Федеральное агентство железнодорожного транспорта

Омский государственный университет путей сообщения

Кафедра «Автоматика и системы управления»

Допущен к защите
«____» _____ 2022 г.

_____ _____
Подпись Расшифровка подписи

ПРОГРАММИРОВАНИЕ ЧИСЛЕННЫХ МЕТОДОВ
Пояснительная записка к курсовой работе по дисциплине
«Теоретические основы аппаратно-программных средств»
ИНМВ.402700.000 ПЗ

Оценка
«____» _____ 2022 г.

_____ _____
Подпись Расшифровка подписи

Студент гр. 21 м

_____ К.Н. Юрукина
«__»_____ 2022 г.

Доцент кафедры АиСУ
_____ А.С. Окишев
«__»_____ 2022 г.

Омск 2022

Реферат

УДК 519.6

Пояснительная записка содержит 36 страниц, 13 рисунков, 6 таблиц, 4 источника.

Численный метод, нелинейное уравнение, корень, итерация, сходимость, аппроксимация, интерполяция, задача Коши, обыкновенное дифференциальное уравнение.

Объектом исследования являются приближенные численные методы решения некоторых математических и инженерных задач, а также программное обеспечение, реализующее эти методы.

Цель работы – ознакомиться с численными методами решения нелинейных и дифференциальных уравнений и интерполяции функций, решить предложенные типовые задачи с помощью предоставленного преподавателем программного обеспечения, сформулировать выводы по полученным решениям, отметить достоинства и недостатки методов, сравнить удобство использования и эффективность работы каждой программы.

Пояснительная записка к курсовой работе оформлена в текстовом редакторе Microsoft Office 2016. Графики нелинейных функций построены с помощью программы Matlab R2022a. При решении обыкновенных дифференциальных уравнений использовалась среда математического моделирования Matlab R2022a.

Содержание

Введение 4
1 Решение нелинейных уравнений 5
1.1 Метод простых итераций 6
1.2 Метод Ньютона 8
1.3 Решение нелинейного уравнения методом простых итераций 10
1.4 Решение нелинейного уравнения методом Ньютона 12
1.5 Вывод 13
2 Интерполяция функций 15
2.1 Локальная и глобальная интерполяция 16
2.2 Кусочно-линейная интерполяция 16
2.3 Кусочно-квадратичная интерполяция 17
2.4 Кубические сплайны 17
2.5 Многочлен Лагранжа 19
2.6 Кусочно-квадратичная интерполяция функции по таблице значений 20
2.7 Интерполяция функции кубическим сплайном по таблице значений 24
2.8 Интерполяция функции многочленом Лагранжа по таблице значений 26
2.9 Вывод 28
3 Решение обыкновенных дифференциальных уравнений 29
3.1 Метод Эйлера 29
3.2 Метод Эйлера-Коши 31
3.3 Решение ОДУ методом Эйлера 31
3.4 Решение ОДУ методом Эйлера-Коши 33
3.5 Вывод 34
Заключение 35
Библиографический список 36

Введение

В связи с развитием новой вычислительной техники инженерная практика наших дней все чаще и чаще встречается с математическими задачами, точное решение которых получить весьма сложно или невозможно. В этих случаях обычно прибегают к тем или иным приближенным вычислениям. Вот почему приближенные и численные методы математического анализа получили за последние годы широкое развитие и приобрели исключительно важное значение.

Новые вычислительные средства вызвали переоценку известных методов решения задач с точки зрения целесообразности их реализации на ЭВМ и стимулировали создание более эффективных, что привело к появлению новой дисциплины – вычислительной математики. Предметом изучения последней являются численные методы решения задач математического анализа: изучение алгоритмов и условий сходимости итерационных методов, определение границ применимости методов, исследования оценок погрешностей методов и вычислений. Главным разделом вычислительной математики является реализация численных методов на ЭВМ, то есть составление программы для требуемого алгоритма и решения с ее помощью конкретной задачи.

Любая прикладная задача формируется исходя из определенного физического смысла некоторого процесса (распределение тепла в стержне, описание траектории движения объектов). Прикладная математическая задача может быть сформулирована, например, из описания некоторой экономической модели (задача распределения ресурсов, задача планирования производства, транспортная задача перевозки грузов, оптимальных в заданном смысле). Следовательно, для постановки любой прикладной задачи нужна математическая модель. Поэтому, можно выделить следующие этапы решения задач на ЭВМ:

1) описание математической модели задачи на основе физической или экономической модели;

2) изучение методов решения поставленной математической модели задачи и создание новых методов;

3) выбор метода решения задачи исходя из заданной точности решения и особенностей задачи;

4) составление блок-схемы программы для решения задачи на ЭВМ;

5) отладка программы и оценка полученных результатов;

6) решение задачи на ЭВМ, построение графиков, получение оценки погрешностей, обоснование результатов.

В курсовой работе рассматриваются не прикладные, а типовые математические задачи, которые могут возникнуть при переходе от реальных систем к их математическим моделям, поэтому основное внимание уделяется последнему этапу.

1 Решение нелинейных уравнений

Нелинейными уравнениями называются уравнения вида

$$f(x)=0, \tag{1.1}$$

где $f(x)$ – нелинейная функция, которая может относиться к трем типам:

1) нелинейная алгебраическая функция вида

$$a_n x^n + a_{n-1} x^{n-1} + ... + a_1 x + a_0;$$

2) трансцендентные функции – тригонометрические, обратные тригонометрические, логарифмические, показательные и гиперболические функции;

3) различные комбинации этих функций, например, $f(x)=x^2+\sin x$.

Решением нелинейного уравнения (1.1) является такая точка x^*, которая при подстановке в уравнение (1.1) обращает его в тождество. На практике не всегда удается подобрать такое решение. В этом случае решение уравнения (1.1) находят с применением приближенных (численных) методов. Тогда решением будет являться такая точка x^*, при подстановке которой в уравнение (1.1) последнее будет выполняться с определенной степенью точности, т.е. $\left|f(x^*)\right| \le \varepsilon$, где ε – малая величина. Нахождение таких решений и составляет основу численных методов и вычислительной математики.

Решение нелинейных уравнений разделяется на два этапа: отделение корней уравнений и уточнение корней нелинейных уравнений.

На первом этапе необходимо исследовать уравнение и выяснить, имеются корни или нет. Если корни имеются, то узнать, сколько их, и затем определить интервалы, в каждом из которых находится единственный корень.

Первый способ отделения корней – графический. Исходя из уравнения (1.1), можно построить график функции $y=f(x)$. Тогда точка пересечения графика с осью абсцисс является приближенным значением корня. Если *f*(*x*) имеет сложный вид, то ее можно представить в виде разности двух функций $f(x)=f_1(x)-f_2(x)$. Так как $f(x)=0$, то выполняется равенство $f_1(x)=f_2(x)$. Если построить два графика $y_1=f_1(x)$, $y_2=f_2(x)$, то абсцисса точки их пересечения будет приближенным значением корня уравнения.

Второй способ отделения корней нелинейных уравнений – аналитический. Он основывается на следующих трех теоремах.

Теорема 1. Если функция $f(x)$ непрерывна на отрезке $[a;b]$ и меняет на концах отрезка знак (т.е. $f(a)\cdot f(b)<0$), то на $[a;b]$ содержится хотя бы один корень.

Теорема 2. Если функция $f(x)$ непрерывна на отрезке $[a;b]$, выполняется условие вида $f(a)\cdot f(b)<0$ и производная $f'(x)$ сохраняет знак на $[a;b]$, то на отрезке имеется единственный корень.

Теорема 3. Если функция $f(x)$ является многочленом *n*-ой степени и на концах отрезка $[a;b]$ меняет знак, то на $[a;b]$ имеется нечетное количество корней (если производная $f'(x)$ сохраняет знак на $[a;b]$, то корень единственный). Если на концах отрезка $[a;b]$ функция не меняет знак, то уравнение (1.1) либо не имеет корней на $[a;b]$, либо имеет четное количество корней.

При аналитическом методе исследований необходимо выявить интервалы монотонности функции $f(x)$. Для этого необходимо вычислить критические точки $(\xi_1, \xi_2, ..., \xi_n)$, в которых первая производная $f'(\xi_i)$ равна нулю или не существует. Тогда вся числовая ось разбивается на интервалы монотонности (ξ_i, ξ_{i+1}). На каждом из них определяется знак производной $f'(x_i)$, где $x_i \in (\xi_i, \xi_{i+1})$. Затем выделяются те интервалы монотонности, на которых функция $f(x)$ меняет знак.

На втором этапе на каждом из этих интервалов для поиска корня используются численные итерационные методы уточнения корней, например методы половинного деления, простых итераций или Ньютона.

1.1 Метод простых итераций

Пусть известно, что нелинейное уравнение (1.1) имеет на отрезке $[a,b]$ единственный вещественный корень $x^* \in [a,b]$. Требуется найти этот корень с заданной точностью. Применяя тождественные преобразования, приведем уравнение к виду

$$x = \varphi(x). \tag{1.2}$$

Выберем произвольно приближенное значение корня $x_0 \in [a,b]$ и вычислим $\varphi(x_0) = x_1$. Найденное значение x_1 подставим в правую часть соотношения (1.2) и вычислим $\varphi(x_1) = x_2$. Продолжая процесс вычислений дальше, получим числовую последовательность $x_0, x_1, x_2, ...$ Если существует предел этой последовательности, то он и является корнем уравнения (1.2). В самом деле, пусть $\lim\limits_{k\to\infty} x_k = x^*$. Тогда, переходя к пределу в равенстве (1.2)

$$\lim_{k\to\infty} x_{k+1} = \lim_{k\to\infty} \varphi(x_k)$$

и учитывая непрерывность функции $\varphi(x_k)$ на отрезке $[a,b]$, получим

$$\lim_{k\to\infty} x_{k+1} = \varphi\left(\lim_{k\to\infty} x_k\right) \text{ или } x^* = \varphi(x^*).$$

Корень можно вычислить с заданной точностью по итерационной формуле

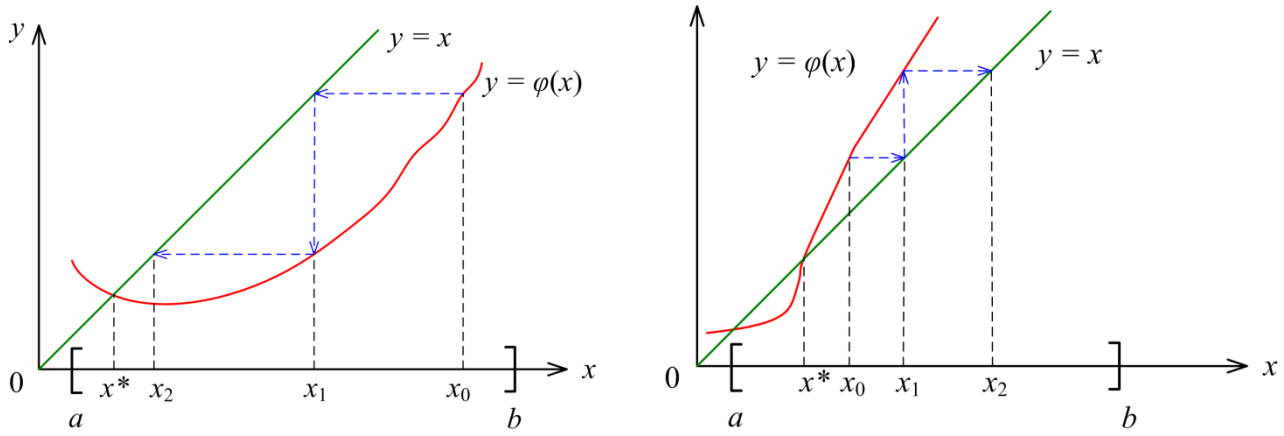
$$x_{k+1} = \varphi(x_k), \quad k = 0, 1, 2, ... \tag{1.3}$$

Достаточное условие, при котором итерационный процесс сходится, определяет следующая теорема: пусть функция $\varphi(x)$ определена и дифференцируема на отрезке $[a,b]$, причем все ее значения $\varphi(x)\in[a,b]$ и выполняется условие

$$|\varphi'(x)|\le q<1, \quad \forall x\in[a,b]. \tag{1.4}$$

тогда процесс итераций (1.3) сходится независимо от начального значения $x_0\in[a,b]$ и предельное значение $x^*=\lim\limits_{k\to\infty}x_k$ является единственным корнем уравнения (1.2) на $[a,b]$.

Геометрическая интерпретация метода простых итераций заключается в следующем: если построить два графика функций $y=x$ и $y=\varphi(x)$, то абсцисса точки их пересечения будет корнем x^*. Построим итерационный процесс. Зададим $x_0\in[a,b]$. Вычислим $x_1=\varphi(x_0)$ – первое приближение и $x_2=\varphi(x_1)$ – второе приближение. В первом случае (рисунок 1.1, а) процесс сходящийся ($|\varphi'(x)|<1$), во втором (рисунок 1.1, б) – расходящийся ($|\varphi'(x)|>1$).



а б

Рисунок 1.1 – Сходящийся (а) и расходящийся (б) итерационные процессы

Часто, если итерационный процесс расходится из-за невыполнения условия (1.4), нелинейное уравнение можно привести к виду, допускающему сходящиеся итерации.

Выполнения условия сходимости можно добиться путем перехода от исходного уравнения $f(x)=0$ к эквивалентному виду $x=\varphi(x)$ следующим образом: сначала умножить обе части уравнения (1.1) на $c=\text{const}\neq 0$, а затем прибавить к обеим частям x, тогда $x+cf(x)=x$. Обозначив $\varphi(x)=x+cf(x)$, получим уравнение (1.2). Константа c выбирается так, чтобы выполнялось достаточное условие сходимости итерационного процесса (1.4), т.е.

$$|\varphi'(x)|=|1+cf'(x)|<1, \quad \forall x\in[a,b]. \tag{1.5}$$

Условие (1.5) равносильно двойному неравенству

$$-1<1+cf'(x)<1,$$

поэтому константа выбирается из соотношений:

$$\begin{cases} \dfrac{-2}{f'(x)} < c < 0, \quad f'(x) > 0, \forall x \in [a,b]; \\ 0 < c < \dfrac{-2}{f'(x)}, \quad f'(x) < 0, \forall x \in [a,b]. \end{cases}$$

Метод простых итераций и почти все другие итерационные методы имеют два достоинства:

1) являются универсальными и самоисправляющимися, то есть любая неточность на каком-либо шаге итераций отражается не на конечном результате, а лишь на количестве итераций. Подобные ошибки устойчивы даже по отношению к грубым ошибкам (сбоям ЭВМ), если только ошибка не выбрасывает очередное приближение за пределы области сходимости;

2) позволяют достигнуть любой заданной точности при любом начальном приближении $x_0 \in [a,b]$.

Недостатки методов:

– трудность приведения уравнения (1.1) к виду (1.2).

– если начальное приближение x_0 находится далеко от корня, то число итераций при этом увеличивается, а объем вычислений возрастает.

Процесс итераций заканчивается при выполнении двух критериев:

1) Когда два последних приближения отличается между собой по модулю на заданную величину ε:

$$|x_{k+1} - x_k| \le \varepsilon. \tag{1.6}$$

Одного критерия (1.6) недостаточно, так как в случае крутизны графика, данное условие будет выполнено, но x_{k+1} может находиться далеко от корня;

2) Когда последнее вычисленное приближение к корню удовлетворяет уравнению с заданной точностью:

$$|f(x_{k+1})| \le \delta. \tag{1.7}$$

Отдельно критерия (1.7) бывает недостаточно, так как при пологой функции $f(x)$ условие может быть выполнено, но x_{k+1} может быть далеко от корня.

1.2 Метод Ньютона

Пусть уравнение (1.1) имеет на интервале $[a,b]$ единственный корень, причем существует непрерывная на $[a,b]$ производная $f'(x)$. Метод Ньютона служит для уточнения корней нелинейных уравнений в заданном интервале. Его можно рассматривать как частный случай метода простых итераций, если

$$\varphi(x) = -\frac{f(x)}{f'(x)} + x.$$

Тогда итерационный процесс осуществляется по формуле:

$$x_{k+1} = x_k - \frac{f(x_k)}{f'(x_k)}, \quad k = 0, 1, \ldots. \tag{1.8}$$

Геометрически этот процесс представлен на рисунке 1.2. Он означает замену на каждой итерации k графика кривой $y = f(x)$ касательной к ней в точках с координатами (x_k, y_k).

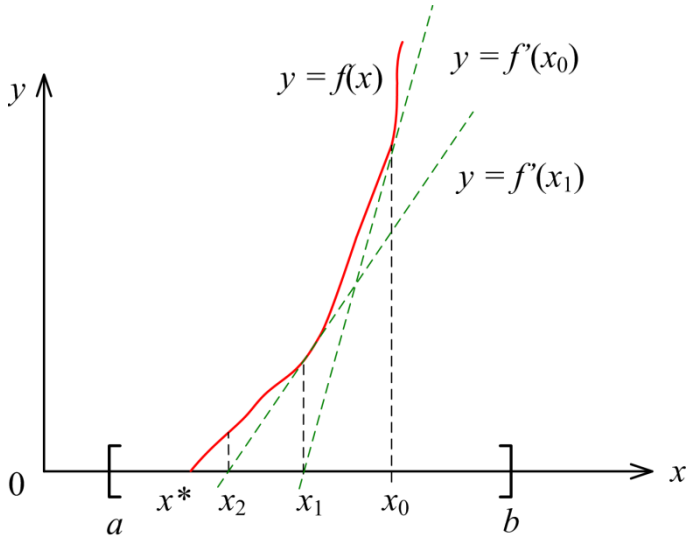


Рисунок 1.2 – Геометрическая интерпретация метода Ньютона

Достаточное условие сходимости обеспечивается выбором начальной точки x_0. Начальным приближением x_0 служит один из концов отрезка $[a,b]$, в зависимости от того, в каком из них выполняется достаточное условие сходимости

$$f(x_0) f''(x_0) > 0. \tag{1.9}$$

При произвольном начальном приближении итерации сходятся, если

$$|\varphi'(x)| = \left| \frac{f(x) \cdot f''(x)}{(f'(x))^2} \right| < 1 \quad \forall x \in [a,b]. \tag{1.10}$$

Метод Ньютона рекомендуется применять для нахождения простых действительных корней уравнения (1.1).

Достоинством метода является то, что он обладает скоростью сходимости, близкой к квадратичной.

Недостатки метода:

– не при любом начальном приближении метод Ньютона сходится, а лишь при таком, для которого $f'(x_0) \neq 0$;

– если $f'(x_k) \to 0$, т.е. касательная к графику почти параллельна оси абсцисс, то $x_{k+1} - x_k \to \infty$ и метод расходится;

– если $f'(x_n) \to \infty$, т.е. касательная к графику почти параллельна оси ординат, то $x_{k+1} - x_k \to 0$ и продвижения к корню не будет.

Последних трудностей можно избежать, применив модификацию метода Ньютона, в которой используется только касательная в точке начального приближения. Рабочая формула при этом имеет вид:

$$x_{k+1} = x_k - \frac{f(x_k)}{f'(x_0)}. \qquad (1.11)$$

1.3 Решение нелинейного уравнения методом простых итераций

Задание: найти приближенное численное решение нелинейного уравнения методом простых итераций с точностью $\varepsilon = 10^{-3}$:

$$e^x - \sin x - 3 = 0. \qquad (1.12)$$

Для приблизительной оценки корней нелинейного уравнения (1.12) построим график функции с помощью программы Matlab (рисунок 1.3).

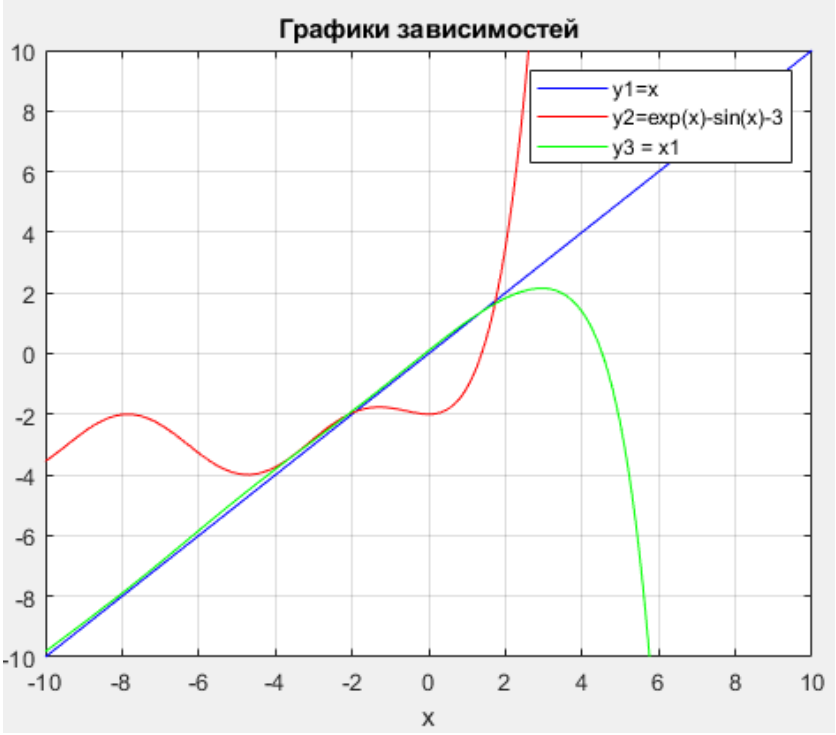


Рисунок 1.3 – График нелинейной функции $y = f(x)$

Из рисунка 1.3 видно, что график функции пересекает ось абсцисс в точке расположенной на отрезке [-4;0]. Начало и конец графика уходят далеко вниз, поэтому делаем вывод, что он имеет только видимые нам корни.

С помощью программы Matlab напишем код для решения нелинейного уравнения методом простых итераций.

Проект будет включать следующие файлы:

lab1_2_1.m – основной скрипт, в котором вводятся исходные данные, выполняется построение графика для отделения корней и выводится результат (листинг 1.1);

fun_D.m – расчет погрешности корня (листинг 1.2);

fun2.m – вычисление значение функции в текущей точке (листинг 1.3);

```
x0=-10:0.1:10;
y1 = x0;
y2 = exp(x0)-sin(x0)-3;
[t] = fun2(x0);
eps=1e-3;
kmax=50;
x = 1;
k = 0;
[x1] = fun2(x);
[D] = fun_D(x);
fprintf('%f\t %f \n', x1, D);
k = k+1;
while (abs(D)>eps)&&(k<kmax)
    x=x1;
    [x1]=fun2(x);
    [D] = fun_D(x);
    fprintf('%f\t %f \n', x1, D);
    k = k+1;
end
fprintf('Root: x =%f\n', x1);
fprintf('Number of iterations: k = %i\n', k);
fprintf('Accuracy: D =%f\n', D);
figure(1);
plot(x0,y1,'b',x0,y2,'r', x0, t, 'g');
axis([-10 10 -10 10]);
xlabel('x');
ylabel('y');
grid on;
legend('y1=x','y2=exp(x)-sin(x)-3', 'y3 = x1');
title ('Графики зависимостей');
```

Листинг 1.1 – Основной скрипт метода простых итераций

```
function [D] = fun_D(x)
M = -20;
F = exp(x)-sin(x)-3;
D = F/M;
end
```

Листинг 1.2 – Расчет погрешности

```
function [x1] = fun2(x)
M = -20;
F = exp(x)-sin(x)-3;
x1 = x+(F/M);
end
```

Листинг 1.3 – Вычисление значения функции

Результат работы программы представлен на рисунке 1.4.

```
Root: x = 1.377646
Number of iterations: k = 23
Accuracy: D = 0.000977
```

Рисунок 1.4 – Результат работы программы

В данном случае итерационный процесс сходится; корень уравнения x= 1,377646 найден с заданной точностью $\varepsilon = 0,001$ за 23 итерации, погрешность решения составляет 0,000977.

1.4 Решение нелинейного уравнения методом Ньютона

Задание: найти приближенное численное решение нелинейного уравнения методом Ньютона с точностью $\varepsilon = 10^{-3}$:

$$(-arctg\ x - 1{,}2)\ x = 0. \qquad (1.13)$$

Для приблизительной оценки корней нелинейного уравнения (1.13) построим график функции с помощью программы Matlab (рисунок 1.5).

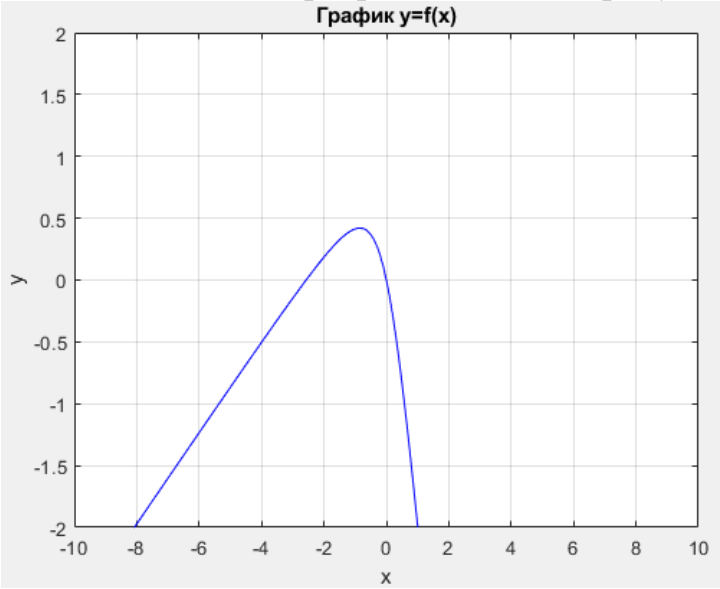


Рисунок 1.5 – График нелинейной функции $y = f(x)$

График функции пересекает ось абсцисс в двух точках, расположенных на отрезке [-4;0]. Начало и конец графика уходят далеко вниз, поэтому делаем вывод, что он имеет только видимые нам корни.

Так как из самого уравнения, очевидно, что второй корень равен нулю, находить потребуется только первый. С помощью программы Matlab напишем код для решения нелинейного уравнения методом Ньютона.

Проект будет включать следующие файлы:

lab1_1.m – основной скрипт метода Ньютона, в котором вводятся исходные данные, выполняется построение графика для отделения корней и выводится результат (листинг 1.4);

Newton.m – функция вычисление текущей точки методом Ньютона (листинг 1.5);

fun1.m – расчет погрешности корня (листинг 1.6).

```
x = -10:0.1:10;
y = (-atan(x)-1.2).*x;
figure(1);
plot(x,y,'-b');
axis([-10 10 -2 2]);
xlabel('x');
ylabel('y');
```

Листинг 1.4, лист 1– Основной скрипт

```
grid on;
title ('График y=f(x)');
eps=1e-3;
kmax=50;
x=-4;
k=0;
[F, F1]=fun1(x);
[x1,D]=Newton(x,F,F1);
k=k+1;
fprintf('%i\t %f\t %f \n', k, x1, D);
while (abs(D)>eps)&&(k<kmax)
    x=x1;
    [F, F1]=fun1(x);
    [x1,D]=Newton(x,F,F1);
    k=k+1;
    fprintf('%i\t %f\t %f \n', k, x1, D);
end
fprintf('Root: x =%f\n', x1);
fprintf('Number of iterations: k = %i\n', k);
fprintf('Accuracy: D =%f\n', D);
```

Листинг 1.4, лист 2

```
function [ x1, D ] = Newton(x, F, F1)
 D=-F/F1;
 x1=x+D;
end
```

Листинг 1.5 – Функция метода Ньютона

```
function [F, F1] = fun1(x)
F=(-atan(x)-1.2).*x; %расчет ф-ции
F1= -(x/(1+x^2))-atan(x)-1.2; %1 производная
end
```

Листинг 1.6 – Функция расчета значения и ее первой производной

Результат работы программы представлен на рисунке 1.6.

```
>> lab1_1
1     -2.606330    1.393670
2     -2.572209    0.034121
3     -2.572152    0.000058
Root: x =-2.572152
Number of iterations: k = 3
Accuracy: D =0.000058
```

Рисунок 1.6 – Результат работы программы

Корень уравнения x = 0,4816 найден с заданной точностью $\varepsilon = 0,001$ за 3 итерации, погрешность решения составляет 0.000058.

1.5 Вывод

Были рассмотрены два метода решения нелинейных уравнений: Ньютона и простых итераций.

Метод простых итераций основан на итерационном приближении с использованием значения функции, коэффициента приближения и значения. Такой метод обладает линейной сходимостью, но при модуле производной приближения больше 1, метод расходится, что является недостатком данного метода.

Метод Ньютона использует для приближения первую производную, что увеличивает скорость сходимости до квадратичной. Однако, метод имеет ряд ограничений, что не делает его таким универсальным.

2 Интерполяция функций

Пусть функция $f(x)$ задана таблицей значений x_i, y_i на интервале $[a; b]$:

$$y_i = f(x_i), \quad i = \overline{0,n}, \quad a \le x_i \le b. \tag{2.1}$$

Задача интерполяции – найти функцию $F(x)$, принимающую в точках x_i те же значения y_i.

Условие интерполяции:

$$F(x_i) = y_i \ . \tag{2.2}$$

При этом предполагается, что среди значений x_i нет одинаковых. Точки x_i называют узлами интерполяции.

Если $F(x)$ ищется только на отрезке $[a; b]$, то это задача интерполяции, а если за пределами первоначального отрезка, то это задача экстраполяции. Таким образом, можно ввести следующие определения:

интерполяция (в узком смысле) – вычисление промежуточных значений функции по известному дискретному набору значений функции на заданном интервале;

экстраполяция – вычисление значений функции за пределами первоначально известного интервала;

аппроксимация – определение в явном виде параметров функции, описывающей распределение точек.

Задача нахождения интерполяционной функции $F(x)$ имеет много решений, так как через заданные точки x_i, y_i можно провести бесконечно много кривых, каждая из которых будет графиком функции, для которой выполнены все условия интерполяции. Для практики важен случай аппроксимации функции многочленами:

$$F(x) = P_m(x_i) = a_0 + a_1 \cdot x + a_2 \cdot x^2 + \ldots + a_m \cdot x^m, \quad i = 0,1,\ldots,m. \tag{2.3}$$

При этом искомый полином называется интерполяционным полиномом.

При построении одного многочлена для всего рассматриваемого интервала $[a; b]$, для нахождения коэффициентов многочлена необходимо использовать все уравнения системы (2.3). Данная система содержит n+1 уравнение, следовательно, с ее помощью можно определить n+1 коэффициент. Поэтому максимальная степень интерполяционного многочлена $m = n$, и многочлен принимает вид:

$$P_n(x_i) = a_0 + a_1 \cdot x + a_2 \cdot x^2 + \ldots + a_n \cdot x^n, \quad i = 0,1,\ldots,n \tag{2.4}$$

2.1 Локальная и глобальная интерполяция

Если задан n+1 узел интерполяции, то на этих узлах можно построить один интерполяционный многочлен n-ой степени, n-1 многочленов первой степени и большой набор многочленов степени меньше n, опирающиеся на некоторые из этих узлов.

Теоретически максимальную точность обеспечивает многочлен более высокой степени. Однако на практике наиболее часто используют многочлены невысоких степеней, во избежание погрешностей при расчетах коэффициентов при больших степенях многочлена, а также из-за возможных колебаний функции между узлами.

Если функция $f(x)$ интерполируется на отрезке $[a;b]$ с помощью единого многочлена $P_m(x)$ для всего отрезка, то такую интерполяцию называют глобальной. В случае локальной интерполяции на каждом интервале $[x_i; x_{i+1}]$ строится отдельный интерполяционный полином невысокой степени.

2.2 Кусочно-линейная интерполяция

Простейшим и часто используемым видом локальной интерполяции является линейная (или кусочно-линейная) интерполяция. Она заключается в том, что узловые точки соединяются отрезками прямых (рисунок 2.1), то есть через каждые две точки (x_i, y_i) и (x_{i+1}, y_{i+1}) проводится полином первой степени:

$$F(x) = a_0 + a_1 \cdot x, \quad x_i \le x \le x_{i+1} \tag{2.5}$$

Коэффициенты a_0 и a_1 разные на каждом интервале $[x_i; x_{i+1}]$ и находятся из выполнения условий интерполяции на концах отрезка:

$$\begin{cases} y_i = a_0 + a_1 \cdot x_i ; \\ y_{i+1} = a_0 + a_1 \cdot x_{i+1}. \end{cases} \tag{2.6}$$

Из системы уравнений (2.6) можно найти коэффициенты a_0 и a_1 и получить уравнение прямой на каждом интервале.

Линейная интерполяция является частным случаем многочлена Лагранжа, построенного по двум точкам, поэтому вместо прямого вычисления коэффициентов можно воспользоваться формулой:

$$F(x) = y_i \frac{x - x_{i+1}}{x_i - x_{i+1}} + y_{i+1} \frac{x - x_i}{x_{i+1} - x_i} \ . \tag{2.7}$$

При использовании кусочно-линейной интерполяции сначала нужно определить интервал i, в который попадает расчетное значение x, а затем в выражение (2.7) подставить значения (x_i, y_i) и (x_{i+1}, y_{i+1}) для данного интервала.

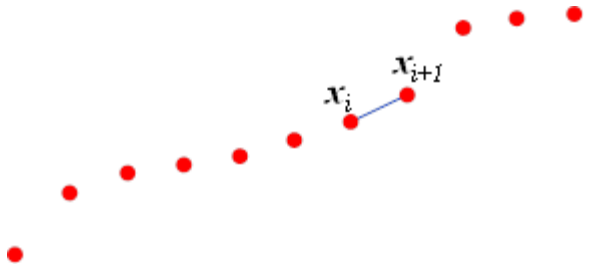


Рисунок 2.1 – Кусочно-линейная интерполяция

2.3 Кусочно-квадратичная интерполяция

В случае квадратичной интерполяции, для каждых трех узловых точек (x_i, y_i), (x_{i+1}, y_{i+1}), (x_{i+2}, y_{i+2}) строится уравнение параболы (рисунок 2.2):

$$F(x) = a_0 + a_1 \cdot x + a_2 \cdot x^2, \quad x_i \le x \le x_{i+2} \tag{2.8}$$

Здесь коэффициенты a_0, a_1 и a_2 разные на каждом интервале $[x_i, x_{i+2}]$ и определяются решением системы уравнений для условия прохождения параболы через три точки:

$$\begin{cases} y_i = a_0 + a_1 \cdot x_i + a_2 \cdot x_i^2; \\ y_{i+1} = a_0 + a_1 \cdot x_{i+1} + a_2 \cdot x_{i+1}^2; \\ y_{i+2} = a_0 + a_1 \cdot x_{i+2} + a_2 \cdot x_{i+2}^2. \end{cases} \tag{2.9}$$

Квадратичная интерполяция также является частным случаем многочлена Лагранжа, построенного по трем точкам, поэтому вместо прямого вычисления коэффициентов можно воспользоваться формулой:

$$F(x) = y_i \frac{(x-x_{i+1})(x-x_{i+2})}{(x_i-x_{i+1})(x_i-x_{i+2})} + y_{i+1} \frac{(x-x_i)(x-x_{i+2})}{(x_{i+1}-x_i)(x_{i+1}-x_{i+2})} + y_{i+2} \frac{(x-x_i)(x-x_{i+1})}{(x_{i+2}-x_i)(x_{i+2}-x_{i+1})} \tag{2.10}$$

При использовании кусочно-линейной интерполяции сначала нужно определить интервал i, в который попадает расчетное значение x, а затем в выражение (2.10) подставить значения $(x_i, y_i), (x_{i+1}, y_{i+1}), (x_{i+2}, y_{i+2})$ для данного интервала.

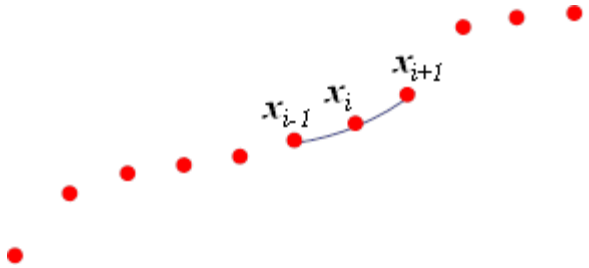


Рисунок 2.2 – Кусочно-квадратичная интерполяция

2.4 Кубические сплайны

Кубическим сплайном называется функция $S(x)$, которая удовлетворяет следующим требованиям:

на каждом i-ом локальном отрезке интерполяции между соседними заданными узлами $[x_i, x_{i+1}], i = \overline{1,n}$ она является многочленом третьей степени;

имеет непрерывные первую и вторую производные на всём интервале интерполяции $[x_1, x_{n+1}]$;

во всех узлах интерполяции x_i выполняются условия равенства значений исходной табличной функции $f(x)$ и функции $S(x)$, а также их первых производных: $S(x_i) = f(x_i)$ и $S'(x_i) = f'(x_i)$;

выполняются граничные условия для вторых производных: $S''(x_1) = S''(x_{n+1}) = 0$ (в этом случае существует единственный такой сплайн).

Кубические сплайны делятся на два класса.

Локальные. Используются первые производные $f'(x)$, вычисляемые по трем ближайшим к точке *x* узлам интерполяции. Локальные сплайны ненамного лучше кусочно-линейной или кусочно-квадратичной интерполяции и используются редко.

Глобальные. Основаны на вычислении массива вторых производных $f''(x_i)$ по всем узлам x_i, который рассчитывается заранее путем решения единой трехдиагональной СЛАУ. Такие сплайны достаточно хороши и являются одним из основных методов интерполяции. В отличие от кусочно-линейной и кусочно-квадратичной интерполяции они дают гладкие кривые за счет непрерывности первой и второй производных. К недостаткам относится то, что на каждом интервале расчетная формула своя.

Пусть имеется таблица значений $f(x_i)$, где $i = \overline{1,(n+1)}$, то есть у нас есть $(n+1)$ узел интерполяции и *n* интервалов между ними. В общем случае шаг сетки узлов может быть неравномерным.

Для нахождения значения сплайна в некоторой введенной точке необходимо сначала проверить, принадлежит ли данная точка общему интервалу интерполяции $[x_1, x_{n+1}]$. Если принадлежит, то нужно определить, между какими заданными узлами находится введенная точка, то есть найти номер *i* подынтервала $x \in [x_i, x_{i+1}], i = \overline{1,n}$.

Формула вычисления значения сплайна в точке имеет вид:

$$F(x) = \frac{1}{6h_i}\left[m_i(x_{i+1}-x)^3 + m_{i+1}(x-x_i)^3\right] + \frac{1}{h_i}\left[\left(f_i - \frac{m_i h_i^2}{6}\right)(x_{i+1}-x) + \left(f_{i+1} - \frac{m_{i+1}h_i^2}{6}\right)(x-x_i)\right], \quad (2.11)$$

где $h_i = x_{i+1} - x_i, \quad i = \overline{1,n}$ – расстояния между соседними узлами;

$m_i = f''(x_i), \quad i = \overline{1,(n+1)}$ – значения вторых производных в узловых точках.

Вторые производные m_i находятся из трех диагональной СЛАУ:

$$h_i m_i + 2(h_i + h_{i+1})m_{i+1} + h_{i+1}m_{i+2} = 6\left(\frac{f_{i+2}-f_{i+1}}{h_{i+1}} - \frac{f_{i+1}-f_i}{h_i}\right), \quad i = \overline{1,(n-1)}. \quad (2.12)$$

Всего в этой системе $(n-1)$ уравнение и $(n+1)$ неизвестных вторых производных $m_1, m_2, \ldots, m_{n+1}$, поэтому система не полностью определяет m_i. Сведем ее к трехдиагональной СЛАУ заданием граничных условий. Для

нормального сплайна $m_1 = m_{n+1} = 0$. Теперь в СЛАУ $(n-1)$ неизвестная $m_2, m_3, \ldots, m_n$ и она является трехдиагональной. Такая система эффективнее всего решается методом прогонки.

Если введенная точка не принадлежит общему интервалу интерполяции $[x_1, x_{n+1}]$, то для расчета значений функции за его пределами можно воспользоваться методами экстраполяции.

Простейший способ – использовать линейную экстраполяцию. В этом случае асимптотическое поведение функции вне интервала описывается формулами линейной экстраполяции влево (2.13) или вправо (2.14):

$$F(x) = f_1 + \left[-\frac{x_2 - x_1}{6} m_2 + \frac{f_2 - f_1}{x_1 - x_0} \right](x - x_1), \quad x < x_1; \tag{2.13}$$

$$F(x) = f_{n+1} + \left[\frac{x_{n+1} - x_n}{6} m_n + \frac{f_{n+1} - f_n}{x_{n+1} - x_n} \right](x - x_{n+1}), \; x > x_{n+1}. \tag{2.14}$$

2.5 Многочлен Лагранжа

Пусть задана система точек $x_i, i = \overline{1, n+1}$, в которых известны значения функции $y_i = f(x_i)$.

Рассмотрим пример построения интерполяционного многочлена Лагранжа по заданной системе точек (в общем случае для неравноудаленных аргументов). Построим некоторый многочлен $P_n(x)$ таким образом, чтобы его значения совпали со значениями функции, заданными в таблице, для тех же аргументов, то есть $P_n(x_i) = y_i, \quad i = \overline{1, n+1}$. Лагранж предложил строить многочлен степени *n* в виде:

$$\begin{aligned} P_n(x) = a_1(x - x_2)\ldots(x - x_{n+1}) + a_2(x - x_1)(x - x_3)\ldots(x - x_{n+1}) + \ldots + \\ + a_{n+1}(x - x_1)\ldots(x - x_n). \end{aligned} \tag{2.15}$$

Здесь в каждом слагаемом коэффициенту a_i соответствует разность $x - x_i$. Найдем неизвестные коэффициенты $a_i, i = \overline{1, n+1}$, называемые коэффициентами Лагранжа, используя условие $P_n(x) = y_i, \quad i = \overline{1, n+1}$.

При $x = x_1$:

$$P_n(x_1) = y_1;$$

$$P_n(x_1) = a_1(x_1 - x_2)(x_1 - x_3)\ldots(x_1 - x_{n+1}) = y_1.$$

Следовательно, коэффициент a_1 вычисляется по следующей формуле:

$$a_1 = \frac{y_1}{(x_1 - x_2)(x_1 - x_3)\ldots(x_1 - x_{n+1})}.$$

При $x = x_2$:

$$P_n(x_2) = y_2;$$

$$P_n(x_2) = a_2(x_2 - x_1)(x_2 - x_3)\ldots(x_2 - x_{n+1}) = y_2.$$

Следовательно, коэффициент a_1 вычисляется по формуле:

$$a_2 = \frac{y_2}{(x_2 - x_1)(x_2 - x_3)\ldots(x_2 - x_{n+1})}.$$

Значения остальных коэффициентов вычисляются аналогично.

С учетом найденных коэффициентов интерполяционный многочлен Лагранжа запишется в виде:

$$P_n(x) = y_1 \frac{(x - x_2)(x - x_3)\times\ldots\times(x - x_{n+1})}{(x_1 - x_2)\times\ldots\times(x_1 - x_{n+1})} + \ldots + y_{n+1} \frac{(x - x_1)(x - x_2)\times\ldots\times(x - x_n)}{(x_{n+1} - x_1)\times\ldots\times(x_{n+1} - x_n)}. \quad (2.16)$$

Погрешность формулы (2.16) определяется остаточным членом:

$$R_n(x) = \frac{f^{n+1}(\xi)}{(n+1)!}(x - x_1)\ldots(x - x_{n+1}), \quad (2.17)$$

где ξ – точка наименьшего промежутка, содержащего все узлы x_i, $i = \overline{1, n+1}$ и точку x.

Используя условные обозначения суммы и произведения, формулу (2.16) можно записать в сокращенном виде:

$$P_n(x) = \sum_{i=1}^{n+1} \left[y_i \cdot \prod_{\substack{j=1, \\ j \neq i}}^{n+1} \frac{x - x_j}{x_i - x_j} \right]. \quad (2.18)$$

В отличие от кусочно-линейной, кусочно-квадратичной интерполяции и кубического сплайна, формула многочлена Лагранжа будет общая для всего интервала $[x_1, x_{n+1}]$, поэтому определять номер интервала $[x_i, x_{i+1}]$ для каждой расчетной точки не нужно.

2.6 Кусочно-квадратичная интерполяция функции по таблице значений

Задание: с помощью кусочно-квадратичной интерполяции построить график приближенной функции, проходящей через экспериментальные точки, представленные в таблице 2.1, и вычислить значения неизвестной функции в промежуточных точках.

Таблица 2.1 – Экспериментальные данные для интерполяции

		Экспериментальные точки									Точки вычисления *f*(*x*)							
1	*x*	-5,5	-3,5	-1	2	4,5	5,5	6,8	7,7		-8	-7	-5	-0,5	2	3	7	9
	y	2,75	2,1	1,6	2,2	4	5,5	8,5	13									
2	*x*	-10	-6,5	-4	-2	-1,5	-0,5	2	4	7,5	1	1,5	5	7,5	11	13	16,5	
	y	-1	3	5	4	0	-2,5	0	3	5								

С помощью среды Matlab напишем программу для интерполяции.

lab4_1.m – главный файл работы программы, в котором вызываются функции интервала, расчета точек и выводятся график и результат вычислений (листинг 2.1);

interval.m – нахождение интервала между расчетными точками (листинг 2.2);

square_val.m – функция расчета промежуточных точек (листинг 2.3);

Gauss.m – функция, находящая корни системы уравнения методом Гаусса (листинг 2.4);

fun_1.m – замена строк местами для решения системы уравнений (листинг 2.5);

```
table = 1; % или 2
[x0, y0, x1] = variant(27,table);
itr = interval(x0, x1);
y1 = square_val(x0,y0,x1,itr);
disp(x1);
disp(y1);
a = min(x0);
b = max(x0);
step = 0.1;
x2 = a:step:b; %
itr = interval(x0,x2);
y2 = square_val(x0,y0,x2,itr);
figure(1);
plot(x0,y0, 'o' , x1,y1, '*', x2,y2, 'r');
axis([a-1 b+1 min([y0 y1' y2'])-1 max([y0 y1' y2'])+1]);
xlabel('x');
ylabel('y');
grid on;
legend('Заданные точки','Расчетные точки','Интерполяционная
функция');
```

Листинг 2.1 – Основной скрипт кусочно-квадратичной интерполяции

```
function [itr] = interval(x, x1)
n = length(x);
n1 = length(x1);
itr = zeros(n1, 1);
k = 1;
for i = 1:
    if  (x1(i) < x(1))
          itr(i) = 0;
    else
       for j = k:n-1 % k = 6
           if (x(j)<=x1(i) && x1(i)<=x(j+1))
               itr(i)= j;
               k = j;
           end
       end
       if x1(i)>x(n)
          itr(i)=n;
       end
   end
end
end
```

Листинг 2.2 – Интервал между расчетными точками

```
function [y1] = square_val(x,y,x1,itr)
n = length(x);
n1 = length(x1);
y1 = zeros(n1,1);
a = ones(n,1);
b = ones(n,1);
c = ones(n,1);
for i = 1:(n-2)
    X = [x(i)^2 x(i) 1;
        x(i+1)^2 x(i+1) 1;
        x(i+2)^2 x(i+2) 1];
    Y = [y(i); y(i+1); y(i+2)];
    C = Gauss(X, Y);
    a(i) = C(1);
    b(i) = C(2);
    c(i) = C(3);
end
disp(X);
disp(Y);
disp(C);
a(n-1) = a(n-2);
b(n-1) = b(n-2);
c(n-1) = c(n-2);
for i = 1:n1
    j = itr(i);
    if j==0
        y1(i)=NaN;
    else
        if (0<j && j<=n-1)
            y1(i) = a(j)*x1(i)^2+b(j)*x1(i)+c(j);
        else
            if j==n
                y1(i)=NaN;
            end
        end
    end
end
end
```

Листинг 2.3 – Расчет промежуточных точек

```
function [X] = Gauss(A0, b)
eps = 0.001;
A = [A0 b];
n = 3;
for k = 1:n
    if abs(A(k,k)) < eps
        [A] = fun_1(A, k);
    end
    for j = (n+1):(-1):k
        A(k,j) = A(k,j)/A(k,k);
    end
    for i = (k+1):n
```

Листинг 2.4, лист 1 – Реализация метода Гаусса

```
        for j = (n+1):-1:k
            A(i,j) = A(i,j) - A(i,k)*A(k,j);
        end
    end
end
for i = (n):-1:1
    X(i) = A(i,n+1);
    for j = (i+1):n
        X(i) = X(i) - A(i,j)*X(j);
    end
end
end
```

Листинг 2.4, лист 2

```
function [A] = fun_1(A, k)
t = A (k);
A(k) = A(k+1);
A(k+1) = t;
end
```

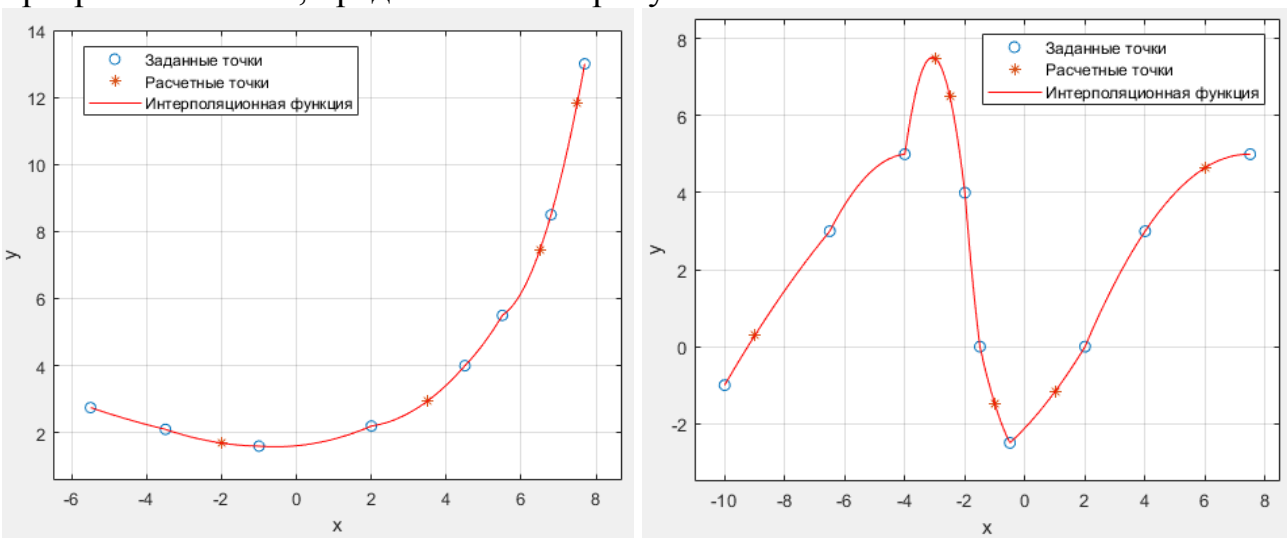
Листинг 2.5 – Замена строк данных уравнения

Значения функции в расчетных точках приведены в таблице 2.2.

Таблица 2.2 – Значения функции кусочно-квадратичной интерполяции

		Точки вычисления $f(x)$						
1	x	-10	-2	3,5	6,5	7,5	8	
	y	NaN	1,6909	2,9457	7,4406	11,8287	NaN	
2	x	-9	-3	-2,5	-1	1	6	9
	y	0,2857	7,5	6,5	-1,5	-1,1667	4,6494	NaN

Графики для первой и второй функции, построенные с помощью программы Matlab, представлены на рисунке 2.3.



а б

Рисунок 2.3 – Квадратичная интерполяция для первой (а) и второй (б) функции

2.7 Интерполяция функции кубическим сплайном по таблице значений

Задание: с помощью интерполяции кубическим сплайном построить график приближенной функции, проходящей через экспериментальные точки, представленные в таблице 2.1, и вычислить значения неизвестной функции в промежуточных точках.

С помощью среды Matlab напишем программу для интерполяции.

lab_5.m – основной скрипт, в котором вводятся исходные данные, вызываются функции и выводятся график и результат (листинг 2.6);

interval.m – функция нахождения интервала (листинг 2.2);

spline_val.m – вычисление значений в расчетных точках (листинг 2.7);

progon.m – вычисление вторых производных в расчетных точках (листинг 2.8).

```
table = 1; % или 2
[x, y, x1] = variant(27,table);
M = progon (x, y);
itr = interval(x, x1);
y1 = spline_val(x,y,x1,itr,M);
disp(x1);
disp(y1);
a = min([x x1]);
b = max([x x1]);
step = 0.1;
x2 = a:step:b;
itr = interval(x,x2);
y2 = spline_val(x,y,x2,itr, M);
figure(1);
plot(x,y, 'o' , x1,y1, '*', x2,y2, 'r');
axis([a-1 b+1 min([y y1 y2])-1 max([y y1 y2])+1]);
xlabel('x');
ylabel('y');
grid on;
legend('Заданные точки','Расчетные точки','Интерполяционная
функция');
```

Листинг 2.6 – Основной код интерполяции кубическим сплайном

```
function [ M ]= progon(x, y)
% входные аргументы: (x,y) - таблица заданных узлов
% выходные аргументы: M - массив вторых производных в узловых
точках
 n = length(x)-1;              % число интервалов
 h(1:n)=x(2:n+1)-x(1:n);       % длины интервалов
 % формирование трехдиагональной СЛАУ
 A(1)=0; A(2:n-1)=h(2:n-1);    % нижняя диагональ СЛАУ
 B(1:n-1)=2*(h(1:n-1)+h(2:n)); % главная диагональ СЛАУ
 C(1:n-2)=h(2:n-1); C(n-1)=0;  % верхняя диагональ СЛАУ
 D=zeros(1:n-1);               % правая часть СЛАУ
 for i=1:n-1
  D(i)=6*((y(i+2)-y(i+1))/h(i+1)-(y(i+1)-y(i))/h(i));
 end
 % прямой ход метода прогонки
```

Листинг 2.7, лист 1 – Метод прогонки

```
 Q=zeros(1,n);
 R=zeros(1,n);
 for i=1:n-1
  Q(i+1)=-(C(i)/(B(i)+A(i)*Q(i)));
  R(i+1)=(D(i)-A(i)*R(i))/(B(i)+A(i)*Q(i));
 end
 % обратный ход метода прогонки
 M=zeros(1,n-1);
 M(n-1)=R(n);
 for i=n-2:-1:1
  M(i)=Q(i+1)*M(i+1)+R(i+1);
 end
 % дополняем массив производных краевыми условиями M(a)=M(b)=0
 M = [0, M, 0];
end
```

Листинг 2.7, лист 2

```
function [y1] = spline_val(x,y,x1,itr,M)
n = length(x)-1;
n1 = length(x1);
y1 = zeros(1,n1);
h(1:n) = x(2:n+1) - x(1:n);
for i = 1:n1
    j = itr(i);
    if (j==0)
        y1(i) = y(1) + ((x(1)-x(2))*M(2)/6 + (y(2)-y(1))/(x(2)-
x(1))) * (x1(i)-x(1));
    else
        if (0 < j && j <= n)
            y1(i) = (1/(6*h(j))) * ((M(j)*(x(j+1)-x1(i))^3) +
M(j+1)*(x1(i)-x(j))^3) + (1/h(j)) * ((y(j)-((M(j)*h(j)^2)/6)) *
(x(j+1)-x1(i)) + (y(j+1)-((M(j+1)*h(j)^2)/6)) * (x1(i)-(x(j))));
        else
            if (j == n+1)
                y1(i) = y(n+1) + ((x(n+1)-x(n))*M(n)/6 + (y(n+1)-
y(n))/(x(n+1)-x(n))) * (x1(i)-x(n+1));
            end
        end
    end
end
end
```

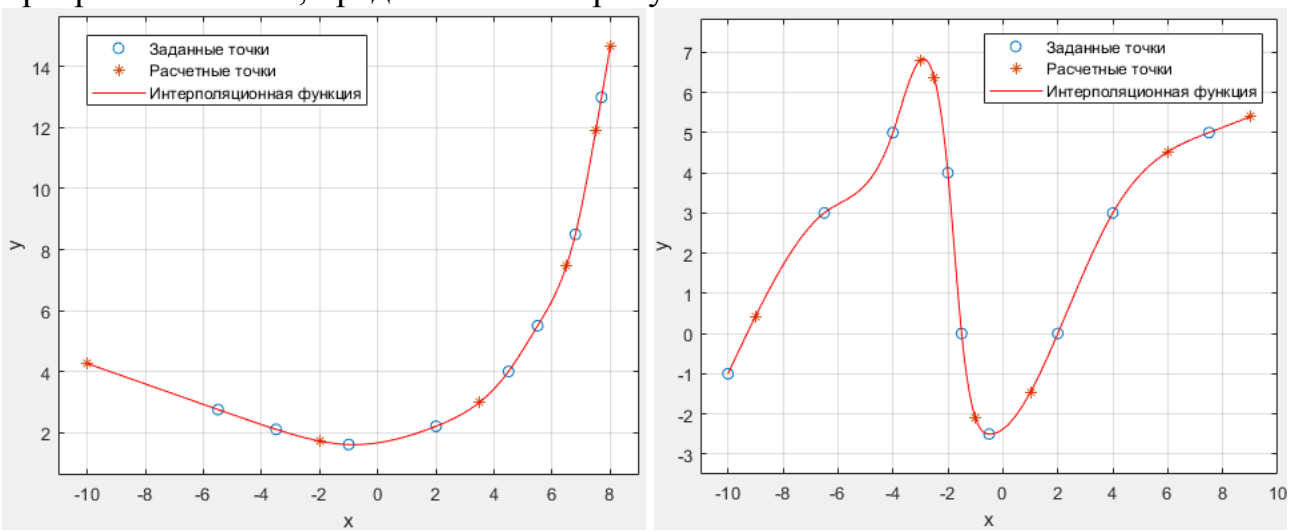
Листинг 2.8 – Расчет значений в точках

Значения функции в расчетных точках приведены в таблице 2.3.

Таблица 2.3 – Значения функции при интерполяции кубическим сплайном

		Точки вычисления $f(x)$						
1	x	-10	-2	3,5	6,5	7,5	8	
	y	4,2589	1,7137	2,9876	7,4878	11,8941	14,6671	
2	x	-9	-3	-2,5	-1	1	6	9
	y	0,4253	6,8045	6,3732	-2,1005	-1,4624	4,5206	5,3944

Графики для первой и второй функции, построенные с помощью программы Matlab, представлены на рисунке 2.4.



а б

Рисунок 2.4 – Интерполяция сплайном для первой (а) и второй (б) функции

2.8 Интерполяция функции многочленом Лагранжа по таблице значений

Задание: с помощью интерполяции многочленом Лагранжа построить график приближенной функции, проходящей через экспериментальные точки, представленные в таблице 2.1, и вычислить значения неизвестной функции в промежуточных точках.

С помощью среды Matlab напишем программу для интерполяции.

lab6_2.m – основной скрипт, в котором вызываются функции для расчета точек и выводится график результата (листинг 2.9);

lagr_val.m – расчет точек методом Лагранжа (листинг 2.10).

```
table = 1;
[x, y, x1] = variant(27,table);
y1 = lagr_val(x,y,x1);
disp(x1);
disp(y1);
a = min([x x1]);
b = max([x x1]);
step = 0.1;
x2 = a:step:b;
y2 = lagr_val(x,y,x2);
figure(1);
plot(x,y, 'o' , x1,y1, '*', x2,y2, 'r');
axis([a-1 b+1 min([y y1 y2])-1 max([y y1 y2])+1]);
xlabel('x');
ylabel('y');
grid on;
legend('Заданные точки','Расчетные точки','Интерполяционная
функция');
```

Листинг 2.9 – Основной скрипт для метода Лагранжа

```
function [y1] = lagr_val(x,y,x1)
n = length(x);
n1 = length(x1);
y1 = zeros(1,n1);
for k = 1:n1
    if ((x1(k) < x(1)) || (x1(k) > x(n)))
        y1(k) = NaN;
    else
        for i = 1:n
            P = 1;
            for j = 1:n
                if (i ~= j)
                    P = P*(x1(k)-x(j))/(x(i)-x(j));
                end
            end
            y1(k) = y1(k)+P*y(i);
        end
    end
end
end
```

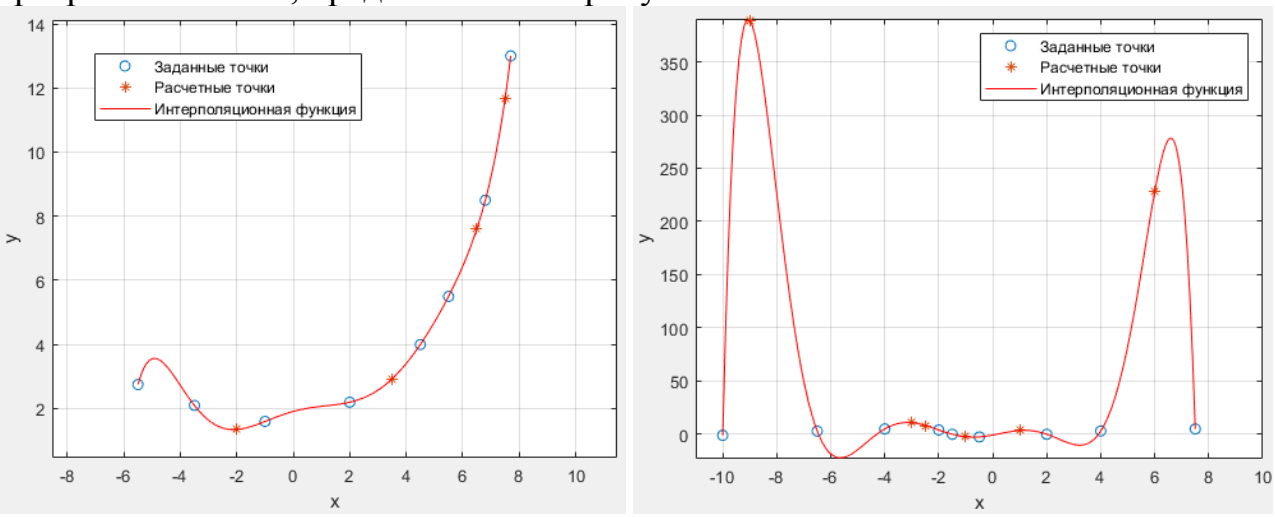
Листинг 2.10 – Функция расчета точек методом Лагранжа

Значения функции в расчетных точках приведены в таблице 2.4.

Таблица 2.4 – Значения функции при интерполяции многочленом Лагранжа

		Точки вычисления $f(x)$						
1	x	-10	-2	3,5	6,5	7,5	8	
	y	NaN	1,3493	2,9297	7,6049	11,6610	NaN	
2	x	-9	-3	-2,5	-1	1	6	9
	y	389,0437	10,8726	8,2426	-2,3617	3,5280	228,0141	NaN

Графики для первой и второй функции, построенные с помощью программы Matlab, представлены на рисунке 2.5.



а б

Рисунок 2.5 – Многочлен Лагранжа для первой (а) и второй (б) функции

2.9 Вывод

Таким образом, были просмотрены три вида интерполяции. На основе заданных значений можно сделать следующие выводы:

Самым точным методом оказался метод кубического сплайна. Он является многочленом третьей степени, имеет непрерывные первую и вторую производные на всём интервале интерполяции. Это позволяет получит сглаженный график.

Квадратичная интерполяция использует уравнения парабол, беря в учет 3 известных значения. Это метод никак не учитывает производные, из-за чего дает более высокую погрешность. Также график получается не целостным, т.е. видны резкие изменения направления.

Интерполяция многочленом Лагранжа является более простой в понимании и реализации, потому что используется одна форума, хоть весьма загроможденная, на всем интервале интерполяции. Однако, повышение степени многочлена приводит к нежелательным колебаниям графика, которые особенно заметны на краях.

3 Решение обыкновенных дифференциальных уравнений

Дифференциальные уравнения являются основным математическим инструментом моделирования и анализа разнообразных явлений и процессов в науке и технике.

Методы их решения подразделяются на два класса:

аналитические методы, в которых решение получается в виде аналитических функций;

численные (приближенные) методы, где искомые интегральные кривые получают в виде таблиц их численных значений.

Применение аналитических методов позволяет исследовать полученные решения методами математического анализа и сделать соответствующие выводы о свойствах моделируемого явления или процесса. К сожалению, с помощью таких методов можно решать достаточно ограниченный круг реальных задач. Численные методы позволяют получить с определенной точностью приближенное решение практически любой задачи.

Решить дифференциальное уравнение

$$\frac{dy}{dx} = f\left(x, y\right). \tag{3.1}$$

численным методом означает, что для заданной последовательности аргументов x_0 , x_1 , x_2 , . . . , x_n и числа $y_0 = y(x_0)$, не определяя аналитического вида функции $y = F\left(x\right)$, найти значения y_1 , y_2 , . . . , y_n, удовлетворяющие начальным условиям:

$$F\left(x_0\right) = y_0 \ , \quad y_k = F\left(x_k\right) \ , \ k = \left(\ \overline{1 \, , \, n} \ \right).$$

3.1 Метод Эйлера

Этот метод является сравнительно грубым и применяется в основном для ориентировочных расчетов. Однако идеи, положенные в основу метода Эйлера, являются исходными для ряда других численных методов.

Пусть дано обыкновенное дифференциальное уравнение (ОДУ) с начальными условиями (задача Коши):

$$y' = f\left(x, y\right), \ y\left(x_0\right) = y_0. \tag{3.2}$$

и удовлетворяются условия существования и единственности решения.

Требуется найти решение $y(x)$ задачи Коши (3.2) на отрезке $\left[a, b\right]$. Найдем решение в виде таблицы $y(x_i) = y_i, i = \overline{1,n}$. Для этого разобьем отрезок $\left[a, b\right]$ на *n* равных частей и построим последовательность

$$x_0, x_1 \ , \ . \ . \ . \ , x_n \ , x_i = x_0 + i \cdot h, \quad i = \overline{0, n},$$

где $h = (b - a)/n$ – шаг интегрирования.

Проинтегрируем исходное уравнение на отрезке $\left[x_k, x_{k+1}\right]$:

$$\int_{x_k}^{x_{k+1}} f(x,y)dx = \int_{x_k}^{x_{k+1}} \frac{dy}{dx}dx = y(x)\Big|_{x_k}^{x_{k+1}} = y(x_{k+1}) - y(x_k) = y_{k+1} - y_k$$

Полученное соотношение можно переписать как

$$y_{k+1} = y_k + \int_{x_k}^{x_{k+1}} f(x,y)dx. \qquad (3.3)$$

Если считать подынтегральную функцию постоянной на участке $[x_k, x_{k+1}]$ и равной значению в начальной точке этого интервала $x = x_k$, то получим

$$\int_{x_k}^{x_{k+1}} f(x_k, y_k)dx = f(x_k, y_k) \cdot x \Big|_{x_k}^{x_{k+1}} = f(x_k, y_k)(x_{k+1} - x_k) = hf(x_k, y_k).$$

Подставляя полученный результат в формулу (3.3), получим основную расчетную формулу метода Эйлера:

$$y_{k+1} = y_k + h \cdot f(x_k, y_k), \quad k = \overline{0, n-1}. \qquad (3.4)$$

Так как для вычисления каждого последующего значения y_{k+1} требуется только одно значение y_k, полученное на предыдущем шаге, то метод Эйлера относят к одношаговым методам.

Вычисление значений $y_1, y_2, \ldots, y_n$ осуществляется с использованием формулы (3.4) следующим образом. По заданным начальным условиям $a = x_0$ и y_0, полагая $k = 0$ в выражении (3.4), вычисляется значение

$$y_1 = y_0 + h\, f(x_0, y_0). \qquad (3.5)$$

Далее, определяя значение аргумента x по формуле $x_1 = x_0 + h$, используя найденное значение y_1 и полагая в формуле (3.4) $k = 1$, вычисляем следующее приближенное значение интегральной кривой $y = F(x)$ как

$$y_2 = y_1 + h\, f(x_1, y_1). \qquad (3.6)$$

Поступая аналогичным образом при $k = \overline{2, n-1}$, определяем все остальные значения y_k, в том числе последнее значение

$$y_{k+1} = y_k + h\, f(x_k, y_k). \qquad (3.7)$$

которое соответствует значению аргумента $x_n = b$.

Погрешность метода Эйлера составляет $O(h^2)$.

Вычислив приближенные значения неизвестной функции в точках $(x_0, y_0), (x_1, y_1), \ldots, (x_n, y_n)$, для более гладкого представления искомой интегральной кривой $y = F(x)$ можно воспользоваться одним из методов интерполяции, например, кубическим сплайном.

3.2 Метод Эйлера-Коши

Метод Эйлера-Коши, как и классический метод Эйлера, относится к одношаговым методам, но каждый шаг включает два этапа.

На первом этапе выполняется приблизительная оценка значения функции по обычной формуле Эйлера:

$$y_{k+1}^{*} = y_k + h \cdot f\left(y_k, x_k\right). \tag{3.8}$$

На втором этапе полученное значение используется для более точного расчета по формуле:

$$y_{k+1} = y_k + \frac{h}{2} \cdot \left[f\left(y_k, x_k\right) + f\left(y_{k+1}^{*}, x_k + h\right) \right]. \tag{3.9}$$

Погрешность метода Эйлера составляет $O(h^3)$, что на порядок выше, чем у метода Эйлера.

3.3 Решение ОДУ методом Эйлера

Задание: найдите методом Эйлера численное решение ОДУ первого порядка на отрезке от a = 1,5 до b = 3,5 с шагом интегрирования h = 0,2:

$$\begin{cases} y' = y + 3 \cdot \left(\dfrac{1}{2\sqrt{x-1}} - \sqrt{x-1} - \cos x - \sin x \right) \\ y(1{,}5) = 2{,}334 \end{cases}$$

Получите таблицу приближенных значений неизвестной функции $y(x)$ на отрезке интегрирования $x \in [a; b]$ с постоянным шагом h.

Аппроксимируйте полученные табличные значения кубическим сплайном и постройте приближенный график функции $y(x)$ на отрезке интегрирования, используя скрипты из главы 2. При построении графика шаг интерполяции примите равным $0{,}1 \cdot h$.

С помощью среды Matlab напишем программу для интерполяции. В нее входят следующие файлы:

lab_7.m – основной скрипт, вызывающий функции для расчета точек методом Эйлера и выводящий их в виде графика (листинг 3.1);

EL.m – метод Эйлера (листинг 3.2);

а также функции для интерполяции кубическим сплайном, которые представлены выше.

```
h = 0.2;
x = 1.5:h:3.5;
y = EL(x);
disp(x);
disp(y);
M = progon (x, y);
x1 = min(x):0.1:max(x);
itr = interval(x, x1);
y1 = spline_val(x,y,x1,itr,M);
a = min([x x1]);
b = max([x x1]);
```

Листинг 3.1, лист 1 – Код для решения ОДУ методом Эйлера

```
figure(1);
plot(x,y, 'o' , x1,y1, 'r');
axis([a-0.5 b+0.5 min([y y1])-0.5 max([y y1])+0.5]);
xlabel('x');
ylabel('y');
grid on;
legend("Расчетные точки", "Интерполяция");
```

Листинг 3.1, лист 2

```
function [y] = EL (t)
n = length(t);
y = zeros(1,n);
y(1) = 2.334;
for i = 1:n-1
  y(i+1) = y(i) + 0.2*(y(i) + 3*(1/(2*sqrt(t(i)-1)) - sqrt(t(i)-1)
- cos(t(i)) - sin(t(i))));
end
end
```

Листинг 3.2 – Метод Эйлера

Численное решение ОДУ приведено в таблице 3.1.

Таблица 3.1 – Численное решение ОДУ методом Эйлера

k	1	2	3	4	5	6	7	8	9	10	11
x_k	1,5	1,7	1,9	2,1	2,3	2,5	2,7	2,9	3,1	3,3	3,5
y_k	2,3340	2,1599	1,9307	1,6901	1,4698	1,2951	1,1859	1,1568	1,2178	1,3735	1,6232

График приближенного решения ОДУ, построенный с помощью интерполяции кубическим сплайном, представлен на рисунке 3.1.

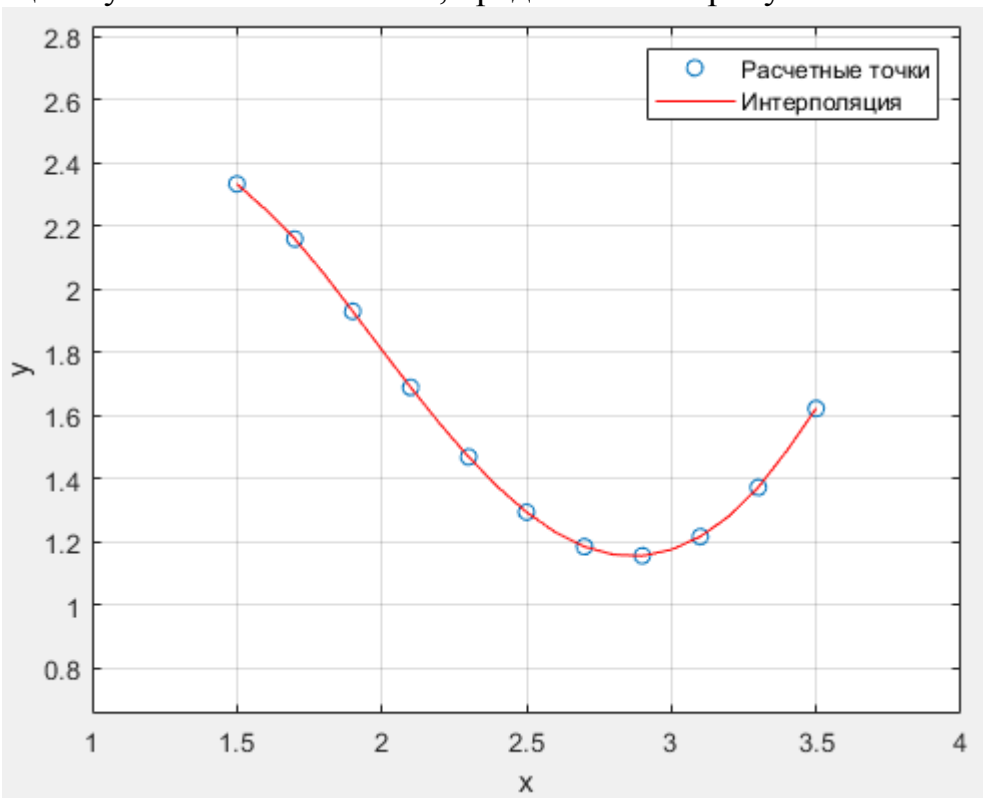


Рисунок 3.1 – График приближенного решения ОДУ методом Эйлера

3.4 Решение ОДУ методом Эйлера-Коши

Задание: найдите методом Эйлера-Коши численное решение ОДУ первого порядка на отрезке от $a = 2$ до $b = 4$ с шагом интегрирования $h = 0{,}2$:

$$\begin{cases} y' = (9x^2 - 1)e^{-x} - y \\ y(2) = 3{,}045 \end{cases}$$

Получите таблицу приближенных значений неизвестной функции $y(x)$ на отрезке интегрирования $x \in [a; b]$ с постоянным шагом h.

Аппроксимируйте полученные табличные значения кубическим сплайном и постройте приближенный график функции $y(x)$ на отрезке интегрирования, используя скрипты из главы 2. При построении графика шаг интерполяции примите равным $0{,}1 \cdot h$.

С помощью среды Matlab напишем программу для интерполяции, которая состоит из:

lab_8.m – основной скрипт, вызывающий функции для расчета точек методом Эйлера-Коши и выводящий их в виде графика (листинг 3.3);

EL_CSH.m – метод Эйлера-Коши (листинг 3.4);

а также функции для интерполяции кубическим сплайном, которые представлены выше.

```
h = 0.2;
x = 2:h:4;
y = EL_CSH(x);
disp(x);
disp(y);
M = progon (x, y);
x1 = min(x):0.01:max(x);
itr = interval(x, x1);
y1 = spline_val(x,y,x1,itr,M);
a = min([x x1]); b = max([x x1]);
figure(1);
plot(x,y, 'o' , x1,y1, 'r');
axis([a-0.5 b+0.5 min([y y1])-0.5 max([y y1])+0.5]);
xlabel('x'); ylabel('y');
grid on;
legend("Расчетные точки", "Интерполяция");
```

Листинг 3.3 – Основной скрипт метода Эйлера-Коши

```
function [y] = EL_CSH (x)
n = length(x);
y = zeros(1,n);
y(1) = 3.045;
for i = 1:n-1
    y1 = y(i)+ 0.2*((9*x(i)^2-1)*exp(-x(i)) - y(i));
    y(i+1) = y(i) + 0.1*((9*x(i)^2-1)*exp(-x(i)) - y(i) +
(9*x(i+1)^2-1)*exp(-x(i+1)) - y1);
end
end
```

Листинг 3.4 – Метод Эйлера-Коши

Численное решение ОДУ приведено в таблице 3.2.

Таблица 3.2 – Численное решение ОДУ методом Эйлера-Коши

k	1	2	3	4	5	6	7	8	9	10	11
x_k	2,0	2,2	2,4	2,6	2,8	3,0	3,2	3,4	3,6	3,8	4,0
y_k	3,045	3,347	3,583	3,752	3,855	3,898	3,886	3,828	3,73	3,6	3,445

График приближенного решения ОДУ, построенный с помощью интерполяции кубическим сплайном, представлен на рисунке 3.2.

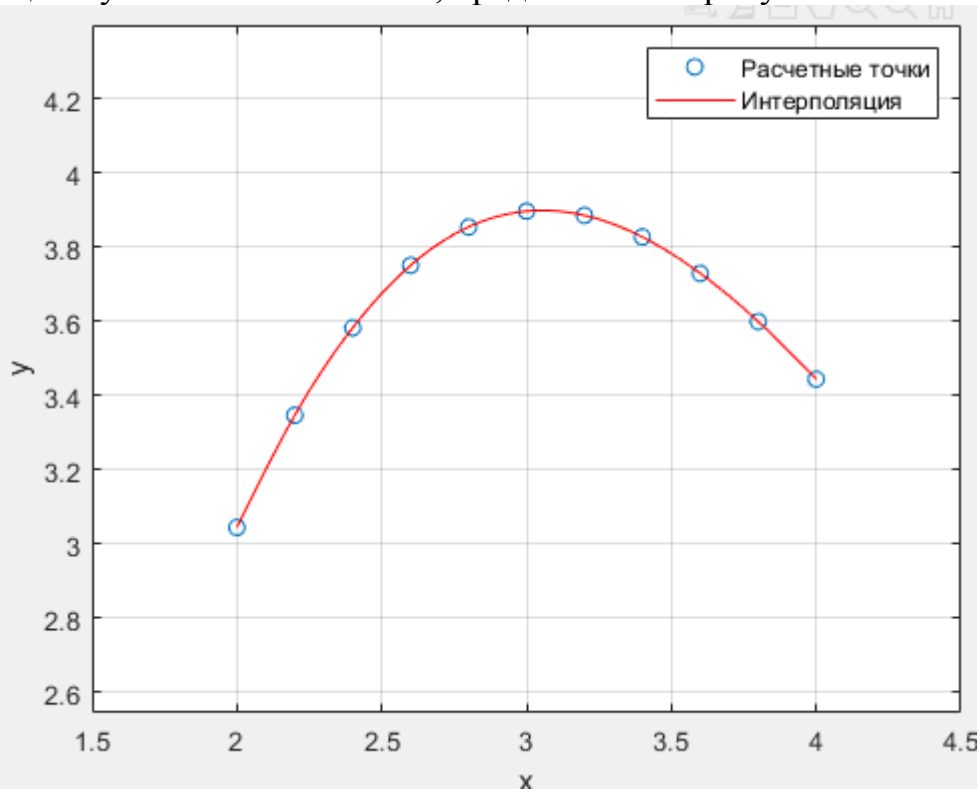


Рисунок 3.2 – График приближенного решения ОДУ методом Эйлера-Коши

3.5 Вывод

Были рассмотрены два метода решения ОДУ первого порядка: Эйлера и Эйлера – Коши. В методе Эйлера происходит движение не по интегральной кривой, а по касательной к ней. На каждом шаге касательная находится уже для новой интегральной кривой, таким образом погрешность решения будет возрастать от шага к шагу. То есть метод дает низкую точность. Метод Эйлера-Коши базируется на предыдущем, однако здесь используется среднее значение угла наклона касательных на левой и правой границе отрезка, что повышает точность.

Заключение

Задачи, на которые ответ нужно дать в виде числа, как известно, решаются с помощью математических методов. На сегодняшний день существует три основных группы таких методов: аналитические, графические и численные.

При использовании аналитических методов решение задачи удается выразить с помощью формул. Например, если задача состоит в решении простейших алгебраических, тригонометрических, дифференциальных и т.д. уравнений, то использование известных из курса математики приемов сразу приводит к цели.

Преимущество аналитических методов: в результате применения аналитических методов за небольшой отрезок сразу получается точный ответ.

Недостаток аналитических методов: аналитические методы применимы лишь к небольшому числу, как правило, не очень сложных по своей структуре задач. Так, например, до сих пор не удалось решить в общем виде уравнение пятой степени.

Основная идея графических методов состоит в том, что решение находится путем геометрических построений. Например, если уравнение $f(x)=0$ не удается решить аналитически, то строят график функции $y=f(x)$ и абсциссу точки пересечения его с осью *Ox* берут за приближенное значение корня.

Недостаток графических методов: в результате применения графических методов ответ получается с погрешностью, недопустимой в силу своей большой величины.

Основным инструментом для решения сложных математических моделей и задач в настоящее время являются численные методы. Они сводят решение задачи к выполнению конечного числа арифметических действий над числами и дают результат в виде числового значения с погрешностью, приемлемой для данной задачи.

Численные методы разработаны давно. Однако при вычислениях вручную они могли использоваться лишь для решения не слишком трудоемких задач. С появлением компьютеров, которые за короткое время могут выполнить миллиарды операций, начался период бурного развития численных методов и внедрения их в практику.

Из примеров применения можно выделить оптимизацию каких-либо процессов, действий или того, что можно описать уравнениями.

Библиографический список

1. Вержбицкий, В. М. Численные методы. Линейная алгебра и нелинейные уравнения / В. М. Вержбицкий. – М.: Оникс 21 век, 2005. – 636 с. – Текст : непосредственный.

2. Вержбицкий, В. М. Численные методы. Математический анализ и обыкновенные дифференциальные уравнения / В. М. Вержбицкий. – М.: Оникс 21 век, 2005. – 400 с. – Текст : непосредственный.

3. Бахвалов, Н. С. Численные методы / Н. С. Бахвалов, Н. П. Жидков, Г. М. Кобельков – М.: Бином. Лаборатория знаний, 2008. – 432 с. – Текст : непосредственный.

4. Ракитин, В. И. Практическое руководство по методам вычислений / В. И. Ракитин – М.: Высшая школа, 1998. – 374 с. – Текст : непосредственный.