**4 вариант**

**Задание 1.**

Найти приближенное значение первой и второй производной функции при заданном значении аргумента ξ = 1,4 с помощью соответствующего интерполяционного полинома Ньютона, если функция задана в равностоящих узлах

yi = f(xi); xi = x0 +i⋅h; h = const; i = 0…6;

y’ξ = f’(ξ); y’ξ - ? 6; y”ξ = f”(ξ); y”ξ - ? .

Оценить погрешность полученного значения.

Провести проверку вычислений в среде MS Excel и программе на C++. Сделать вывод о точности решения.

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| *i* | 0 | 1 | 2 | 3 | 4 | 5 | 6 |
| *x* | 1 | 1,15 | 1,3 | 1,45 | 1,6 | 1,75 | 1,9 |
| *y* | 0,3679 | 0,3064 | 0,2399 | 0,1771 | 0,1237 | 0,0819 | 0,0514 |

**Решение.** Используя конечные разности шестого порядка найдем приближенные значения первой и второй производных этой функции в точке ξ = *x* = 1,4, с помощью соответствующего интерполяционного полинома Ньютона. При подстановке табличного значения аргумента *xi* по-хорошему мы должны получить соответствующее табличное значение *y*i. Последнее утверждение является следствием того, что используем интерполяционные многочлены (которые должны проходить через заданные в таблице узловые точки). Именно это утверждение и является критерием проверки правильности вычислений.

По семи узлам (i = 0...6) табличной функции рассчитываем таблицу разностей.

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| *i* | *x*i | *y*i | Δ*y*0 | Δ2 *y*0 | Δ3 *y*0 | Δ4 *y*0 | Δ5 *y*0 | Δ6 *y*0 |
| 0 | 1 | 0,3679 | -0,0615 | -0,005 | 0,0087 | -0,003 | -0,0005 | 0,0015 |
| 1 | 1,15 | 0,3064 | -0,0665 | 0,0037 | 0,0057 | -0,0035 | 0,001 |  |
| 2 | 1,3 | 0,2399 | -0,0628 | 0,0094 | 0,0022 | -0,0025 |  |  |
| 3 | 1,45 | 0,1771 | -0,0534 | 0,0116 | -0,0003 |  |  |  |
| 4 | 1,6 | 0,1237 | -0,0418 | 0,0113 |  |  |  |  |
| 5 | 1,75 | 0,0819 | -0,0305 |  |  |  |  |  |
| 6 | 1,9 | 0,0514 |  |  |  |  |  |  |

Определяем с помощью полученной таблицы 1 интерполяционный полином Ньютона и соответствующие формулы для численного дифференцирования первой и второй производной.

)

)(

)(

(

)

)(

)(

(

!

6

...

)

(

)

(

5

4

3

2

1

0

6

0

6

0

0

0

*x*

*x*

*x*

*x*

*x*

*x*

*x*

*x*

*x*

*x*

*x*

*x*

*h*

*y*

*x*

*x*

*h*

*y*

*y*

*P*

*x*

*f*

*В*







































































*y*

*q*

*q*

*q*

*q*

*q*

*y*

*q*

*q*

*y*

*q*

*y*

*x*

*y*

5

0

2

0

0

!

5

)

4

)(

3

)(

2

)(

1

(

...

!

2

)

1

(

)

(

0

где и



Производя перемножение биномов, получим:



Так как



находим первую производную 













































0

4

1

2

3

0

3

1

2

0

2

0

24

6

22

18

4

6

2

6

3

.

2

1

2

1

)

(

*y*

*q*

*q*

*q*

*y*

*q*

*q*

*y*

*q*

*y*

*h*

*x*

*y*



Аналогично, находим вторую производную

*dq*

*y*

*d*

*h*

*dx*

*dq*

*dq*

*y*

*d*

*x*

*y*

)

(

1

)

(

)

(

2

















+…



Если и , тогда



,



.



Если  и , тогда



Приведенные формулы позволяют вычислить приближенное значение первой и второй производной функции при заданном значении аргумента ξ = 1,4 и для табличных значений  с помощью соответствующего интерполяционного полинома Ньютона, когда функция задана таблично в равностоящих узлах.

Приближенное значение первой и второй производной функции при заданном значении аргумента ξ = 1,4 вычислены в среде MS Excel:

|  |  |
| --- | --- |
| *y*'(ξ)= | -0,41165 |
| *y*''(ξ)= | 0,36460 |

Проведем проверку вычислений в среде MS Excel, для чего вычислим значения функции  во всех табличных значениях  и сверим эти значения с табличными:

|  |  |  |
| --- | --- | --- |
| *x* | *y* |  |
| 1 | 0,3679 | 0,36790 |
| 1,15 | 0,3064 | 0,30640 |
| 1,3 | 0,2399 | 0,23990 |
| 1,45 | 0,1771 | 0,17710 |
| 1,6 | 0,1237 | 0,12370 |
| 1,75 | 0,0819 | 0,08190 |
| 1,9 | 0,0514 | 0,05140 |

Оценим погрешность интерполирования полиномом . Вроде для табличных значений аргумента  погрешность равна нулю, поэтому речь идет об оценке  при значениях . Пусть функция , значения которой занесены в таблицу, имеет непрерывную  производную на отрезке .

Тогда вот эта погрешность, которую я никак не вычислю, определяется формулой:

, где , - точка из .

Так как точка  - неизвестна, то эта формула позволяет только оценить погрешность , где .

Найти приближенное значение первой и второй производной функции при заданном значении аргумента ξ = 1,4 с помощью соответствующего интерполяционного полинома Лагранжа, если функция задана в равностоящих узлах

yi = f(xi); xi = x0 +i⋅h; h = const; i = 0…6;

y’ξ = f’(ξ); y’ξ - ? 6; y”ξ = f”(ξ); y”ξ - ? .

Оценить погрешность полученного значения.

Для формирования вывода о точности решения воспользуемся MathCad т.к. я не совсем уверен в формулах MS Excel.

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| *i* | 0 | 1 | 2 | 3 | 4 | 5 | 6 |
| *x* | 1 | 1,15 | 1,3 | 1,45 | 1,6 | 1,75 | 1,9 |
| *y* | 0,3679 | 0,3064 | 0,2399 | 0,1771 | 0,1237 | 0,0819 | 0,0514 |

**Решение.**

Интерполяционный полином Лагранжа , для семи табличных точек – узлов интерполяции, ищется в виде:

, где

,

,

,

– – –

.

Каждый такой множитель  выдает значение  и выдает значение  при , где .

Подставив табличные значения узлов интерполяции, и проведя сведение коэффициентов уравнения, приведем его к виду:

.

MathCad – листинг с проведенными расчетами:















На графике видим, что полученная функция проходит через узловые точки из таблицы. Это означает, что функция интерполирования задана верно и я ошибался насчет формул Excel, все действительно неплохо.

Мы получили интерполяционный полином Лагранжа:

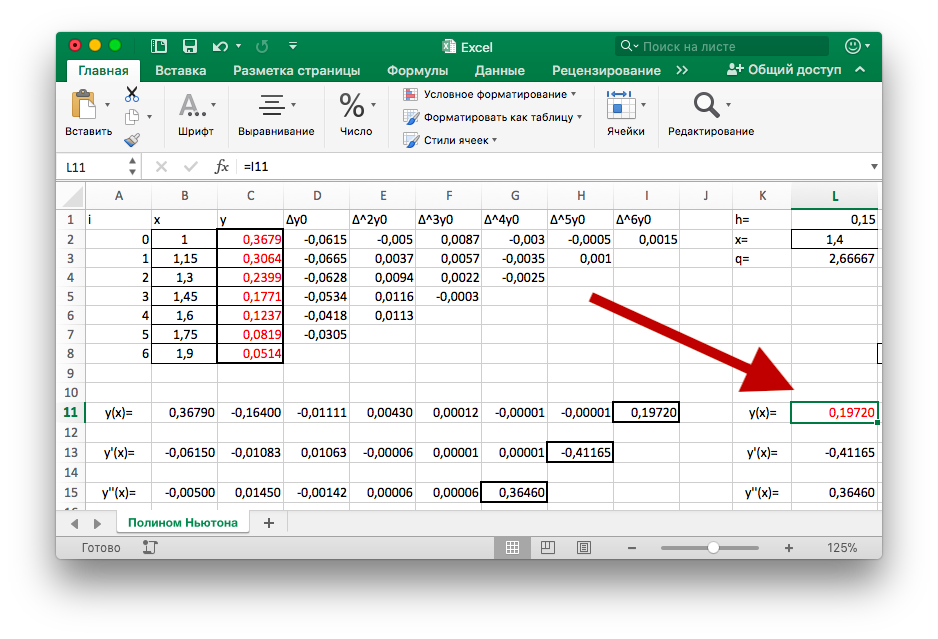
.

Продифференцировав его и подставив в полученные производные функции аргумент ξ = 1,4, найдем значения производной и второй производной в требуемой точке:

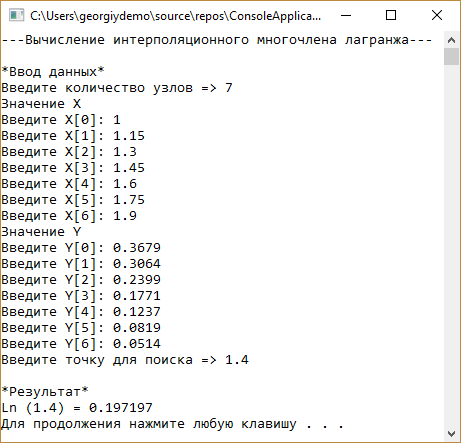


|  |  |
| --- | --- |
| *y*'(*x*)= | -0,4116 |
| *y*''(*x*)= | 0,3646 |

Результат Excel:



Результат работы программы:



Как видим, полученные значения производных интерполяционных многочленов в заданной точке совпадают по всем знакам точности кроме последнего, с которой заданы табличные значения функции:

Полином Ньютона

|  |  |
| --- | --- |
| *y*'(ξ)= | -0,41165 |
| *y*''(ξ)= | 0,36460 |

Полином Лагранжа

|  |  |
| --- | --- |
| *y*'(*x*)= | -0,4116 |
| *y*''(*x*)= | 0,3646 |

Также погрешность результатов работы программы на C++ и Excel относительно наших вычислений незначительна, что подтверждает верность вычислений:

* MathCad:
* Microsoft Excel:
* Программа на C++:

Листинг программы:

#include **"stdafx.h"**#include **<iostream>**;  
**using namespace** std;  
  
**double** \*X, \*Y;  
**int** n = 0;  
  
**double** decider(**double** \*X, **double** \* Y, **double** t) {  
 **double** result, n1, n2;  
 result = 0;  
 **for** (**int** j = 0; j<n; j++) {  
 n1 = 1; n2 = 1;  
 **for** (**int** i = 0; i<n; i++) {  
 **if** (i == j) {  
 n1 = n1 \* 1; n2 = n2 \* 1;  
 }  
 **else** {  
 n1 = n1 \* (t - X[i]);  
 n2 = n2 \* (X[j] - X[i]);  
 }  
 }  
 result = result + Y[j] \* n1 / n2;  
 }  
 **return** result;  
}  
  
**int** main() {  
 setlocale(LC\_ALL, **"RUS"**);  
 cout << **"---Вычисление интерполяционного многочлена лагранжа---\n\n"**;  
 cout << **"\*Ввод данных\*\n"**;  
 cout << **"Введите количество узлов => "**; cin >> n;  
 X = **new double**[n];  
 Y = **new double**[n];  
  
 cout << **"Значение X\n"**;  
 **for** (**int** i = 0; i<n; i++) {  
 cout << **"Введите X["** << i << **"]: "**;  
 cin >> X[i];  
 }  
  
 cout << **"Значение Y\n"**;  
 **for** (**int** i = 0; i < n; i++) {  
 cout << **"Введите Y["** << i << **"]: "**;  
 cin >> Y[i];  
 }  
  
 cout << **"Введите точку для поиска => "**;  
 **double** t = 0;  
 cin >> t;  
  
 cout << **"\n\*Результат\*\n"**;  
 cout << **"Ln ("** << t << **") = "** << decider(X, Y, t) << **"\n"**;  
 system(**"pause"**);  
}

**Задание 2.** Численное интегрирование

1. Найти приближенное значение интеграла а) по формулам прямоугольников, трапеции с точностью, =10-3.
2. Найти приближенное значение интеграла б) по формулам Симпсона с точностью, =10-3.
3. Сравнить полученные результаты.

Интегралы для вычисления (N=4):

1. 

б) 

1. Найдем приближенное значение интеграла  по формулам прямоугольников, трапеции с точностью, =10-3.

Для начала определим количество шагов разбиения и шаг разбиения интервала интегрирования.

, ; .

Согласно формулы погрешности приближенного вычисления определенного интеграла по методу прямоугольников , где  на отрезке .

Еще раз воспользуемся помощью MathCad:



; 

.

При решении данного неравенства получаем, что .

Таким образом, для достижения необходимой точности методом прямоугольников для данного интеграла достаточно положить количество точек разбиения , ну мы тогда и возьмем .

Определим тогда шаг разбиения:



Зададим значения абсциссы точек разбиения отрезка  на  равных частей 

 и определяем значения подынтегральной функции для каждого  :

.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | 0 | 1 | 2 | 3 | 4 |
|  | 0,400 | 0,6 | 0,8 | 1 | 1,2 |
|  | 0,36358 | 0,33619 | 0,30445 | 0,26969 | 0,23319 |

Все найденные значения подставляем в формулу левых прямоугольников:

=0,2(0,36358+0,33619+0,30445+0,26969)=

=0,24748.

Теперь все найденные значения подставляем в формулу правых прямоугольников:

=0,2(0,33619+0,30445+0,26969+0,23319)=0,2214.

Вычислим интеграл методом трапеций. Поскольку формула для вычисления погрешности отличается, то найдем какое  нам подходит:

; 

.

При решении данного неравенства получаем, что .

Таким образом, для достижения необходимой точности методом трапеций для данного интеграла достаточно положить количество точек разбиения . Тогда давайте возьмем .

Некоторые узловые значения будут те же, что и для метода прямоугольников:

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |
|  | 0,400 | 0,5 | 0,6 | 0,7 | 0,8 | 0,9 | 1 | 1,1 | 1,2 |
|  | 0,364 | 0,351 | 0,336 | 0,321 | 0,304 | 0,287 | 0,27 | 0,252 | 0,233 |

Все найденные значения подставляем в формулу трапеций

0,241890,242.

Вычислим интеграл б)  методом Симпсона (парабол):

Для начала определим количество шагов разбиения и шаг разбиения интервала интегрирования.

, ; .

Согласно формулы погрешности приближенного вычисления определенного интеграла по методу Симпсона , где  на отрезке .

В последний раз обратимся к MathCad:



; 

.

При решении данного неравенства получаем, что .

Таким образом, для достижения необходимой точности методом Симпсона для данного интеграла достаточно положить любое количество точек разбиения. Возьмем .

Определим теперь шаг разбиения:



Зададим значения абсциссы точек разбиения отрезка  на  равных частей ,

 и определяем значения подынтегральной функции для каждого  :

.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | 0 | 1 | 2 | 3 | 4 |
|  | 1,2 | 1,425 | 1,65 | 1,875 | 2,1 |
|  | 0,42875 | 0,40721 | 0,38569 | 0,36477 | 0,34483 |

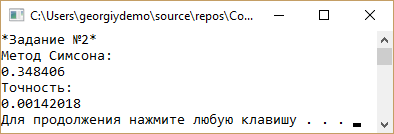
Все найденные значения подставляем в формулу Симпсона:

Как следствие, получаем ответ:

(0,42875+4\*(0,40721+0,36477)+2\*0,38569+0,34483)=

=0,34747.

Проверим вычисления на программе C++:



Значения отличаются незначительно, следовательно, вычисления верны.

Листинг программы:

#include **"stdafx.h"**#include **<iostream>**#include **<cmath>  
  
using namespace** std;  
  
**const int** N = 4;  
**double** SimpsonFunction(**double** x);  
**double** MainSimpson(**double** a, **double** b, **int** n);  
  
**int** main()  
{  
 setlocale(LC\_ALL, **"RUS"**);  
 **double** y;  
 **double** raz1, y1;  
 cout << **"\*Задание №2\*\nМетод Симсона:\n"**;  
 y = MainSimpson(1.2, 2.1, 220);  
 cout << y << **"\n"**;  
 y1 = MainSimpson(1.2, 2.1, 219);  
 raz1 = abs(y - y1);  
 cout << **"Точность:\n"** << raz1 << **"\n"**;  
 system(**"pause"**);  
 **return** 0;  
}  
  
**double** SimpsonFunction(**double** x)  
{  
 **return** (1 / (sqrt(x \* x + N)));  
}  
  
**double** MainSimpson(**double** a, **double** b, **int** n)  
{  
 **double** h, s, x, f;  
 **int** i = 1;  
 h = (b - a) / n;  
 x = a;  
 f = SimpsonFunction(x);  
 s = f;  
 **while** (i <= n) {  
 x = x + h;  
 f = SimpsonFunction(x);  
 s = s + 4 \* f;  
 i = i + 2;  
 x = x + h;  
 f = SimpsonFunction(x);  
 s = s + 2 \* f;  
 }  
 x = b;  
 f = SimpsonFunction(x);  
 s = (s + f) \* (h / 3);  
 **return** s;  
}