**ПРАКТИЧЕСКОЕ ЗАНЯТИЕ № 4**

ПРИБЛИЖЕНИЕ ФУНКЦИЙ

ЦЕЛЬ РАБОТЫ

Целью данного практического занятия является ознакомление с основами приближения функций в среде *MatLAB*: изучение специальных функций, работа с полиномами и матрицами, построение графиков, продолжение ознакомления со встроенным языком программирования.

ВВЕДЕНИЕ

Методы аппроксимации очень важны для приложений, в которых желательно заменить относительно сложную функцию (или фактически, множество данных) простым полиномом, с которым легко обращаться. Здесь имеется выбор между очень хорошим приближением функции *f*(*x*)вблизи некоторого заданного значения *х*0 аргумента *х,* которое монотонно ухудшается по мере удаления *х* (полиномы Тейлора), и достаточно хорошим приближением в более обширной области (аппроксимация «подгонкой»). У каждого из них есть свои области применения.

## 1. Полиномы

Полиномы вида *р*(*х*) *= х4* + 2*x*3 – 3*x*2 + 4*x* + 5 вводятся как набор их коэффициентов:

» р=[1 2 -3 4 5];

Обратите внимание на *пробелы* между числами, но допустимо также использовать в качестве разделителей и запятые. Другими словами, полином будет просто *вектором,* содержащим его коэффициенты, начиная со «старшего», стоящего перед наибольшей степенью переменной *х.*

*Корни* полинома *р* (т.е. решения уравнения *р*(*х*) *=* 0) получим, выполнив

» roots(р)

Обратите внимание, что эта команда находит как вещественные, так и комплексные корни. Мы можем увеличить число отображаемых цифр с помощью

» format long

Можно получить график полинома р:

» х=-4:.05:2;

» y=polyval(p,x);

» plot(х,у)

Первая строка создает вектор *х* с элементами от -4 до 2 с шагом 0.05, т.е.

*х* = [-4.0 -3.95 -3.9... 1.95 2.0].

Как обычно, точка с запятой запрещает вывод на экран.

Вторая строка вычисляет значения полинома *р* для всех *х*, создавая вектор *у* той же длины, что и *х*, и содержащий *р*(-4),..., *р*(2).

Третья строка строит график зависимости *у* от *х*, отображая соответствующие пары чисел и соединяя их (очень короткими) отрезками прямых. Добавив

» hold on

» plot ([-4,2], [0,0], ‘k’)

нарисуем ось *х.* Команда *plot* берет первые значения из каждой квадратной скобки, образуя точку (-4,0), и соединяет ее с точкой (2,0), полученной с помощью вторых элементов из квадратных скобок. `*k*` предписывает *MatLAB* нарисовать ось черным цветом. Таким образом, на экране рисуется ось *х.* Теперь видно, что *р* имеет два вещественных корня на интервале от -4 до 2 и их положение соответствует вычисленным значениям корней -3.18... и -0.728....

## 2. Полиномы Тейлора

Хорошей иллюстрацией графических возможностей является сравнение графика такой функции, как sin *x*, и графика ее приближения полиномом Тейлора

.

*М*-файл *tsine.m*, текст которого приведен ниже, запрашивает значение *k* (это число членов полинома), затем рисует зеленым цветом «истинную» кривую и, после нажатия <*Enter*>, к этой кривой дорисовывает красным цветом ее приближение. Чтобы выполнить этот *М*-файл, достаточно набрать *tsine* с последующим <*Enter*>. Набирать .*m* не надо.

% Рисует приближение Тейлора степени 2k-1 к синусоиде

k = input('Введите число членов ряда Тейлора для синуса: ');

x = 0:.05:2\*pi;

z = sin(x);

plot(x,z,'g') % Точная кривая синуса рисуется зеленым цветом

hold on

pause % Это приостанавливает выполнение до нажатия клавиши Enter

w = x;

y = x;

s = -1;

for j = 1:k-1

w = w.\*x.\*x/(2\*j\*(2\*j+1));

y = y+s\*w;

s = -s;

end

plot(x,y,'--r')

% Приближение рисуется красным цветом

hold off

На полученном рисунке синусоида изображается сплошной линией, а приближение Тейлора для *k* = 5 (что соответствует степени 2*k* – 1 = 9) – пунктирной линией.

Проверьте, что приближение с возрастанием *k* все меньше отличается от кривой при удалении от *х* = 0. По мере приближения *k* к 10 кривые становятся более или менее неразличимыми в области от 0 до 2π.

Заметьте, что мы используем '*g*', чтобы кривая рисовалась зеленым цветом. Пунктирная линия рисуется командой *plot*(*х, у*, '--'); когда это требуется, символ «-» применяется для задания сплошной линии. Команда *hold on* позволяет сохранить график и поверх него нарисовать другой. Поэтому в конце необходимо отменить режим сохранения графика, выполнив команду *hold off*. Обратите внимание на использование в этом файле поэлементного умножения .\*.

## 3. Приближения с помощью функции *polyfit*

Другой подход к приближению функций полиномами дает использование функции *polyfit*, которая подбирает для кривой полином заданной степени по методу, минимизирующему расстояние между графиками полинома и истинной кривой, усредненное по всей их длине. Например, выполните *М*-файл с названием *polyex.m*:

a = 0:.05:2\*pi;

b = sin(a);

c = polyfit(a,b,5);

d = polyval(c,a);

plot(a,d,'r')

hold on

pause

plot(a,b,'--g')

hold off

Файл находит приближение синусоиды полиномом 5-й степени в области от 0 до 6.3, используя а в качестве независимой переменной. Полиномиальное приближение с есть просто список *коэффициентов;* чтобы вычислить его в точках а, используется функция *polyval*. Первая команда *plot* рисует красным цветом приближение, а вторая команда *plot* рисует зеленым цветом собственно синусоиду. Обратите внимание, как сильно отличается это приближение от приближения Тейлора той же степени, полученное с помощью *tsine.m* при *k* = 3.

## 4. Приближение по методу наименьших квадратов

Даны две последовательности данных ***х*** и ***у.*** Рассмотрев графики их расположения, мы можем сделать некоторые выводы о соотношении или связи между ними. Мы хотим найти функцию, которая приближенно описывает такую связь. Конечно, для этой цели функции можно выбирать различными способами, исходя из внешнего вида графиков расположения. Предположим, что такая пара множеств данных задана как (*xi, yi*)для *i* = 1, 2, ... , *n*. Удобно записать их в виде двух векторов:

***x*** = [*x1,x2,…,xn*]Т; ***у*** = [*у1,у2,…,уn*]Т.

Здесь мы их взяли как *векторы-столбцы.* Вспомним, что вектор столбец задается как x=[x1; x2; … ; xn];

### 4.1. Приближение прямой линией (по методу наименьших квадратов)

Вероятно, простейшая функция, которую стоит проверить, — это линейная,

*Y = a + bx,*

когда мы пытаемся найти два скаляра *а* и *b*, такие, что линейные соотношения

 или 

дают *Yi = yi* для всех *i.* Обозначим эту линейную систему через *А****с***= **Y**, где **с**T = (*а b*)*.* Существуют ли такие скаляры? Это переопределенная система линейных уравнений с неизвестными величинами *а* и *b,* а это означает, что в общем случае ее нельзя удовлетворить точно, каковы бы ни были *а* и *b.*

Произвольный выбор *а* и *b* даст ошибку величины |*yi – Yi|* для каждого *i,* и попытка найти приемлемое решение системы приводит к необходимости выработать какое-то соглашение относительно этих невязок. В *методе наименьших квадратов* Гаусса предлагается находить такое единственное решение *а* и *b,* которое минимизирует сумму квадратов невязок

,

где для всех *i, Yi = a + bxi*

4.1.1. Первый метод: применение оператора \

В *MatLAB*'e для заданных ***х*** и ***у*** можно легко найти наилучшее решение *а* и *b,* выполнив

» А = [ones(size(х)) х];

» с = А\у;

Здесь *А* – матрица, приведенная выше, с единицами в первом столбце и вектором ***х*** в качестве второго столбца. Вторая строка находит наилучшее приближение ***с*** решения уравнений *А****с*** *=* ***у***. Как было определено выше, ***с***(1) = *а* и ***с***(2) = *b.*

Поскольку *а* и *b* уже известны, можно воспользоваться соотношением *Y = а +bx,* чтобы найти любую точку на кривой наилучшего приближения: например, после

» Y = с(1) + с(2)\*х;

» plot(x,Y)

будет получен график кривой наилучшего приближения (по методу наименьших квадратов).

Чтобы найти невязку R2 приближения по методу наименьших квадратов, наберите

» Y = A\*c;

» r = y – Y;

» R2 = r’\*r;

4.1.2. Второй метод: применение polyfit

Для приведенных выше данных наберите

» С = polyfit(х,у,1);

чтобы найти наилучшее приближение полиномом порядка 1 (прямой). Это даст ***С***(2) = *а* и ***С***(1) *= b. Заметьте, что* ***с*** *был вектором-столбцом, а* ***С*** *является вектором-строкой!* Чтобы найти невязку R2 приближения по методу наименьших квадратов, используя вектор-строку *С*, наберите

» У = polyval(C,x);

» r = у-Y;

» R2 = r’\*r

Вспомните, что команда *polyval* вычисляет значения полинома *С* для значений *х* из вектора ***х.*** Символ «‘» обозначает функцию транспонирования матрицы. Таким образом, она находит точки кривой наилучшего приближения. Поскольку ***х*** и ***у*** являются векторами-столбцами, таковы же ***Y*** и ***r***.

### 4.2. Приближение полиномами

Если соотношение между множествами данных ***х*** и ***у*** на самом деле не является линейным, мы можем попробовать нелинейные функции. Проще всего воспользоваться квадратичной функцией

*Y = b2x2 + b1x + а.*

Мы хотим найти коэффициенты *b2, b1* и *а* так, чтобы минимизировать невязку

,

где для всех *i*

*Yi = b2xi2 + b1xi + а.*

В MatLAB'e надо просто набрать

» С = polyfit(x, у, 2)

чтобы получить коэффициенты ***С***(1) = *b2*, ***С***(2) = *b1* и ***С***(3) = *а*. Для нахождения с помощью вектора-строки ***С*** невязки метода наименьших квадратов наберите

» Y = polyval(С, х);

» r = у - Y;

» R2 = r’\*r

Набрав *help polyfit*, вы найдете, что *polyfit* может приблизить множества данных ***х*** и ***у*** полиномом любой степени *k*. В общем случае функция имеет вид

*Y* = *bkxk* +… + *b2x2 + b1x + а.*

и команда MatLAB С = polyfit(*х, у, k*) дает требуемые коэффициенты в виде вектора-строки ***С*** = [*bk*,… , *b2* , *b1* , *а*].

Теперь оцениваемые величины для всех *хi*, могут быть найдены как

» Y = polyval(С, х);

Итак, с помощью *polyfit* легко могут быть найдены полиномы высоких порядков. Но это неприменимо в случае многих переменных.

Тем, кто получил удовольствие от использования метода \, будет приятно узнать, что он применим и здесь. Воспользовавшись подсказкой при формировании множества *А*, вы получите то же решение, что и при использовании *polyfit*.

» А=[ones(size(х)) х х.^2]; %, операция .^ выполняется покомпонентно

Нелинейная взаимосвязь между множествами данных ***х*** и ***у*** не обязательно задается полиномом: она может выражаться и другими функциями, например как

*Y* = *b2* sin *х +b1* cos *x + а.*

**ЗАДАНИЕ К ПРАКТИЧЕСКОМУ ЗАНЯТИЮ**

В ходе выполнения практического занятия требуется написать *М*-файлы, которые выполняют заданную последовательность действий. Выполнение лабораторной работы в виде файла-скрипта не допускается.

### Общая часть

1. Напишите *М*-файл для вычисления заданной функции. Постройте ее график.
2. Напишите *М*-файл, дающий приближение заданной функции рядом Тейлора-Маклорена. Выполните этот файл, чтобы найти минимальное число членов *k,* необходимых для получения хорошего полиномиального приближения.
3. Для этого числа *k* напишите *М*-файл, выполняющий приближение с помощью функции *polyfit*.
4. Сравните все приближения на одном графике. Вывести легенду графика (legend()), подписать оси.
5. Для функции следующего по циклу варианта напишите *М*-файл, формирующий массивы ***x*** и ***y***, причем к массиву ***y*** должен быть добавлен шум, сгенерированный по равномерному закону распределения (функция rand()). Амплитуда шума *a = 0.1*. Осуществите полиномиальное приближение функции по методу наименьших квадратов. Оцените минимальную невязку.

### Вариант 1

*y = arctg x; |x| ≤ 1*.[[1]](#footnote-1)\*

### Вариант 2

*y = ln(1 + x); -0.999 < x ≤ 1.[[2]](#footnote-2)\*\**

### Вариант 3

*y = ; |x| < ∞.*

### Вариант 4

*y = ; |x| < ∞.*

***Примечание.*** Для выполнения практического занятия может потребоваться использование дополнительных функций или свойств объектов *MatLAB*. Для их поиска необходимо обращаться к встроенной справочной системе (*Help -> MATLAB Help*). Система содержит описания различных встроенных функций и операторов, сгруппированных по категориям. Следует помнить, что большинство объектов в *MatLAB* имеют дополнительные свойства (обращаться к которым можно при помощи функций *set* и *get*), которые также описаны в справочной системе. Так, например, для поиска свойств графиков можно во вкладке *Index* справочной системы ввести *figure* или *axes*.

1. \* В *MatLAB* функция arctg записывается как *atan*(). [↑](#footnote-ref-1)
2. \*\* Для обозначения натурального логарифма в *MatLAB* используется функция log(). [↑](#footnote-ref-2)