNK, typeQuadra);

% Считаем значения X и Y в равноотстоящих точках на отрезке [a, b]

X1 = linspace(a, b, 10000); % 10000 узловых точек

Y1 = f(X1);

% Считаем значения полинома в этих 10000 точках:

Y2 = Y1 \* 0;

for i = 1:length(X1)

Y2(i) = pointMNK(C, X1(i));

end

% Считаем значения ошибок

E(k) = deltaMNK(Y1, Y2);

catch ME

E(k) = E(k-1);

end

k = k+1;

end

% 3 Печатаем график:

plot (EN, E, color(typeMNK));

end

% 5 Подписываем легенду

title('Точность аппроксимации МНК', 'FontName', 'Courier');

h1 = legend('для полинома №1 (непрерывн.)', 'для полинома №2 (дискретн.)', 'для полинома №3 (полудискр.)');

set(h1, 'FontName', 'Courier');

% 6 Выставляем более-менее приемлемый масштаб:

axis([N\_start N\_end 0 0.24])

end

|  |
| --- |
| Модуль MatLab plot3DeltaMNK.m |

function plot3DeltaMNK(a, b, n\_start, n\_end , N\_start, N\_end, typeMNK, typeQuadra)

% Формируем сетку (плоскость независимых переменных)

[n, N] = meshgrid(n\_start:1:n\_end, N\_start:1:N\_end);

% Формируем заготовку для матрицы ошибок, покрывающей сетку

E = n + N;

% Считаем значения X и Y в равноотстоящих точках на отрезке [a, b]

X1 = linspace(a, b, 10000); % 10000 узловых точек (a, b, 10000)

Y1 = f(X1);

for i = 1:size(E, 2) %

nn = n(1, i);

for j = 1 : size(E, 1) %

NN = N(j, 1);

try

C = coefMNKBase(-1, 2, nn, NN, typeMNK, typeQuadra);

% Считаем значения полинома в этих 10000 точках:

Y2 = Y1 \* 0;

for k = 1:length(X1)

Y2(k) = pointMNK(C, X1(k));

end

% Считаем значения ошибок

E(j, i) = deltaMNK(Y1, Y2);

catch ME

E(j, i) = E(j-1, i-1);

end

end

end

figure;

xlabel('Степень полинома, n');

ylabel('Число узловых точек, N');

zlabel('Абсолютная погрешность');

grid on; hold on;

surf(n, N, E) % Строим график

% 5 Подписываем легенду

title('Зависимость точности аппроксимации МНК от n и N', 'FontName', 'Courier');

end