МИНИСТЕРСТВО ВЫСШЕГО И СРЕДНЕГО СПЕЦИАЛЬНОГО ОБРАЗОВАНИЯ РЕСПУБЛИКИ УЗБЕКИСТАН

САМАРКАНДСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ ИМЕНИ АЛИШЕРА НАВОИ

МЕХАНИКО – МАТЕМАТИЧЕСКИЙ ФАКУЛЬТЕТ

КАФЕДРА «ПРИКЛАДНАЯ МАТЕМАТИКА»

ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА
для получения степени бакалавра по направлению образования: «5130200 – Прикладная математика и информатика»

ОРИПОВОЙ НАРГИЗЫ АБДУХАКИМОВНОЙ
на тему:

ЧИСЛЕННОЕ РЕШЕНИЕ ЗАДАЧИ ИНТЕРПОЛИРОВАНИЕ ФУНКЦИИ С ПОМОЩЬЮ МАТЕМАТИЧЕСКИХ ПАКЕТОВ И ИХ ПРАКТИЧЕСКОЕ ПРИМЕНЕНИЕ

Выполнил(а) _____ **Н.А.Орипова**

Научный руководитель _____ **доц. Абдирашидов А.**

Работа обсуждена и допущена к защите протоколом №11,а заседания кафедры от 23 мая 2016 г.

Заведующий кафедрой: _____ **доц. Махмудов Ж.**

Декан факультета: _____ **доц. Х.Рузимурадов**

Работа обсуждена на заседании ГАК от «___» июня 2016 г. и протоколом №____ оценена на балл «____».

Председатель ГАК _____

Члены комиссии _____

_____

_____

Самарканд – 2016

Оглавление

ВВЕДЕНИЕ

Актуальность работы. Любому специалисту в ходе практической деятельности приходится совершать операции над количественными данными, которые осуществляются в соответствии с математическими законами. Поэтому для специалиста-нематематика наиболее важным является практический аспект математики и умение провести необходимые вычисления. Математическая теория изменяется сравнительно медленно, однако технология применения вычислительных методов претерпела более существенные изменения. В настоящее время специалист, даже хорошо знающий математику, но не умеющий применять математические методы, а именно методов вычислительной математики на компьютере, не может считаться специалистом современного уровня.

Цель работы. Настоящая работа посвящено описанию методов проведения интерполирования функции (экспериментальных данных) и их реализации с помощью пакета MATLAB. Наиболее важной отличительной особенностью предлагаемого в работе материала является рассмотрение вопросы интерполирования функции не в традиционном изложении, а с перспективой дальнейшего применения компьютера. При этом изложение материала ведется от математической постановки задач к способам их решения на компьютере.

Научная и практическая ценность работы. Существует значительное количество специализированных математических пакетов, таких как MATLAB, MathCad, Mathematica, Maple и др. Все они охватывают основные разделы математики и позволяют производить подавляющее большинство необходимых математических расчетов. Однако освоение этих пакетов самостоятельно – довольно трудоемкая задача. В то же время в курсах специализации, изучаемый в различных вузах страны, включено изучение прикладной программы по расчету в MATLAB. Поэтому пред-

ставляется оправданным описанный в данной работе подход, основанный на применении вычислительных методов интерполирования именно с помощью программы MATLAB.

Настоящая работа предназначено, в первую очередь, для студентов и магистрантов вуза и ориентировано на дальнейшее использование информационных технологий и вычислительных методов в процессе обучения выбранной специальности. Однако оно может быть рекомендовано широкому кругу пользователей персонального компьютера, сталкивающихся с необходимостью интерполирования вычислительных (экспериментальных) данных.

Структура работы. Работа состоит из введение, двух глав (7 параграфов), заключения и список литературы.

Содержание работы. Данная работа содержит основные теоретические сведения о таких вычислительных методах интерполирования, как: интерполяция функции с интерполяционными полиномами разной степени, в форме Лагранжа и Чебышева. Изложение материала осуществляется в следующей последовательности. Вначале приводятся основные определения и формулы, сходимость интерполирования, оценка точности расчета, а затем дается описание соответствующих программ MATLAB, после чего рассматриваются решения типовых примеров.

ГЛАВА 1.

ИНТЕРПОЛЯЦИЯ ФУНКЦИЙ ИНТЕРПОЛЯЦИОННЫМИ ПОЛИНОМАМИ

В вычислительной математике существенную роль играет интерполяция функций, т.е. построение по заданной функции другой (как правило, более простой), значения которой совпадают со значениями заданной функции в некотором числе точек. Причем интерполяция имеет как практическое, так и теоретическое значение. На практике часто возникает задача о восстановлении непрерывной функции по ее табличным значениям, например, полученным в ходе некоторого эксперимента. Для вычисления многих функций оказывается эффективно приблизить их полиномами или дробно-рациональными функциями (см., например [1]). Теория интерполирования используется при построении и исследовании квадратурных формул для численного интегрирования, для получения методов решения дифференциальных и интегральных уравнений.

Все перечисленные выше вопросы рассмотрены в классических учебниках по численным методам (см., например, [2-5] Ссылки в списке литературы). Цель этого раздела - демонстрация возможностей MATLAB для изучения вопросов, возникающих при интерполяции функций, в основном при помощи интерполяционных полиномов. В данном разделе приводятся необходимые сведения об интерполяции функций и при помощи небольших программ, написанных на языке пакета MATLAB, изучаются проблемы, возникающие при интерполяции функций. Простота языка пакета MATLAB в сочетании с широким набором его функций, в том числе и графических, позволяет вместо написания собственных программ интерполяции и визуализации результатов сосредоточиться на исследовании большого числа примеров, что может быть использовано при проведении лабораторных работ по численным методам для студентов технических факультетов вузов и втузов.

1.1. Задача интерполяции функции, интерполяционные полиномы

Пусть на отрезке $[a,b]$ задана функция $f(x)$. Задача интерполяции (или интерполирования) состоит в построении функции $g(x)$, совпадающей с заданной $f(x)$ в некотором наборе точек $\{x_1,x_2,...,x_{n+1}\}$ из отрезка $[a,b]$ (эти точки называются узлами интерполяции), т.е. должны выполняться условия:

$$g(x_k)=y_k,\ k=1,2,...,n+1,$$

где y_k - известные значения функции $f(x)$ в точках x_k. Функция $g(x)$ называется интерполянтом функции $f(x)$.

Пример интерполяции с четырьмя узлами приведен на рис.1.1, из которого видно, что узлы интерполяции не обязательно должны располагаться равномерно на отрезке $[a,b]$.

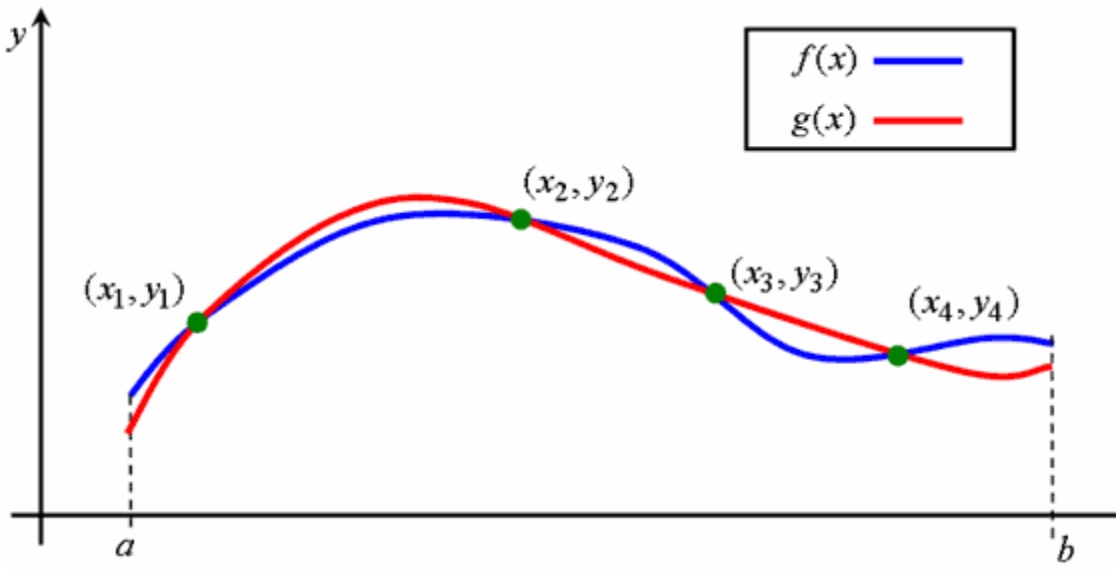

Рис.1.1. Иллюстрация интерполирование функции $f(x)$ через интерполянтом $g(x)$ с четырьмя узлами.

Если $f(x)$ табличная функция, скажем полученная из эксперимента, т.е. известны только ее значения y_k в точках x_k, то, вообще говоря, о качестве полученного приближения судить трудно. Однако, если значения $f(x)$

могут быть вычислены в любой точке отрезка [*a,b*], то в этом случае можно исследовать качество получающегося приближения, например найдя максимальное уклонение функции *g(x)* от функции *f(x)*. На качество приближения сильное влияние оказывает количество и расположение узлов, а также гладкость функции *f(x)*. Эти вопросы и будут численно исследованы в следующих разделах.

Мы рассмотрим только линейную интерполяцию, т.е. такую, при которой функция *g(x)* разыскивается в виде линейной комбинации некоторых функций

$$g(x) = \sum_{k=1}^{n+1} a_k \varphi_k(x),$$

где для *k=1,2,...,n+1*: $\varphi_k(x)$ - заданные функции, а a_k - искомые коэффициенты.

Ясно, что из постановки задачи интерполяции (т.е. из совпадения значений интерполянта *g(x)* и интерполируемой функции *f(x)* в точках x_k) следует, что коэффициенты a_k определяются из решения следующей системы линейных алгебраических уравнений:

$$\sum_{k=1}^{n+1} a_k \varphi_k(x_j) = y_j \;, j = 1,2,\ldots n+1,$$

или в развернутой форме

$$\begin{cases} a_1\varphi_1(x_1) + a_2\varphi_2(x_1) + \ldots + a_{n+1}\varphi_{n+1}(x_1) = y_1 \\ a_1\varphi_1(x_2) + a_2\varphi_2(x_2) + \ldots + a_{n+1}\varphi_{n+1}(x_2) = y_2 \\ \vdots \\ a_1\varphi_1(x_{n+1}) + a_2\varphi_2(x_{n+1}) + \ldots + a_{n+1}\varphi_{n+1}(x_{n+1}) = y_{n+1} \end{cases}$$

Совершенно ясно, почему число коэффициентов a_k должно совпадать с числом узлов интерполяции x_k. Это нужно для того, чтобы матрица системы была квадратной (т.е. число неизвестных совпадало бы с числом условий, из которых находятся эти неизвестные). Кроме того, для однозначной разрешимости данной системы (при произвольной правой части) необходимо и достаточно, чтобы ее определитель был отличен от нуля, т.е.

$$\begin{vmatrix} \varphi_1(x_1) & \varphi_2(x_1) & \dots & \varphi_{n+1}(x_1) \\ \varphi_1(x_2) & \varphi_2(x_2) & \dots & \varphi_{n+1}(x_2) \\ \vdots & \vdots & \ddots & \vdots \\ \varphi_1(x_{n+1}) & \varphi_2(x_{n+1}) & \cdots & \varphi_{n+1}(x_{n+1}) \end{vmatrix} \neq 0$$

Очень часто в качестве системы функций $\varphi_k(x)$ выбирают полиномы, например степени x, именно:

$$\varphi_1(x)=1, \quad \varphi_2(x)=x, \quad \varphi_3(x)=x^2, \quad \dots, \quad \varphi_{n+1}(x)=x^n.$$

Тогда соответствующий интерполянт называют интерполяционным полиномом. Существование и единственность интерполяционного полинома гарантируется, если все узлы интерполяции x_k различны. Действительно, определитель системы линейных алгебраических уравнений для нахождения коэффициентов a_k

$$\begin{vmatrix} x_1^0 & x_1^1 & \dots & x_1^n \\ x_2^0 & x_2^1 & \dots & x_2^n \\ \vdots & \vdots & \ddots & \vdots \\ x_{n+1}^0 & x_{n+1}^1 & \cdots & x_{n+1}^n \end{vmatrix}$$

является определителем Вандермонда, который, как известно, равен

$$\prod_{\substack{j=1 \\ i>j}}^{n+1} \left(x_i - x_j\right)$$

и, следовательно, отличен от нуля в случае, когда все узлы x_k различны. Поскольку матрица системы невырождена, то решение системы существует и единственно. Итак, интерполяционный полином существует и единственный.

Можно рассуждать и по-другому. Предположим, что есть два интерполяционных полинома g_k и h_k степени nтакие, что для произвольного набора значений выполняются равенства $g(x_k) = h(x_k) = y_k)$ для всех$k=1,2,...,n+1$, т.е. для $n+1$-ой точки. Тогда их разность h_k - g_k является полиномом степени не выше n, но обращается в ноль в $n+1$-ой точке. По известной теореме алгебры у полинома степени n не может быть больше

чем n корней, следовательно h_k - $g_k \equiv 0$ и $h_k \equiv g_k$. Единственность установлена. А так как полином единственный, то у соответствующей системы линейных алгебраических уравнений есть только одно решение (для произвольной правой части). Из результатов линейной алгебры известно, что у системы может быть либо бесконечное число решений при некоторых правых частях, либо единственное, для произвольной правой части. Последнее как раз и выполняется.

Итак, число узлов интерполяционного полинома всегда должно быть на единицу больше его степени. Это понятно также из следующих простых соображений: через две точки проходит единственная прямая, через три - единственная парабола и т.д. Полином может получиться и степени меньшей, чем, например, если три точки лежат на одной прямой, то через них проходит единственный полином первой степени, однако это не нарушает наших рассуждений (просто коэффициент при старшей степени равен нулю). Однако, существует бесконечно много парабол, проходящих через две точки.

Казалось бы, при практической реализации интерполяционного процесса коэффициенты интерполяционного полинома a_k можно найти, непосредственно решая систему линейных алгебраических уравнений

$$\begin{cases} a_1 x_1^0 + a_2 x_1^1 + \ldots + a_{n+1} x_1^n = y_1 \\ a_1 x_2^0 + a_2 x_2^1 + \ldots + a_{n+1} x_2^n = y_2 \\ \vdots \\ a_1 x_{n+1}^0 + a_2 x_{n+1}^1 + \ldots + a_{n+1} x_{n+1}^n = y_{n+1} \end{cases}$$

каким-нибудь численным методом. Однако, у такого подхода есть существенный недостаток. Число обусловленности матрицы этой системы быстро растет с ростом числа узлов интерполяции, что может привести к большим ошибкам при решении системы с ней.

Установим зависимость числа обусловленности этой матрицы от числа узлов интерполяции, предполагая, что они распределены равномерно на

отрезке [-1,1]. Для этого напишем небольшую программу в MATLAB, в которой в цикле по числу узлов генерируются матрицы, находится их число обусловленности при помощи функции cond и строится график зависимости числа обусловленности матрицы от количества узлов.

Программа № 1.1 в MATLAB

Зависимость числа обусловленности матрицы от количества узлов интерполяции

```
% задание границ отрезка
a=-1; b=1;
% задание количества узлов
N=1:30;
% инициализируем массив (пока пустой) для записи чисел обусловленности
C=[];
% проходим по узлам в цикле
for n=N
   % формируем матрицу
   x=linspace(a,b,n)';
   M=[];
   for k=0:n
      M=[M x.^k];
   end
   % считаем число обусловленности для текущего числа узлов и записываем в массив C
   C=[C cond(M)];
end
% выводим график зависимости числа обусловленности от числа узлов
% в полулогарифмическом масштабе
```

```
figure
semilogy(N,C)
grid on
```

В результате получаем зависимость числа обусловленности матрицы от количества узлов интерполяции, приведенную на рис.1.2 (масштаб по оси ординат логарифмический).

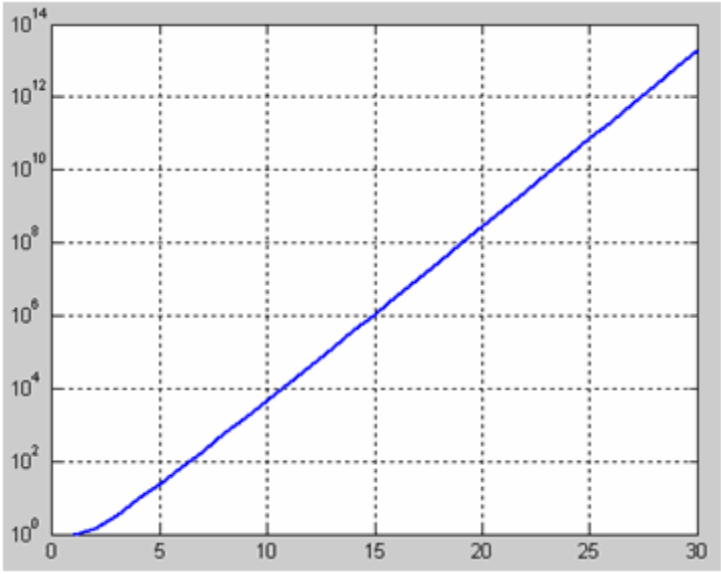

Рис.1.2. Зависимость числа обусловленности матрицы от количества узлов интерполяции.

Из-за плохой обусловленности матрицы применяют другие подходы для построения интерполяционных полиномов, которые также требуют также меньшего числа операций по сравнению с решением системы, имеющим вычислительную сложность порядка n^3 операций.

Важно, что какие бы подходы для построения интерполяционного полинома не применялись, они всегда должны привести к одинаковому результату (если все вычисления проводятся точно, а не на компьютере), поскольку интерполяционный полином степени существует и единственный при различных n-ом узлах интерполяции. Другое дело, что разные способы построения интерполяционного полинома могут обладать разными вычис-

лительными свойствами. Рассмотрим сначала интерполяционный полином в форме Лагранжа. Далее мы будем использовать обозначение для интерполяционного полинома в зависимости от способа его построения.

1.2. Интерполяционный полином в форме Лагранжа

Вывод формулы. Итак, мы ищем полином $L_n(x)$ степени не выше n, значения которого совпадают со значениями y_k заданной функции $f(x)$ в узлах x_k, где $k=1,2,...,n+1$ и все узлы различны.

Одним из способов записи интерполяционного полинома является форма Лагранжа. Предположим, что для $k=1,2,...,n+1$ функции $\Phi_n(x)$ являются полиномами степени n, которые обладают следующим свойством

$$\Phi_k(x_j) = \begin{cases} 1, & k = j \\ 0, & k \neq j \end{cases}.$$

Тогда полином

$$L_n(x) = \sum_{k=1}^{n+1} y_k \Phi_k(x)$$

будет как раз тем, который нам и нужен, поскольку это полином степени не выше n и $L_n(x_k) = y_k$ для всех $k=1,2,...,n+1$.

Функции $\Phi_n(x)$ строятся легко. Действительно, функция

$$(x-x_1)\cdot(x-x_2)\cdot\ldots\cdot(x-x_{k-1})\cdot(x-x_{k+1})\cdot\ldots\cdot(x-x_n)\cdot(x-x_{n+1})$$

является полиномом степени n, который обращается в ноль для всех x_j не равных x_k. В точке x_k она принимает значение

$$(x_k-x_1)\cdot(x_k-x_2)\cdot\ldots\cdot(x_k-x_{k-1})\cdot(x_k-x_{k+1})\cdot\ldots\cdot(x_k-x_n)\cdot(x_k-x_{n+1}).$$

Тогда

$$\Phi_k(x) = \frac{(x-x_1)\cdot(x-x_2)\cdot\ldots\cdot(x-x_{k-1})\cdot(x-x_{k+1})\cdot\ldots\cdot(x-x_n)\cdot(x-x_{n+1})}{(x_k-x_1)\cdot(x_k-x_2)\cdot\ldots\cdot(x_k-x_{k-1})\cdot(x_k-x_{k+1})\cdot\ldots\cdot(x_k-x_n)\cdot(x_k-x_{n+1})},$$

или, что то же самое:

$$\Phi_k(x) = \prod_{\substack{j=1,2,\ldots,n+1 \\ j\neq k}} \frac{x - x_j}{x_k - x_j}.$$

Для примера построим графики функций $\Phi_n(x)$ для пяти узлов, равномерно распределенных на отрезке [0,1], для чего можно воспользоваться следующей небольшой программой.

Программа № 1.2 в MATLAB

Построение графика функции с помощью интерполяционного полинома в форме Лагранжа

```
% задаем границы отрезка и значение n
a=0; b=1; n=4
% генерируем набор из n+1 точек, равномерно распределенных на [0,1]
x=linspace(a,b,n+1);
% генерируем вспомогательный набор из 1000 точек, равномерно распределенных на [0,1]
% для вычисления в них функций Psi_k для построения их графиков
xx=linspace(a,b,1000);
% создаем графическое окно и оси, наносим сетку
figure
axes('NextPlot','add')
grid on
% в цикле вычисляем значения каждой функции Psi_k в точках, записанных в xx
% и строим графики функции Psi_k
for k=1:n+1
   Phi=ones(size(xx));
   J=[1:k-1 k+1:n+1];
   for j=J
      Phi=Phi.*(xx-x(j))/(x(k)-x(j));
```

```
    end
    plot(xx,Phi,'Color',[rand rand rand],'LineWidth',2)
end
% отображаем узлы интерполяции красными маркерами
plot(x,zeros(size(x)),'LineStyle','none','Marker','.','Color','r','MarkerSize',20)
```

На рис.1.3 приведены полученные графики $\Phi_n(x)$ для $k=1,2,3,4,5$, узлы отмечены красными маркерами. Видно, что каждая из функций $\Phi_n(x)$ равна единице только в одном из узлов интерполяции, а в остальных она равна нулю.

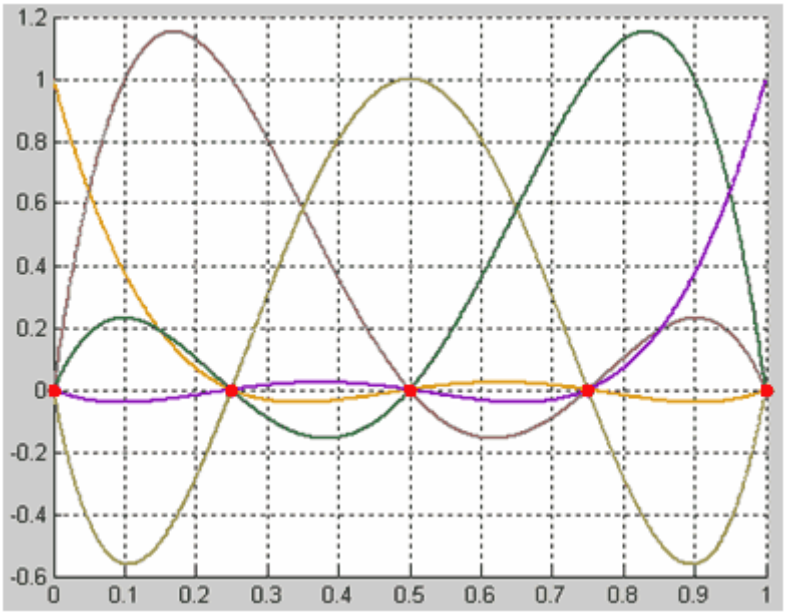

Рис.1.3. Графики $\Phi_n(x)$ для $k=1,2,3,4,5$.

Итак, интерполяционный полином в форме Лагранжа имеет вид

$$L_n(x) = \sum_{k=1}^{n+1} y_k \prod_{\substack{j=1,2,\ldots,n+1 \\ j \neq k}} \frac{x - x_j}{x_k - x_j}$$

или в развернутой форме:

$$L_n(x) = \sum_{k=1}^{n+1} y_k \frac{(x-x_1)\cdot(x-x_2)\cdot\ldots\cdot(x-x_{k-1})\cdot(x-x_{k+1})\cdot\ldots\cdot(x-x_n)\cdot(x-x_{n+1})}{(x_k-x_1)\cdot(x_k-x_2)\cdot\ldots\cdot(x_k-x_{k-1})\cdot(x_k-x_{k+1})\cdot\ldots\cdot(x_k-x_n)\cdot(x_k-x_{n+1})}$$

1.3. Вычисление интерполяционного полинома в форме Лагранжа

Несложно найти количество операций, необходимых для вычислении значения интерполяционного полинома в какой-либо точке. Если вычислять интерполяционный полином прямо по формуле

$$L_n(x) = \sum_{k=1}^{n+1} y_k \prod_{\substack{j=1,2,\ldots,n+1 \\ j \neq k}} \frac{x - x_j}{x_k - x_j}$$

то для вычисления каждого из $n+1$-го произведений

$$\prod_{\substack{j=1,2,\ldots,n+1 \\ j \neq k}} \frac{x - x_j}{x_k - x_j}$$

требуется n делений, n-1 умножение и $2n$ сложений (вычитаний). Далее, когда произведения найдены, для вычисления суммы требуется $n+1$ умножений и n сложений. Т.е. получается, что для каждого значения x вычисление $L_n(x)$ требует порядка n^2 операций.

Если требуется найти значения интерполяционного полинома в достаточно большом числе точек, то можно поступить более эффективным способом. Сначала вычислить все значения

$$z_k = \frac{y_k}{\prod_{\substack{j=1,2,\ldots,n+1 \\ j \neq k}} (x_k - x_j)}$$

за $n-1$ умножений, n сложений и 1 деление (эта операция делается для данного интерполяционного полинома только один раз перед вычислением его значений в точках), а затем находить значения интерполяционного полинома по формуле

$$L_n(x) = w(x)s(x),$$

где

$$w(x) = \prod_{k=1}^{n+1} (x - x_k), \qquad s(x) = \sum_{k=1}^{n+1} \frac{z_k}{x - x_k}.$$

При таком способе вычисления значения интерполяционного полинома в точке требуется предварительная работа порядка n^2 операций, зато для каждого значения x время вычисления интерполяционного полинома требует порядка n операций. В знаменателе в выражении для функции $s(x)$ находится разность $x - x_k$, однако при $x = x_k$ значение интерполяционного полинома вычислять не требуется, оно известно и равно y_k.

Для вычисления интерполяционного полинома в MATLAB можно применить оба вышеописанных способа. Ниже приведены тексты функций, в которых реализованы эти способы:

- в функции lagrange - первый способ;
- в функции lagrangef - второй способ;

Входные массивы x и y должны содержать значения x_k и y_k, соответственно, для всех k=1,2,...,n+1. Входной аргумент xx функций lagrange и lagrangef может быть массивом значений аргумента, для которых требуется вычислить значение интерполяционного полинома. Тогда в выходном аргументе yy вернется массив соответствующих значений полинома. При программировании функций lagrange и lagrangef не потребовалось делать цикла по элементам массива xx благодаря поддержке поэлементных операции при работе с массивами в MATLAB.

ГЛАВА 2.
СХОДИМОСТЬ И ОЦЕНКА ОШИБКИ ИНТЕРПОЛИРОВАНИЯ

2.1. Примечание (про поэлементные операции).

Для массивов в MATLAB допустимы поэлементные умножение, деление и возведение в степень. Для этих операций применяются, соответственно, следующие сочетания (с точкой): .* , ./ , .^. Приведем несколько примеров поэлементных операций с векторами:

```
>> a=[10 20 30];
>> b=[2 4 6];
>> c=a./b
c =
    5     5     5
>> d=a.*b
d =
   20    80   180
```

При программировании интерполяционного полинома по второму способу в функции lagrangef не делалась проверка на равенство x какому-либо узлу, поскольку в MATLAB операция деления на ноль допустима (при делении на ноль числа не равного нулю получается Inf, а при делении нуля на ноль получается NaN, т.е Not a Number, не число).

В качестве тестового примера проинтерполируем функцию $\sin(x)$ на отрезке [1,9] с шагом 2 и построим графики $\sin(x)$ и полученного интерполяционного полинома. В данном случае будет 5 узлов и, следовательно, интерполяционный полином получится 4-ой степени. Для построения графика интерполяционного полинома вычислим его значения при помощи

функции lagrange в 1000 равномерно отстоящих друг от друга точках на отрезке [1,9].

Программа № 2.1 в MATLAB

Построение графика функции sin(*x*) *с помощью интерполяционным полиномом 4-ой степени в форме Лагранжа*

Процедура-функция lagrange :

```
function yy=lagrange(x,y,xx)
% вычисление интерполяционного полинома в форме Лагранжа
% x - массив координат узлов
% y - массив значений интерполируемой функции
% xx - массив значений аргумента, для которых надо вычислить значения полинома
% yy - массив значений полинома в точках xx
% узнаем число узлов интерполяции (N=n+1)
N=length(x);
% создаем нулевой массив значений интерполяционного полинома
yy=zeros(size(xx));
% в цикле считаем сумму по узлам
for k=1:N
   % вычисляем произведения, т.е. функции Psi_k
   t=ones(size(xx));
   for j=[1:k-1, k+1:N]
      t=t.*(xx-x(j))/(x(k)-x(j));
   end
   % накапливаем сумму
   yy = yy + y(k)*t;
end
```

Процедура-функция lagrangef :

```
function yy=lagrangef(x,y,xx)
% вычисление интерполяционного полинома в форме Лагранжа
% x - массив координат узлов
% y - массив значений интерполируемой функции
% xx - массив значений аргумента, для которых надо вычислить значения полинома
% yy - массив значений полинома в точках xx
% узнаем число узлов интерполяции (N=n+1)
N=length(x);
% предварительное вычисление значений z_k
z=zeros(size(x));
for k=1:N
   t=1;
   for j=[1:k-1 k+1:N]
      t=t*(x(k)-x(j));
   end
   z(k)=y(k)/t;
end
% вычисление w(x)
w=ones(size(xx));
for k=1:N
   w=w.*(xx-x(k));
end
s=zeros(size(xx));
% вычисление s(x)
for k=1:N
   s=s+z(k)./(xx-x(k));
end
```

```
% вычисление значений интерполяционного полинома
yy=w.*s;
```

Основная программа :

```
% задание узлов интерполяции
x=1:2:9;
y=sin(x);
% задание точек, в которых требуется найти значения интерполяционного полинома
xx=linspace(1,9,1000);
% нахождение значений интерполяционного полинома
yy=lagrange(x,y,xx);
% построение графиков
figure('Color','w')
% вывод графика sin(x)
fplot(@sin,[1 9])
hold on
% вывод графика полинома
plot(xx,yy,'r')
% вывод узлов интерполяции
plot(x,y,'bo')
% размещение легенды
legend('sin\itx','{\itL_n}({\itx})','nodes',-1)
```

Получающийся результат приведен на рисунке ниже (рис.2.1).

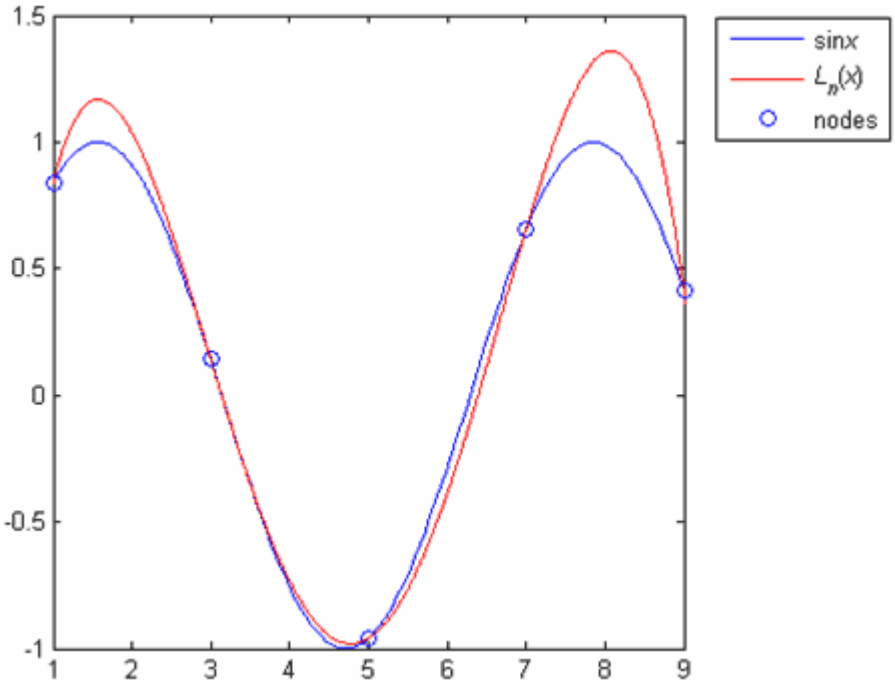

Рис.2.1. Интерполяция функции sin(x) полиномом 4-ой степени.

Хорошим тестовым примером для функций lagrange и lagrangef является приближение полиномиальной функции. Выберем в качестве интерполируемой функции полином пятой степени

$$p_5(x) = 4x^5 - 3x^4 + 14x^3 - 22x^2 - x + 5$$

и проинтерполируем его полиномом пятой степени $L_5(x)$ на отрезке [-1, 1.5]. Для интерполяции понадобится шесть узлов, мы выберем их равноотстоящими на этом отрезке и проведем интерполяцию при помощи интерполяционного полинома в форме Лагранжа, вычисляемого приведенной выше функцией lagrange. Мы выведем не только график исходного полинома $p_5(x)$ и график интерполяционного полинома $L_5(x)$, но и график ошибки, т.е. разности $p_5(x) - L_5(x)$.

Программа № 2.2 в MATLAB

Построение графика полиномиальной функции пятой степени с помощью интерполяционным полиномом пятой степени в форме Лагранжа

```
% выбор 6 равноотстоящих узлов на отрезке [-1, 1.5]
x=linspace(-1,1.5,6);
% задание inline-функции для вычисления полинома
p5=inline('4*x.^5 - 3*x.^4 + 14*x.^3 - 22*x.^2 - x + 5');
% вычисление полинома p5 в узлах интерполяции
y=p5(x);
% задание 10000 равноотстоящих точек на отрезке [-1, 1.5] для вычисления
в них значений
% интерполяционного полинома
xx=linspace(-1,1.5,10000);
% вычисление значений интерполяционного полинома
yy=lagrange(x,y,xx);
% графический вывод результатов
figure('Color','w')
subplot(2,1,1)
% вывод графика исходного полинома p5
fplot(p5,[-1 1.5])
hold on
% вывод графика интерполяционного полинома
plot(xx,yy,'r')
% вывод легенды
legend('{\itp}_5({\itx})','{\itL}_5({\itx})',-1)
subplot(2,1,2)
% вывод графика распределения ошибки на отрезке [-1, 1.5]
plot(xx,p(xx)-yy)
```

```
% вывод заголовка к графику для ошибки
title('error = {\itp}_5({\itx}) - {\itL}_5({\itx})')
```

В результате получаем, что график интерполяционного полинома пятой степени практически совпадает с графиком исходного полинома, однако ошибка вовсе не равна нулю. Она имеет порядок 10^{-14}, что объясняется ошибками округлений при вычислениях значений интерполяционного полинома. Более того, график распределения ошибки имеет характерный вид - в середине интервала ошибка меньше, чем по краям (рис.2.2).

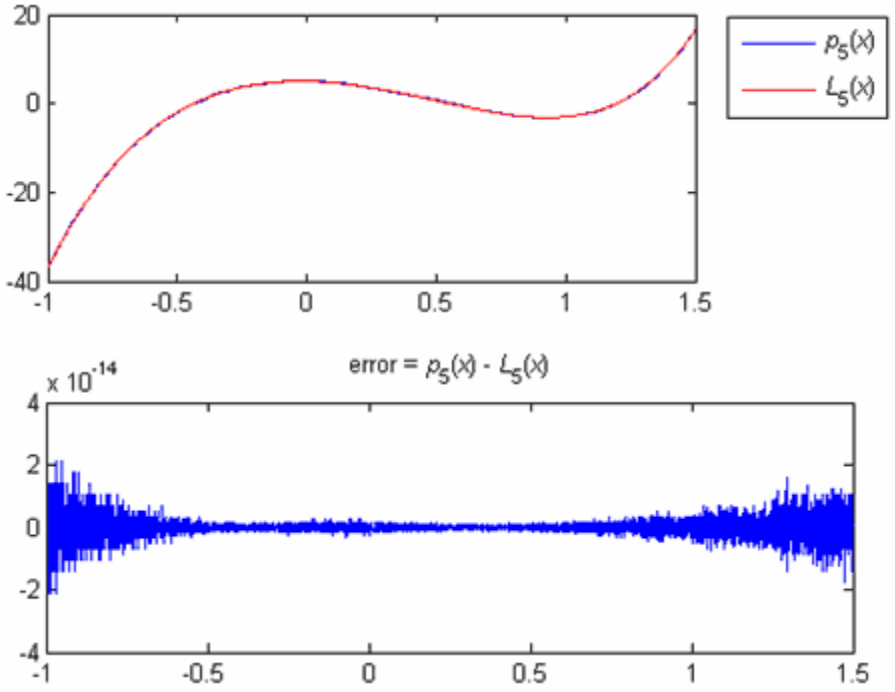

Рис.2.2. Интерполяция полинома пятой степени по шести узлам

Подробно о поведении ошибки при увеличении числа узлов и при различных способах выбора узлов интерполирования а так же для функций различной гладкости написано в разделе Сходимость интерполяционного полинома.

Проведем еще ряд экспериментов. Приблизим полином пятой степени интерполяционным полиномом с числом узлов меньше шести и с числом узлов, значительно большим, чем 6.

Сначала проинтерполируем полином

$$p_5(x) = 4x^5 - 3x^4 + 14x^3 - 22x^2 - x + 5$$

на отрезке [-1, 1.5] с тремя равноотстоящими узлами и получим интерполяционный полином $L_2(x)$ второй степени, затем с четырьмя и пятью равноотстоящими узлами и получим, соответственно, полиномы $L_3(x)$ и $L_4(x)$третьей и четвертой степени.

Программа № 2.3 в MATLAB

Интерполирование полинома пятой степени с тремя, четырьмя и пятью узлами с помощью интерполяционным полиномом в форме Лагранжа

```
% задание inline-функции для вычисления полинома
p5=inline('4*x.^5 - 3*x.^4 + 14*x.^3 - 22*x.^2 - x + 5');
% задание трех равноотстоящих узлов на [-1, 1.5]
x3=linspace(-1,1.5,3);
% вычисление в них исходного полинома p5(x)
y3=p5(x3);
% задание четырех равноотстоящих узлов на [-1, 1.5]
x4=linspace(-1,1.5,4);
% вычисление в них исходного полинома p5(x)
y4=p5(x4);
% задание пяти равноотстоящих узлов на [-1, 1.5]
x5=linspace(-1,1.5,5);
% вычисление в них исходного полинома p5(x)
y5=p5(x5);
```

```
% задание 10000 узлов на [-1, 1.5] для вычисления в них значений интер-
поляционных полиномов
xx=linspace(-1,1.5,10000);
% вычисление в них значений интерполяционных полиномов 2-ой, 3-ей и
4-ой степени
yy3=lagrange(x3,y3,xx);
yy4=lagrange(x4,y4,xx);
yy5=lagrange(x5,y5,xx);
% графический вывод результатов
figure('Color','w')
% построение графика исходного полинома p5(x)
fplot(p5,[-1 1.5],'r')
hold on
% построение графиков интерполяционных полиномов 2-ой, 3-ей и 4-ой
степени
plot(xx,yy3,'k')
plot(xx,yy4,'b')
plot(xx,yy5,'g')
% вывод легенды
legend('{\itp}_5({\itx})','{\itL}_2({\itx})',...
   '{\itL}_3({\itx})','{\itL}_4({\itx})',-1)
```

В результате получаем вполне предсказуемый результат, чем ближе степень интерполяционного полинома к 5-ой, тем лучше приближение (рис.6).

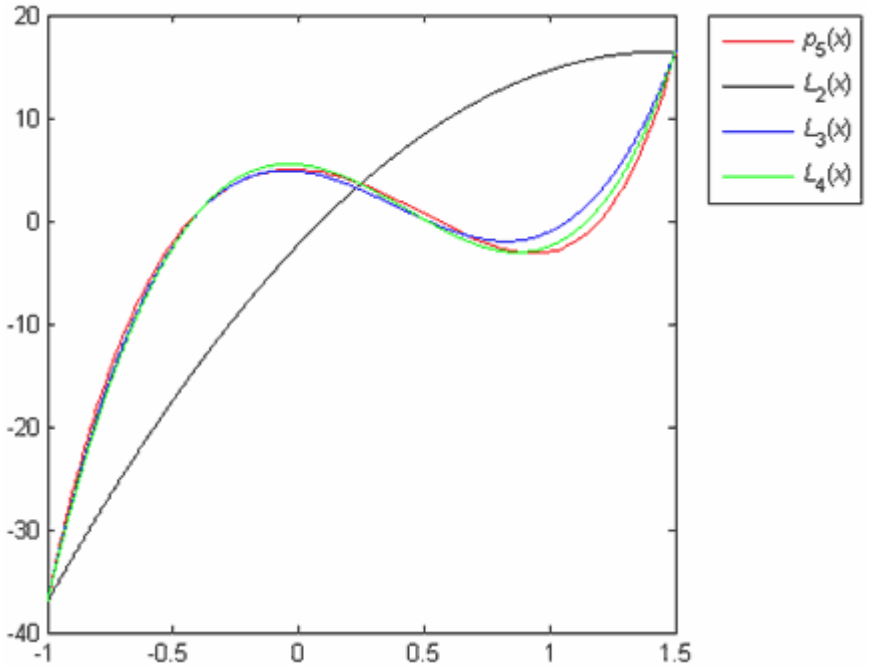

Рис.2.3. Интерполирование полинома пятой степени с тремя, четырьмя и пятью узлами.

Теперь возьмем интерполяционные полиномы достаточно высокой степени и проследим за ошибкой интерполирования. Для примера проинтерполируем полином

$$p_5(x) = 4x^5 - 3x^4 + 14x^3 - 22x^2 - x + 5$$

на отрезке [-1, 1.5] с десятью, двадцатью и тридцатью равноотстоящими узлами и получим, интерполяционные полиномы $L_9(x)$, $L_{19}(x)$ и $L_{29}(x)$, соответственно, девятой, девятнадцатой и двадцать девятой степени.

Программа № 2.4 в MATLAB

Интерполирование полинома пятой степени равностоящими узлами с помощью интерполяционным полиномом высокой степени в форме Лагранжа

```
% задание inline-функции для вычисления полинома
p5=inline('4*x.^5 - 3*x.^4 + 14*x.^3 - 22*x.^2 - x + 5');
% задание десяти равноотстоящих узлов на [-1, 1.5]
x10=linspace(-1,1.5,10);
```

```
% вычисление в них исходного полинома p5(x)
y10=p5(x10);
% задание двадцати равноотстоящих узлов на [-1, 1.5]
x20=linspace(-1,1.5,20);
% вычисление в них исходного полинома p5(x)
y20=p5(x20);
% задание тридцати равноотстоящих узлов на [-1, 1.5]
x30=linspace(-1,1.5,30);
% вычисление в них исходного полинома p5(x)
y30=p5(x30);
% задание 10000 узлов на [-1, 1.5] для вычисления в них значений интерполяционных полиномов
xx=linspace(-1,1.5,10000);
% вычисление в них значений интерполяционных полиномов 9-ой, 19-ой и 29-ой степени
yy10=lagrange(x10,y10,xx);
yy20=lagrange(x20,y20,xx);
yy30=lagrange(x30,y30,xx);
% графический вывод результатов
figure('Color','w')
% вывод ошибки интерполирования для интерполяционного полинома 9-ой степени
subplot(3,1,1)
plot(xx,yy10-p5(xx),'b')
title('error = {\itp}_5({\itx}) - {\itL}_{9}({\itx})')
% вывод ошибки интерполирования для интерполяционного полинома 19-ой степени
subplot(3,1,2)
plot(xx,yy20-p5(xx),'g')
```

```
title('error = {\itp}_5({\itx}) - {\itL}_{19}({\itx})')
% вывод ошибки интерполирования для интерполяционного полинома 29-ой степени
subplot(3,1,3)
plot(xx,yy30-p5(xx),'r')
title('error = {\itp}_5({\itx}) - {\itL}_{29}({\itx})')
```

В результате мы видим, что ошибки округлений при вычислении значений интерполяционного полинома приводят к тому, что ошибка интерполирования растет (порядка 10^{-14} для интерполяционного полинома 9-ой степени, порядка 10^{-11} для интерполяционного полинома 19-ой степени и порядка 10^{-9} для интерполяционного полинома 29-ой степени), хотя она должна была бы равняться нулю (при вычислениях в точной арифметике). Заметим, что характер распределения ошибки сохраняется - чем ближе к границам отрезка интерполирования, тем ошибка больше (рис.2.4).

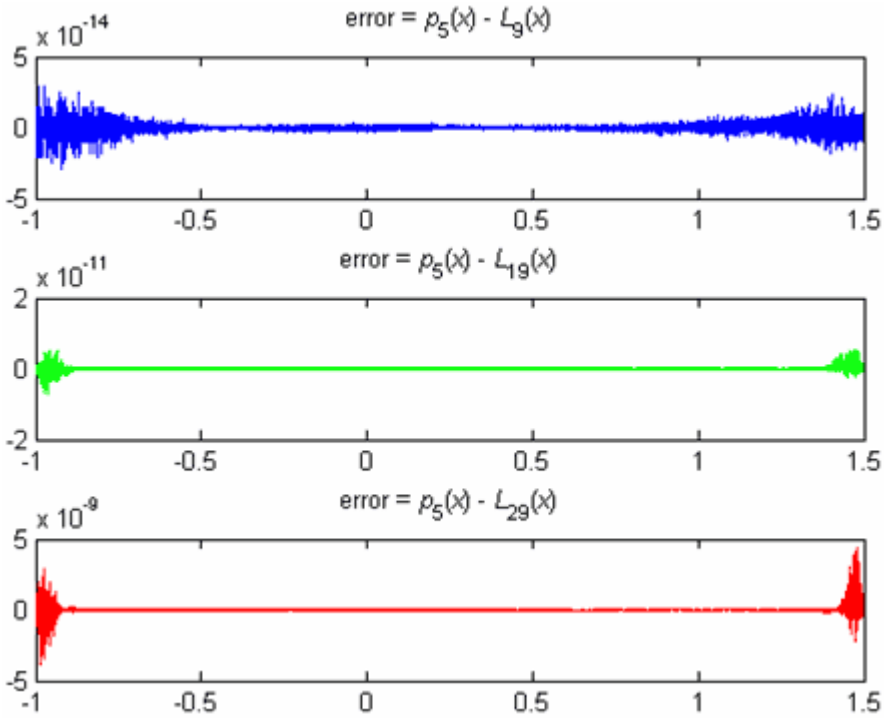

Рис.2.4. Поведение ошибки для интерполяционных полиномов $L_9(x)$, $L_{19}(x)$, $L_{29}(x)$

Вычислительные свойства интерполяционного полинома делают его неприменимым при достаточно большом числе узлов интерполирования. Изучим зависимость максимального уклонения интерполяционного полинома от интерполируемой функции снова на примере полинома

$$p_5(x) = 4x^5 - 3x^4 + 14x^3 - 22x^2 - x + 5$$

который мы будем интерполировать полиномами на отрезке [-1, 1.5] с числом узлов интерполирования, изменяющимся от двух до ста. Т.е. наша цель - получить график зависимости

$$\max_{x\in[-1,1.5]} \left|p_5(x) - L_n(x)\right|$$

от степени интерполяционного полинома *n*. Для этого в цикле по числу узлов интерполяции будем строить интерполяционный полином и вычислять максимальное уклонение его от $p_5(x)$, определяя это уклонение по 1000 равномерно отстоящим точкам на отрезке [-1, 1.5].

Программа № 2.5 в MATLAB

Интерполирование полинома пятой степени достаточно большими числами узлов с помощью интерполяционным полиномом высокой степени в форме Лагранжа

```
% задание inline-функции для вычисления полинома
p5=inline('4*x.^5 - 3*x.^4 + 14*x.^3 - 22*x.^2 - x + 5');
% инициализация массива для записи ошибок интерполяции
err=[];
% задание массива с количеством узлов интерполяции
K=2:100;
% построение в цикле интерполяционных полиномов и вычисление ошибки интерполяции
for k=K
  % задание k равноотстоящих на отрезке [-1, 1.5] узлов интерполяции
```

```
    x=linspace(-1,1.5,k);
    % вычисление в них значений интерполяционного полинома
    y=p5(x);
    % задание 1000 равноотстоящих на отрезке [-1, 1.5] точек
    xx=linspace(-1,1.5,1000);
    % нахождение в них значений интерполяционного полинома
    yy=lagrange(x,y,xx);
    % вычисление ошибки интерполяции
    e=max(abs(p5(xx)-yy));
    % добавление ее в массив
    err=[err e];
end
% графический вывод результатов
figure('Color','w')
% построение зависимости ошибки от степени интерполяционного полинома
semilogy(K-1,err,'LineWidth',2)
% нанесение сетки
grid on
% добавление заголовка
title('Dependence of the max|{\itL_n}({\itx})-{\itp}_5({\itx}))| on {\itn}')
```

В результате работы этой программы мы видим (см. рис.8), что сначала ошибка уменьшается и достигает своего минимума (когда степень интерполяционного полинома равна 5). Как уже обсуждалось выше, она не равна в точности нулю из-за ошибок округления при вычислении интерполяционного полинома. Далее, при повышении степени интерполяционного полинома, ошибка начинает быстро возрастать, что свидетельствует о практической неприменимости интерполяционных полиномов высоких степеней.

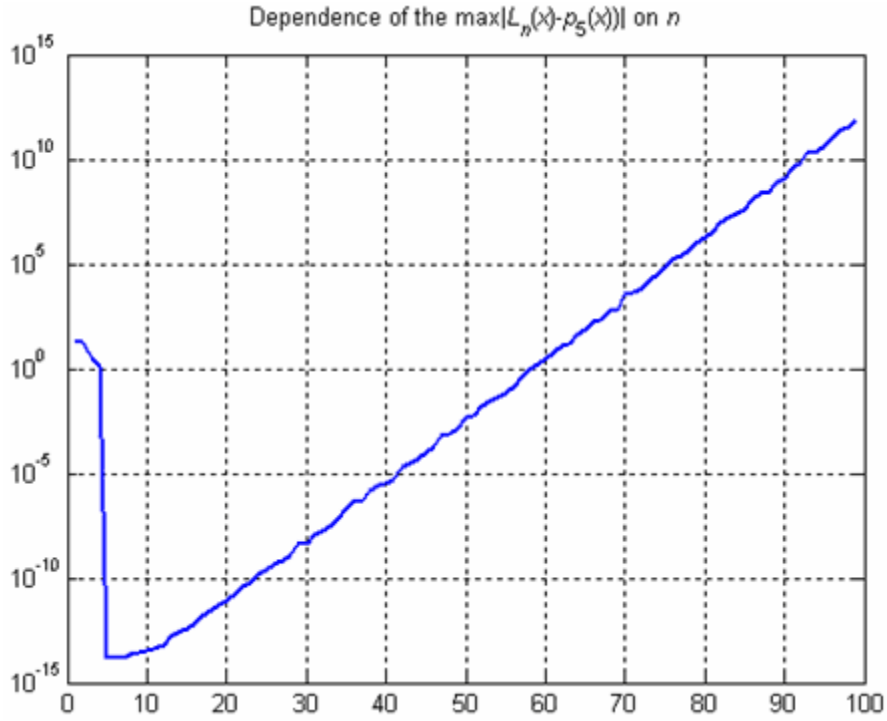

Рис.2.5 Зависимость ошибки интерполяции от степени интерполяционного полинома

Аналогичным образом ошибки округления ведут себя и при интерполировании других гладких функций, не обязательно полиномиальных, а так же и при интерполировании самой простой функции - константы.

2.2. Классические теоремы о сходимости интерполяционных полиномов

Приведем несколько классических теорем о сходимости интерполяционных полиномов, в которых имеется ввиду, что интерполяция осуществляется на бесконечной последовательности, состоящей из наборов узлов интерполирования (интерполяционных сеток):

$$x_1^{(1)}$$
$$x_1^{(2)},\ x_2^{(2)}$$
$$x_1^{(3)},\ x_2^{(3)},\ x_3^{(3)}$$
$$\dots$$
$$x_1^{(m)},\ x_2^{(m)},\ x_3^{(m)},\dots,x_m^{(m)}$$
$$\dots$$

Здесь верхний индекс обозначает номер интерполяционной сетки, а нижний - номер узла.

На первый взгляд кажется, что чем больше узлов в сетке с увеличением $f(x)$, тем все ближе и ближе интерполяционный полином будет к интерполируемой функции. Однако, как утверждает теорема Фабера, это не так.

Теорема (Фабер). Для любой последовательности интерполяционных сеток найдется некоторая непрерывная функция $f(x)$, для которой последовательность соответствующих интерполяционных полиномов не сходится равномерно к $f(x)$.

С другой стороны, теорема Марцинкевича утверждает, что для заданной функции можно подобрать хорошую последовательность сеток.

Теорема (Марцинкевич). Для каждой непрерывной функции $f(x)$ существует последовательность интерполяционных сеток, такая, что построенные по этой последовательности интерполяционные полиномы равномерно сходятся к функции $f(x)$.

Для целых функций, т.е. таких, которые в любой точке представить в виде сходящегося бесконечного степенного ряда, ситуация намного лучше, как утверждает следующая теорема.

Теорема. Если $f(x)$ целая функция, то тогда последовательность интерполяционных полиномов, построенных по любой последовательности сеток с узлами из отрезка $[a,b]$, сходится равномерно к $f(x)$ на $[a,b]$.

Следует заметить, что существование всех производных у функции $f(x)$ недостаточно для того, чтобы утверждение этой теоремы было верным. Обычно приводят пример, выбирая в качестве интерполируемой функции следующую кусочно-заданную функцию на отрезке [-1,1]:

$$f(x)=\begin{cases} e^{-1/x^2}, & x\geq 0 \\ 0, & x<0 \end{cases}.$$

Очевидно, что функция $f(x)$ непрерывна вместе со всеми своими производными на отрезке [-1,1] (да и вообще для любого x). Однако, если

брать такую последовательность интерполяционных сеток, что все ее узлы принадлежат отрезку [-1,0], то тогда всегда $L_n(x) \equiv 0$ и для любого $x > 0$ никакой сходимости не будет.

Интерполяция по чебышевским узлам, рассмотренная в разделе Ошибка интерполяции, чебышевские узлы, оказывается очень хорошей для функций, на которых наложены не слишком сильные ограничения на гладкость.

Теорема. Если у функции $f(x)$ существует ограниченная на отрезке $[a,b]$ производная $f'(x)$, а последовательность интерполяционных сеток состоит из наборов чебышевских узлов, то последовательность соответствующих интерполяционных полиномов равномерно сходится к $f(x)$ на отрезке $[a,b]$.

2.3. Ошибка интерполяции, чебышевские узлы

При исследовании ошибки интерполяции важно ограничится некоторым классом функций, поскольку значения произвольной функции могут сколь угодно сильно отличаться от значений ее интерполянта в точках, лежащих между узлами интерполяции. Мы будем предполагать, что на отрезке интерполирования $[a,b]$ интерполируемая функция $f(x)$ имеет непрерывные производные до n-го порядка, а ее n-ая производная дифференцируема на отрезке $[a,b]$. Так же важно предположить, что все вычисления производятся без ошибок округления, поскольку, как было показано в предыдущем разделе Вычисление интерполяционного полинома в форме Лагранжа, ошибки округления сильно влияют на качество интерполяции при высоких степенях интерполяционных полиномов.

Погрешность интерполирования функции $f(x)$ интерполяционным полиномом $L_n(x)$ n-ой степени в точке x выражается следующим образом [3]

$$f(x) - L_n(x) = \frac{f^{(n+1)}(\xi)}{(n+1)!}\omega_{n+1}(x),$$

где ξ - некоторая точка из промежутка $[a,b]$ и $\omega_{n+1}(x)$ - полином степени $n+1$, определяемый по формуле

$$\omega_{n+1}(x) = (x-x_1)(x-x_2)\ldots(x-x_{n+1})$$

где $x_1,x_2,\ldots,x_{n+1}$ - узлы интерполяционного полинома $L_n(x)$.

Значение ω вообще говоря неизвестно, однако, если известно максимальное значение $\left|f^{(n+1)}(x)\right|$ на отрезке $[a,b]$:

$$M_{n+1} = \sup_{x\in[a,b]}\left|f^{(n+1)}(x)\right|$$

(или его оценка сверху), то можно пользоваться следующей оценкой сверху для ошибки интерполяции в точке x:

$$\left|f(x)-L_n(x)\right| \le \frac{M_{n+1}}{(n+1)!}\left|\omega_{n+1}(x)\right|.$$

Продемонстрируем использование этой оценки для проверки ошибки интерполяции на примере интерполяции функции $f(x) = \sin x$ на отрезке $[-\pi,\pi]$ интерполяционным полиномом 4-ой степени с равномерно отстоящими пятью узлами интерполяции. Для вычисления интерполяционного полинома $L_4(x)$ воспользуемся функцией lagrange, текст которой приведен в разделе Вычисление интерполяционного полинома в форме Лагранжа. Сначала зададим узлы интерполяции и вычислим в них значения функции $\sin x$:

```
>> x=linspace(-pi,pi,5)
x =
  -3.1416   -1.5708         0    1.5708    3.1416
>> y=sin(x)
y =
  -0.0000   -1.0000         0    1.0000    0.0000
```

Теперь найдем значение yy интерполяционного полинома $L_4(x)$ и ошибку интерполяции err в точке $\pi/6$:

```
>> xx=pi/6
xx =
```

```
   0.5236
>> yy=lagrange(x,y,xx)
yy =
   0.4321
>> err=abs(sin(xx)-yy)
err =
   0.0679
```

Вычислим правую часть оценки est с учетом того, что производные функции sin$x$ не превосходят единицу, т.е. заменим M_{n+1} на 1:

```
>> est=abs(prod(x-xx))/factorial(5)
est =
   0.0918
```

Видим, что est (оценка) больше, чем err (ошибка интерполяции) при $x = \frac{\pi}{6}$.

Мы проделали проверку оценки ошибки в точке, а следующие команды приводят к построению графиков ошибки интерполяции функции $f(x)$ = sin$x$ и ее оценки сверху на отрезке $[-\pi,\pi]$ интерполяционным полиномом 4-ой степени с равномерно отстоящими пятью узлами интерполяции.

Программа № 2.6 в MATLAB

Интерполирование классические функции с помощью интерполяционным полиномом в форме Лагранжа (*интерполяционным полиномом 4-ой степени с равномерно отстоящими пятью узлами интерполяции*)

```
% задание узлов интерполяции
x=linspace(-pi,pi,5);
% вычисление в них значений функции
y=sin(x);
% задание промежуточных точек
xx=linspace(-pi,pi,300);
% вычисление в них ошибки интерполяции
```

```
yy=lagrange(x,y,xx);
err=abs(yy-sin(xx));
% вычисление оценки ошибки в промежуточных точках
est=zeros(size(xx));
for i=1:length(xx)
   est(i)=abs(prod(x-xx(i)))/factorial(5);
end
% построение графиков ошибки и ее оценки сверху
figure('Color','w')
plot(xx,err,'g',xx,est,'r')
legend('error','estimate')
```

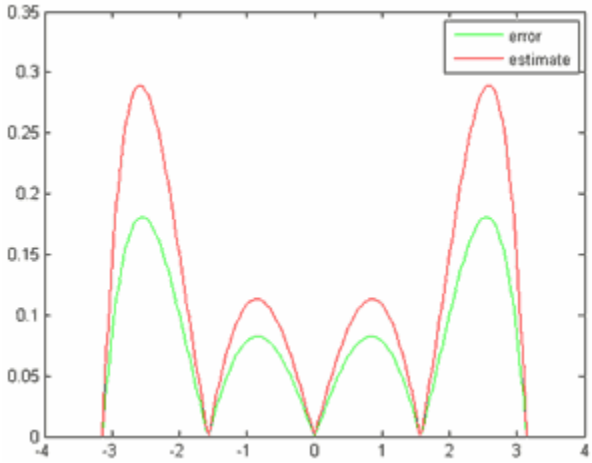

Рис.2.6. График ошибки интерполяции и ее оценки сверху

То, что в узлах интерполяции ошибка интерполяции и ее оценка сверху обращаются в ноль, очевидно, т.к. значения интерполянта в узлах совпадают со значениями интерполируемой функции и правая часть оценки

$$|f(x)-L_n(x)|\le\frac{M_{n+1}}{(n+1)!}|\omega_{n+1}(x)|$$

обращается в ноль, т.к.

$$\omega_n(x) = (x-x_1)(x-x_2)\dots(x-x_{n+1}) \equiv 0$$

в узлах интерполяции $x_1,x_2,\dots,x_{n+1}$.

Если нас интересует максимальное уклонение интерполяционного полинома от интерполируемой функции, то в последней оценке можно взять максимум по x:

$$\max_{x\in[a,b]}\left|f(x)-L_n(x)\right|\le\frac{M_{n+1}}{(n+1)!}\max_{x\in[a,b]}\left|\omega_{n+1}(x)\right|.$$

Из этой оценки видно, что мы можем управлять максимальной ошибкой за счет подходящего выбора узлов интерполяции. Цель выбора узлов интерполяции $\{x_1,x_2,\ldots,x_{n+1}\}$ - сделать максимальное значение модуля полинома

$$\omega_{n+1}(x) = (x-x_1)(x-x_2)\ldots(x-x_{n+1}) \equiv 0$$

на отрезке $[a,b]$ как можно меньше. Разумеется, эта задача имеет смысл только если мы можем вычислять интерполируемую функцию $f(x)$ в любых точках на отрезке $[a,b]$. Если функция задана таблично, то такой возможности нет.

Ограничимся сначала случаем $a = -1,\ b = 1$. Известно, что если узлы интерполяции $\{x_1,x_2,\ldots,x_{n+1}\}$ являются корнями полинома Чебышева степени $n + 1$, то величина

$$\max_{x\in[-1,1]}\left|\omega_{n+1}(x)\right|$$

принимает наименьшее возможное значение по сравнению с любым другим выбором набора узлов интерполяции.

Полиномы Чебышева, предложенные и исследованные П.Л. Чебышевым в середине 19-го века, определяются следующим образом:

$$T_k(x) = \cos(k \arccos x),\ |x|\le 1.$$

Очевидно, что для $k = 1$ функция $T_1(x)$ действительно является полиномом первой степени, поскольку

$$T_1(x) = \cos(\arccos x) = x$$

В случае $k = 2$ тоже очевидно, что $T_2(x)$ есть полином второй степени, если воспользоваться известным тригонометрическим тождеством

$$\cos 2\theta = 2\cos^2\theta - 1,$$

положив $\theta = \arccos x$. Тогда получаем

$$T_2(x) = 2x^2 - 1,$$

Тригонометрическое тождество

$$\cos(k + 1)\theta = 2\cos\theta\cos k\theta - \cos(k - 1)$$

позволяет легко установить, что для полиномов Чебышева справедливо следующее рекуррентное соотношение

$$T_{k+1}(x) = 2xT_k(x) - T_{k-1}(x)$$

При помощи этого рекуррентного соотношения можно последовательно получить формулы для полиномов Чебышева любой степени. Из этого рекуррентного соотношения видно, что коэффициент при старшей степени x^k равен 2^{k-1}.

В MATLAB полиномы Чебышева можно вывести при помощи Symbolic Math Toolbox, обратившись к функции Maple, которая называется ChebyshevT и функции simplify (для вызова функций Maple из MATLAB предназначена функция maple):

```
>> p2=maple('simplify(ChebyshevT(2,x),"ChebyshevT")')
p2 =
-1+2*x^2

>> p3=maple('simplify(ChebyshevT(3,x),"ChebyshevT")')
p3 =
4*x^3-3*x

>> p4=maple('simplify(ChebyshevT(4,x),"ChebyshevT")')
p4 =
1+8*x^4-8*x^2

>> p5=maple('simplify(ChebyshevT(5,x),"ChebyshevT")')
p5 =
```

16*x^5-20*x^3+5*x

Построим графики полиномов Чебышева для нескольких различных значений, для чего выполним следующие команды:

Программа № 2.7 в MATLAB

Интерполирование классические функции с помощью интерполяционным полиномом Чебышева любой степени

```
% создание графического окна с осями и координатной сеткой
figure('Color','w')
axes
grid on
hold on
% запись в строковую переменную выражения для полинома Чебышева 2-ой степени
p2=maple('simplify(ChebyshevT(2,x),"ChebyshevT")');
% создание inline-функции
t2=inline(p2);
% построение ее графика на отрезке [-1,1]
fplot(t2,[-1,1],'r')
% аналогично для полиномов 4-ой, 7-ой и 10-степеней
p4=maple('simplify(ChebyshevT(4,x),"ChebyshevT")');
t4=inline(p4);
fplot(t4,[-1,1],'g')
p7=maple('simplify(ChebyshevT(7,x),"ChebyshevT")');
t7=inline(p7);
fplot(t7,[-1,1],'b')
p10=maple('simplify(ChebyshevT(10,x),"ChebyshevT")');
```

```
t10=inline(p10);
fplot(t10,[-1,1],'m')
% вывод легенды
legend('{\itk}=2','{\itk}=4','{\itk}=7','{\itk}=10')
```

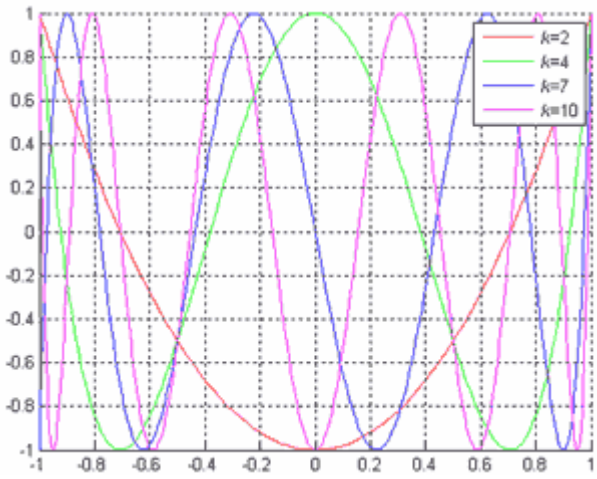

Рис.2.7. Графики полиномов Чебышева

Корни полинома Чебышева легко найти, решив уравнение

$$T_k(x) = \cos(k \ \arccos \ x) = 0,$$

из которого видно, что полином $T_k(x)$ имеет ровно k различных корней, расположенных на отрезке [-1,1]:

$$x_m = \cos\frac{2m+1}{2k}\pi, \quad m = 0,1,\ldots,k-1,$$

которые и следует выбирать в качестве узлов интерполирования. Корни полиномов Чебышева расположены симметрично относительно нуля на отрезке [-1,1] и неравномерно - чем ближе к краям отрезка, тем корни расположены плотнее.

Максимальное значение модуля полинома Чебышева $|T_k(x)|$ равно 1 и достигается в точках $\cos\frac{m}{k}\pi$.

Если мы положим

$$\omega_k(x) = \frac{1}{2^{k-1}}T_k(x)$$

для того, чтобы коэффициент при старшей степени полинома $\omega_k(x)$ был равен 1 (по определению полинома $\omega_k(x)$), то получим, что

$$\max_{x\in[-1,1]}|\omega_k(x)| = \frac{1}{2^{k-1}}.$$

Несложно показать (ссылка в списке литературы), что для любого полинома $p_k(x)$ степени k с коэффициентом при старшей производной равным единице будет верно следующее неравенство

$$\max_{x\in[-1,1]}|p_k(x)| \geq \frac{1}{2^{k-1}},$$

т.е. полиномы Чебышева являются полиномами наименее уклоняющимися от нуля.

Таким образом, выбор в качестве узлов интерполирования корней полинома Чебышева является наилучшим в смысле минимизации правой части оценки

$$\max_{x\in[-1,1]}|f(x)-L_n(x)| \leq \frac{M_{n+1}}{(n+1)!}\max_{x\in[-1,1]}|\omega_{n+1}(x)|,$$

которая теперь приобретает вид:

$$\max_{x\in[-1,1]}|f(x)-L_n(x)| \leq \frac{M_{n+1}}{2^n(n+1)!},$$

где

$$M_{n+1} = \sup_{x\in[-1,1]}\left|f^{(n+1)}(x)\right|.$$

Продемонстрируем преимущество выбора корней полиномов Чебышева в качестве узлов интерполяции. Для примера будем интерполировать функцию $f(x) = \sin x$ на отрезке [-1,1] полиномами различных степеней для равноотстоящих и чебышевских узлов и сравнивать получающуюся ошибку интерполяции, строя графики ошибки $L_n(x)$ - $sin(x)$. Для этого можно воспользоваться последовательностью команд, приведенной ниже, в которой следует изменять значение k (т.е. число узлов интерполяции).

Программа № 2.8 в MATLAB

Построение графиков ошибки интерполяции для равноотстоящих и чебышевских узлов для нескольких чисел узлов интерполяции

```
% задание числа узлов интерполяции
k=7;
% генерация равномерно отстоящих узлов на отрезке [-1,1]
x=linspace(-1,1,k);
% вычисление в них значений функции синус
y=sin(x);
% генерация 500 равномерно отстоящих точек на отрезке [-1,1]
xx=linspace(-1,1,500);
% вычисление в них значений интерполяционного полинома
yy=lagrange(x,y,xx);
% нахождение ошибки интерполяции в этих точках
err=yy-sin(xx);

% вычисление корней полинома Чебышева в качестве узлов на отрезке [-1,1]
m=0:k-1;
xcheb=cos((2*m+1)/k*0.5*pi);
% вычисление в них значений функции синус
ycheb=sin(xcheb);
% вычисление значений интерполяционного полинома в 500 равномерно отстоящих
% точек на отрезке [-1,1]
yy=lagrange(xcheb,ycheb,xx);
% нахождение ошибки интерполяции в этих точках
errcheb=yy-sin(xx);
% построение графиков ошибок для равномерных и чебышевских узлов
figure('Color','w')
plot(xx,err,xx,errcheb)
% вывод легенды и заголовка
```

```
legend('equidistance','chebyshev')
title('Interpolation error')
```

Ниже приведены графики ошибки интерполяции для равноотстоящих и чебышевских узлов для нескольких чисел узлов интерполяции (рис.2.8). На них отчетливо прослеживается типичная ситуация, когда при интерполяции с равноотстоящими узлами ошибка интерполяции больше на краях промежутка интерполяции, а при интерполяции с чебышевскими узлами ошибка более равномерно распределена на интервале интерполирования.

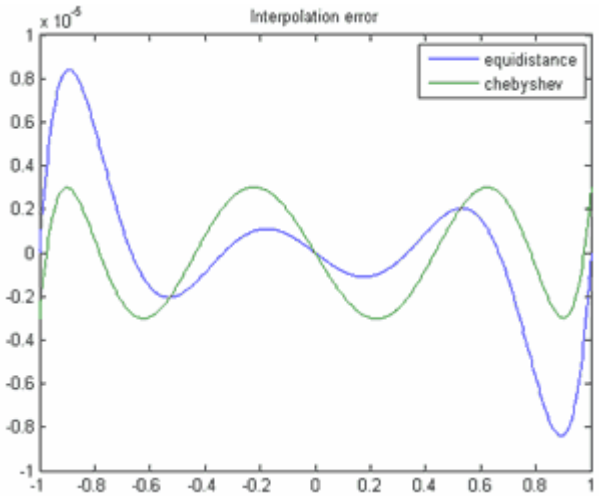

Рис.2.8,*а*. Ошибки интерполяции для 7 узлов

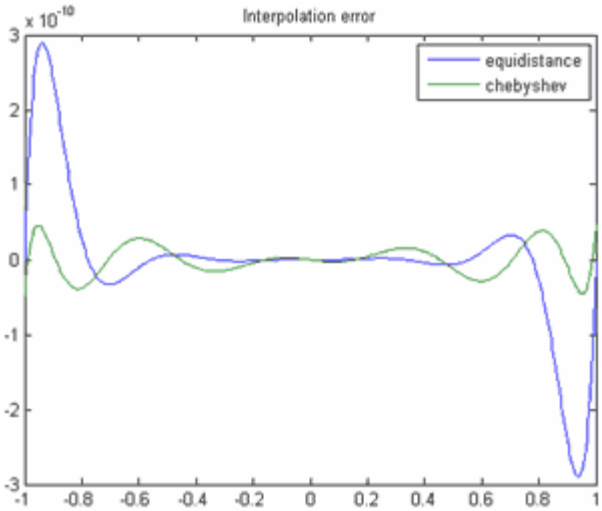

Рис.2.8,*б*. Ошибки интерполяции для 10 узлов

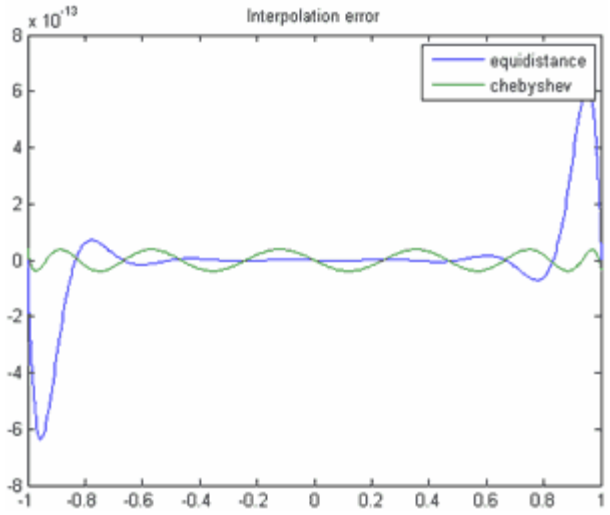

Рис.2.8,*в*. Ошибки интерполяции для 13 узлов

Мы рассмотрели выбор чебышевских узлов, когда отрезок, на котором производится интерполирование функции, есть [-1,1]. Переход к общему случаю интерполирования функции на произвольном отрезке [*a,b*] не представляет никаких трудностей. Достаточно сделать линейную замену переменных, переводящую отрезок [-1,1] в отрезок [*a,b*]. Такая замена выражается следующей простой формулой:

$$x = \frac{(b-a)\xi + a + b}{2}, \quad \xi \in [-1,1], \quad x \in [a,b].$$

Тогда для произвольного отрезка [*a,b*] чебышевскими узлами интерполяции будут узлы, вычисляемые по формуле:

$$x_m = \frac{(b-a)\cos\dfrac{2m+1}{2k}\pi + a + b}{2},$$

где $m = 0,1,...,k-1$. Здесь k - число узлов интерполяции, т.е. $k = n + 1$, где n - степень интерполяционного полинома $L_n(x)$.

Оценка ошибки интерполяции для отрезка [*a,b*] приобретает вид:

$$|f(x) - L_n(x)| \le \frac{M_{n+1}}{2^{2n+1}(n+1)!}(b-a)^{n+1},$$

где

$$M_{n+1} = \sup_{x \in [a,b]} \left| f^{(n+1)}(x) \right|.$$

Приведем пример интерполирования с чебышевскими и равноотстоящими узлами функции

$$f(x) = \sin^2 x + \sin x^2$$

на отрезке [-2,3]. Будем выводить график ошибки интерполирования для различного числа равноотстоящих и чебышевских узлов, для чего в приведенной ниже программе следует менять значение переменной k.

Программа № 2.9 в MATLAB

Построение графиков ошибки интерполирования для различного числа равноотстоящих и чебышевских узлов для нескольких чисел узлов интерполяции

```
% задание границ отрезка интерполирования
a=-2;
b=3;
% задание inline-функции
fun=inline('sin(x).^2+sin(x.^2)');
% задание числа узлов интерполяции
k=7;
% генерация равномерно отстоящих узлов на отрезке [a,b]
x=linspace(a,b,k);
% вычисление в них значений функции sin(x)^2+sin(x^2)
y=fun(x);
% генерация 500 равномерно отстоящих точек на отрезке [a,b]
xx=linspace(a,b,500);
% вычисление в них значений интерполяционного полинома
yy=lagrange(x,y,xx);
% нахождение ошибки интерполяции в этих точках
```

```
err=yy-fun(xx);

% вычисление чебышевских узлов на отрезке [a,b]
m=0:k-1;
xcheb=0.5*((b-a)*cos((2*m+1)/k*0.5*pi)+a+b);
% вычисление в них значений функции sin(x)^2+sin(x^2)
ycheb=fun(xcheb);
% вычисление значений интерполяционного полинома в 500 равномерно отстоящих
% точек на отрезке [a,b]
yy=lagrange(xcheb,ycheb,xx);
% нахождение ошибки интерполяции в этих точках
errcheb=yy-fun(xx);
% построение графиков ошибок для равномерных и чебышевских узлов
figure('Color','w')
plot(xx,err,xx,errcheb)
% вывод легенды и заголовка
legend('equidistance','chebyshev')
title('Interpolation error')
```

Ниже приведены графики ошибки интерполяции для равноотстоящих и чебышевских узлов для нескольких чисел узлов интерполяции (рис.2.9). Ситуация аналогична случаю отрезка, рассмотренному выше. При интерполяции с равноотстоящими узлами ошибка интерполяции больше на краях промежутка интерполяции, а при интерполяции с чебышевскими узлами ошибка более равномерно распределена на интервале интерполирования и меньше по значению, чем при интерполяции с равноотстоящими узлами.

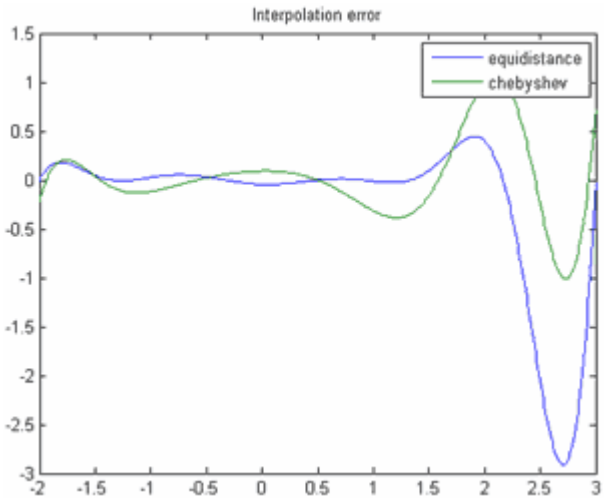

Рис.2.9,*а*. Ошибки интерполяции для 7 узлов

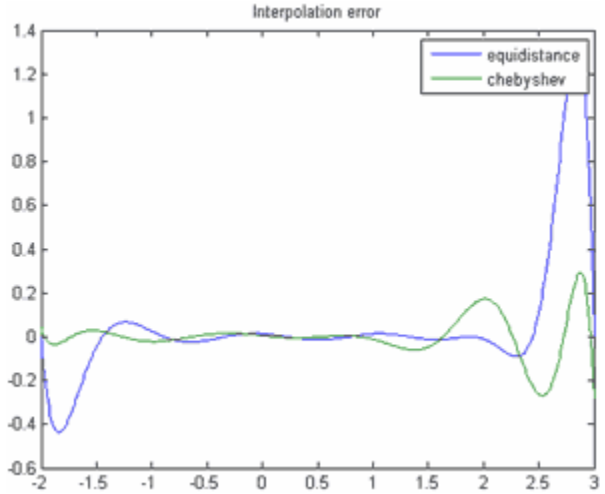

Рис.2.9,*б*. Ошибки интерполяции для 10 узлов

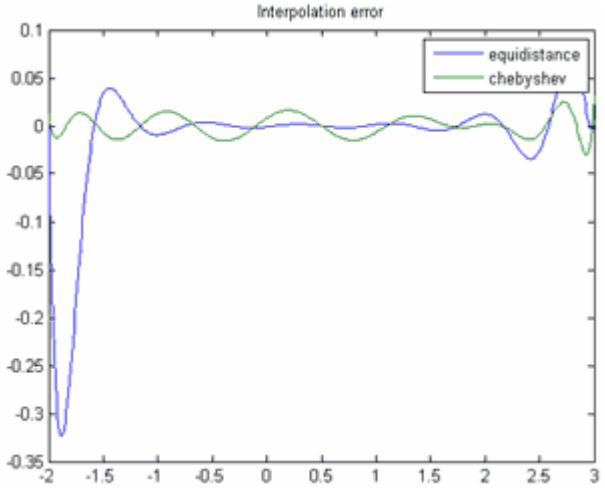

Рис.2.9,*в*. Ошибки интерполяции для 13 узлов

2.4. Сходимость интерполяционного полинома

Важно определить, что мы понимаем под сходимостью интерполяционного полинома $L_n(x)$ к интерполируемой функции $f(x)$.

Говорят, что интерполяционный полином $L_n(x)$ сходится к $f(x)$ в точке x, если

$$L_n(x) \xrightarrow[n\to\infty]{} f(x)$$

и интерполяционный полином $L_n(x)$ равномерно сходится к $f(x)$, если

$$\max_{x\in[a,b]} |L_n(x) - f(x)| \xrightarrow[n\to\infty]{} 0 .$$

Отметим, что из поточечной сходимости не следует равномерная и, если нет поточечной сходимости, то нет и равномерной.

Как гласит теорема Вейерштрасса, всякая непрерывная функция на отрезке может быть равномерно приближена полиномом с любой точностью, однако из этой теоремы не следует существование именно интерполяционного полинома, который приближал бы с любой точностью заданную непрерывную функцию.

Мы начнем изучение сходимости интерполяционного полинома с классического примера Рунге (1901г.). Рунге исследовал интерполяцию полиномами функции

$$r(x) = \frac{1}{1 + 25x^2}$$

с равноотстоящими узлами на отрезке [-1,1] и доказал, что вблизи границ этого отрезка при $x>0.72$ погрешность интерполяции $f(x)$-$L_n(x)$ неограниченно увеличивается с увеличением степени интерполяционного полинома n.

Проинтерполируем функцию Рунге $r(x)$ на отрезке [-1,1], увеличивая число равноотстоящих узлов, и отобразим графически функцию $r(x)$ и интерполяционные полиномы на одних осях, а погрешности интерполирования на других осях в одном графическом окне.

Программа № 2.10 в MATLAB

Построение графиков интерполирования функции Рунге r(x) и погрешности интерполирования на отрезке [-1,1], увеличивая число равноотстоящих узлов

```
% задание функции Рунге
r=inline('1./(1+25*x.^2)');
% создание графического окна и двух пар осей
figure('Color','w')
subplot(2,1,1)
fplot(r,[-1,1],'r')
hold on
subplot(2,1,2)
hold on
% генерация 500 равномерно отстоящих точек на отрезке [-1,1]
xx=linspace(-1,1,500);
        % задание 6 равноотстоящих узлов интерполирования
k=6;
x=linspace(-1,1,k);
% вычисление в них значения функции Рунге
y=r(x);
% вычисление значений интерполяционного полинома в точках xx
% и построение его графика
yy=lagrange(x,y,xx);
subplot(2,1,1)
plot(xx,yy,'b')
% нахождение ошибки интерполяции в этих точках
% и построение ее графика
err=yy-r(xx);
subplot(2,1,2)
```

```
plot(xx,err,'b')
```

% задание 10 равноотстоящих узлов интерполирования

```
k=10;
x=linspace(-1,1,k);
% вычисление в них значения функции Рунге
y=r(x);
% вычисление значений интерполяционного полинома в точках xx
% и построение его графика
yy=lagrange(x,y,xx);
subplot(2,1,1)
plot(xx,yy,'g')
% нахождение ошибки интерполяции в этих точках
% и построение ее графика
err=yy-r(xx);
subplot(2,1,2)
plot(xx,err,'g')
```

% задание 12 равноотстоящих узлов интерполирования

```
k=12;
x=linspace(-1,1,k);
% вычисление в них значения функции Рунге
y=r(x);
% вычисление значений интерполяционного полинома в точках xx
% и построение его графика
yy=lagrange(x,y,xx);
subplot(2,1,1)
plot(xx,yy,'k')
% нахождение ошибки интерполяции в этих точках
% и построение ее графика
err=yy-r(xx);
```

```
subplot(2,1,2)
plot(xx,err,'k')
% вывод легенд на оси
subplot(2,1,1)
legend('{\itr}({\itx})', '{\itL}_6({\itx})', '{\itL}_{10}({\itx})',
'{\itL}_{12}({\itx})',-1)
subplot(2,1,2)
legend('{\itL}_6({\itx})-{\itr}({\itx})', '{\itL}_{10}({\itx})-{\itr}({\itx})',...
   '{\itL}_{12}({\itx})-{\itr}({\itx})',-1)
```

В результате получаем, что погрешность интерполирования вблизи границ промежутка при $|x|>0.72$ растет с ростом степени интерполяционного полинома (рис.2.10).

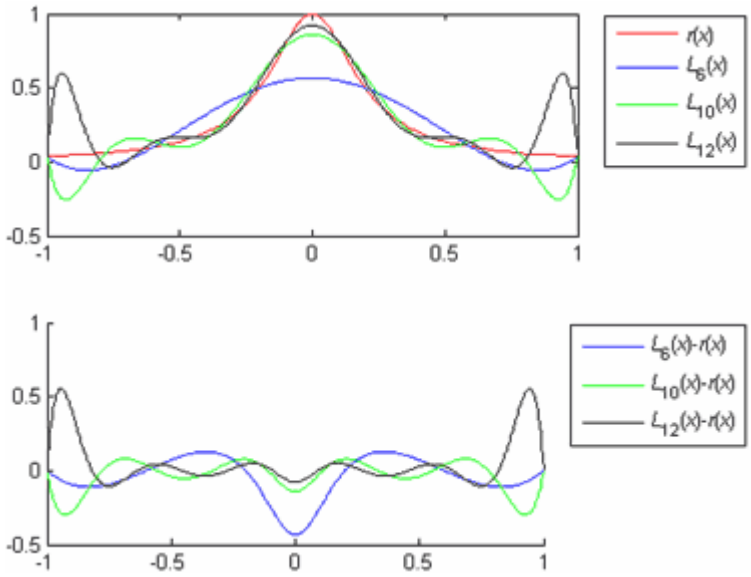

Рис.2.10. Функция Рунге и интерполяционные полиномы (сверху), ошибка интерполяции (снизу).

Однако, если в качестве узлов интерполирования выбрать чебышевские узлы, то мы увидим, что интерполяционный полином сходится к функции Рунге (для больших степеней интерполяционных полиномов ошибки округления не дадут проследить сходимость интерполяционного полинома к функции Рунге).

ЗАКЛЮЧЕНИЕ

- демонстрированы возможностей MATLAB для изучения вопросов, возникающих при интерполяции функций, в основном при помощи интерполяционных полиномов;
- приведены необходимые сведения об интерполяции функций;
- написаны небольшие программы на языке пакета MATLAB для изучения проблемы, возникающие при интерполяции функций;
- простота языка пакета MATLAB в сочетании с широким набором его функций, в том числе и графических, позволяет вместо написания собственных программ интерполяции и визуализации результатов сосредоточиться на исследовании большого числа примеров;
- число узлов интерполяционного полинома всегда должно быть на единицу больше его степени;
- какие бы подходы для построения интерполяционного полинома не применялись, они всегда должны привести к одинаковому результату;
- график интерполяционного полинома пятой степени практически совпадает с графиком исходного полинома, однако ошибка вовсе не равна нулю (график распределения ошибки имеет характерный вид - в середине интервала ошибка меньше, чем по краям);
- сначала ошибка интерполирования уменьшается и достигает своего минимума (когда степень интерполяционного полинома равна 5), но она не равна в точности нулю из-за ошибок округления при вычислении интерполяционного полинома. Далее, при повышении степени интерполяционного полинома, ошибка начинает быстро возрастать, что свидетельствует о практической неприменимости интерполяционных полиномов высоких степеней;
- при интерполяции с равноотстоящими узлами ошибка интерполяции больше на краях промежутка интерполяции, а при интерполяции с чебышевскими узлами ошибка более равномерно распределена на интер-

вале интерполирования и меньше по значению, чем при интерполяции с равноотстоящими узлами;

- если интерполируем функцию Рунге, то погрешность интерполирования вблизи границ промежутка [-1;1] при $|x|>0.72$ растет с ростом степени интерполяционного полинома;
- если требуется найти значения интерполяционного полинома в достаточно большом числе точек, то нужно использовать более эффективный способ;
- результаты работы может быть использовано при проведении лабораторных и самостоятельных работ по численным методам для студентов вузов и втузов.

СПИСОК ИСПОЛЬЗОВАННОЙ ЛИТЕРАТУРЫ

1. Иванов В.В. Методы вычислений на ЭВМ. Справочное пособие. – Киев: Изд-во "Наукова думка", 1986. – 340 с.
2. Де Бор К. Практическое руководство по сплайнам. – Москва: Изд-во "Радио и связь", 1985. – 286 с.
3. Форсайт Дж., Мальком М., Моулер К. Машинные методы математических вычислений. – Москва: Изд-во "Мир", 1980. – 348 с.
4. Алексеев Е.Р., Чеснокова О.В. Решение задач вычислительной математики в пакетах Mathcad, Mathlab, Maple (Самоучитель). – М.: НТ Пресс, 2006. – 496 с.
5. Бахвалов Н. С., Жидков Н. П., Кобельков Г. М. Численные методы. – М.: Изд-во Бином. Лаборатория знаний, 2011. – 640 с.
6. Вержбицкий В. М. Основы численных методов. – М.: Высшая школа, 2009. – 848 с.
7. Калиткин Н.Н., Альшина Е.А. Численные методы: в 2 кн. Кн. 1. Численные анализ. - М. : Издательский центр «Академия», 2013. - 304 с.
8. Кетков Ю.Л., Кетков А.Ю., Шульц М.М. MATLAB 7: программирование, численные методы. – СПб.: БХВ-Петербург, 2005. – 752 с.
9. Кирянов Д.В. Mathcad 13. – СПб.: БХВ-Петербург, 2006. – 608 с.
10. Макаров Е. Инженерные расчеты в Mathcad 15: Учебный курс. – СПб.: Питер, 2011. – 400 с.
11. Марчук Г.И. Методы вычислительной математики. – М.: Изд-во Лань, 2010. – 608 с.
12. Мэтьюз Джон Г., Финк Куртис Д. Численные методы. Использование Matlab. 3-издание: Пер. с англ. – М.: Изд-во дом «Вильямс», 2001. – 720 с.

13. Половко А.М., Бутусов П.Н. MATLAB для студента. – СПб.: БХВ-Петербург, 2005. – 320 с.
14. Самарский А.А. Введение в численные методы. – М.: Изд-во Лань, 2009. - 288 с.
15. http://www.edu.uz – образовательный сайт.
16. http://www.edu.ru – образовательный сайт.
17. http://www.intuit.ru – дистанционный образовательный сайт.
18. http://www.eqworld.ru – электронный вариант литературы.
19. http://www.twirpx.com электронный вариант литературы.
20. http://www.ziyonet.uz - электронный вариант литературы.