Министерство образования и науки РФ

ФГБОУ ВПО «Тамбовский государственный технический университет»

Кафедра

Отчёт по лабораторной работе №5

по дисциплине «Вычислительная математика»

«Интерполяция»

Вариант 7

Выполнил: студент гр.

.

Проверил:.

Тамбов 20

**Постановка задачи:**

Составить блок-схему алгоритма и реализовать его в программе для вычислений на ЭВМ для следующих методов интерполяции:

1. Ньютона (1 и 2 формулы);
2. Гаусса (1 и 2 формулы);
3. Стирлинга;
4. Бесселя;
5. Лагранжа;
6. Кубических сплайнов.

При *х0*=1; *х1*=2; *х2*=3; *х3*=4; *х4*=5; y0=6; y1=3; y2=4; y3=10; y4=4;

1. ***Первая интерполяционная формула Ньютона:***

Таблица разностей

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| x | y | Δy | Δ2y | Δ3y | Δ4y |
| 1 | 6 | -3 | 4 | 1 | -6 |
| 2 | 3 | 1 | 5 | -5 |  |
| 3 | 4 | 6 | 0 |  |  |
| 4 | 10 | -6 |  |  |  |
| 5 | 4 |  |  |  |  |

Интерполирующий полином ищется в виде:

Примем n=4, x0=1, тогда интерполяционная формула Ньютона принимает вид:

Задача построения многочлена сводится к определению коэффициентов а из условий:

Полагаем *x* = *x*0, тогда, т.к. второе, третье и другие слагаемые равны 0,

Общая формула нахождения коэффициента *а*К, имеет вид:

При раскрытии скобок получим полином:

a0=6.000000; a1=-3.000000; a2=2.00000; a3=-1.666666; a4=-0.250000;

При подстановке значений получим:

Блок-схема:

a

начало

x0=1; h=1; y[-2]=6; y[-1]=3; y[0]=4; y[1]=10; y[2]=4

a[0]= Δy[0]

Δy[0]=y[0]

3 i=1

1 j=1

A[i]= Δy[i]/i!

2 i=0

3

i++

Пока i<5

y[i]=y[i+1]-y[i]

k0=a[0]-a[1]+2a[2]-6a[3]+24a[4]

2

i++

Пока i<4

K1=a[1]-3a[2]+11a[3]-50a[4]

Δy[j]=y[0]

K2=a[2]-6a[3]+35a[4]

1

j++

Пока i<5

K3=a[3]-10a[4]

a

K4=a[4]

b

Ввод *x,n*

конец

Вывод P(x)

b

P(x)=k4x4+k3x3+k2x2+k1x+k0

4 i=0

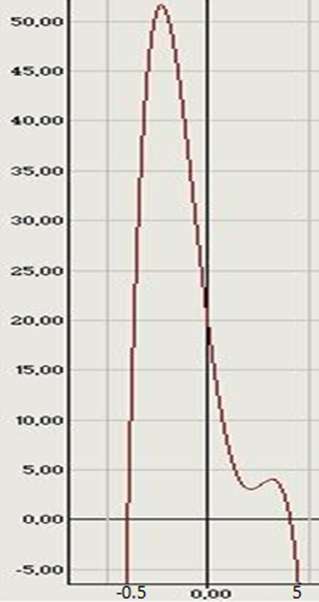
5

i++

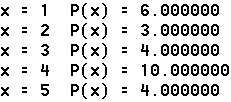
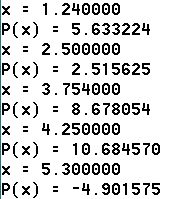
Пока i<n

График функции:

y=6-3\*(x-1)+2\*(x-1)\*(x-2)-1.666666\*(x-1)\*(x-2)\*(x-3)-0.25\*(x-1)\*(x-2)\*(x-3)\*(x-4)



Результаты программы:

Листинг программы:

#include <stdio.h>

#include <conio.h>

#include <math.h>

int fact(int n)

{

return n?n\*fact(n-1):1;

}

int main()

{

double y[5], x=1, a[5], dy[5], k4,k3,k2,k1,k0, P;

y[0]=6; y[1]=3; y[2]=4; y[3]=10; y[4]=4;

dy[0]=y[0];

for(int j=1; j<5; j++)

{

for (int i=0; i<4; i++)

{

y[i]=y[i+1]-y[i];

}

dy[j]=y[0];

}

a[0]=dy[0];

for(int i=1; i<5; i++ )

{

a[i]=dy[i]/fact(i);

}

k4=a[4];

k3=a[3]-10\*a[4];

k2=a[2]-6\*a[3]+35\*a[4];

k1=a[1]-3\*a[2]+11\*a[3]-50\*a[4];

k0=a[0]-a[1]+2\*a[2]-6\*a[3]+24\*a[4];

printf("k4 = %lf k3 = %lf k2 = %lf k1 = %lf k0 = %lf\n",k4,k3,k2,k1,k0);

for(int x=1; x<6; x++)

{

P=k4\*pow(x,4)+k3\*pow(x,3)+k2\*pow(x,2)+k1\*x+k0;

printf("x = %d P(x) = %lf\n",x, P);

}

getch();

return 0;

}

1. ***Вторая интерполяционная формула Ньютона***

Для нахождения значений функций в точках, расположенных в конце интервала интерполирования, используют второй интерполяционный полином Ньютона.

Таблица разностей

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| x | y | Δy | Δ2y | Δ3y | Δ4y |
| 1 | 6 | -3 | 4 | 1 | -6 |
| 2 | 3 | 1 | 5 | -5 |  |
| 3 | 4 | 6 | 0 |  |  |
| 4 | 10 | -6 |  |  |  |
| 5 | 4 |  |  |  |  |

Примем n=4, xn=5, тогда интерполяционная формула Ньютона принимает вид:

т.е. формула нахождений будет следующей: .

При раскрытии скобок получим полином:

a0=4.000000; a1=-6.000000; a2=0.000000; a3=-0.833333; a4=-0.250000;

При подстановке значений получим:

Блок-схема:

начало

a

x0=1; h=1; y[-2]=6; y[-1]=3; y[0]=4; y[1]=10; y[2]=4

a[0]= Δy[0]

Δy[0]=y[4]

3 i=1

1 j=1; l=3

a[i]= Δy[i]/i!

2 i=0

3

i++

Пока i<5

y[i]=y[i+1]-y[i]

k0=a[0]-5a[1]+20a[2]-60a[3]+120a[4]

2

i++

Пока i<4

k1=a[1]-9a[2]+47a[3]-154a[4]

Δy[j]=y[l]

k2=a[2]-12a[3]+71a[4]

1

j++, l--

Пока i<5

k3=a[3]-14a[4]

k4=a[4]

a

b

Ввод *x, n*

конец

Вывод P(x)

b

P(x)=k4x4+k3x3+k2x2+k1x+k0

4 i=0

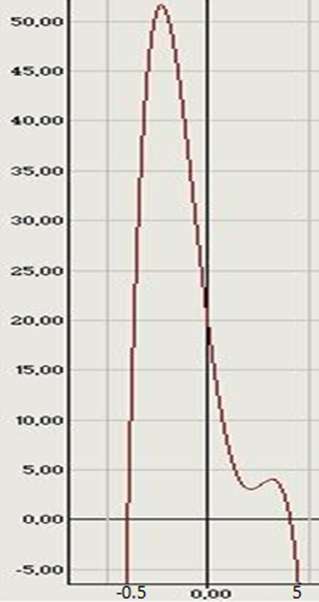
5

i++

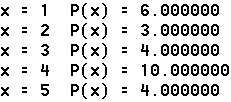
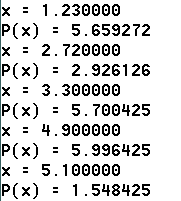
Пока i<n

График функции:

y= 4-6\*(x-5)+0\*(x-5)\*(x-4)-0.833333\*(x-5)\*(x-4)\*(x-3)-0.25\*(x-5)\*(x-4)\*(x-3)\*(x-2)



Результаты программы:

Листинг программы:

#include <stdio.h>

#include <conio.h>

#include <math.h>

int fact(int n)

{

return n?n\*fact(n-1):1;

}

int main()

{

double y[5], x=1, a[5], dy[5], k4,k3,k2,k1,k0, P;

y[0]=6; y[1]=3; y[2]=4; y[3]=10; y[4]=4;

dy[0]=y[4];

for(int j=1, l=3; j<5; j++, l--)

{

for(int i=0; i<5; i++)

y[i]=y[i+1]-y[i];

dy[j]=y[l];

}

a[0]=dy[0];

for(int i=1; i<5; i++)

a[i]=dy[i]/fact(i);

k4=a[4];

k3=-14\*a[4]+a[3];

k2=71\*a[4]-12\*a[3]+a[2];

k1=-154\*a[4]+47\*a[3]-9\*a[2]+a[1];

k0=120\*a[4]-60\*a[3]+20\*a[2]-5\*a[1]+a[0];

printf("k4 = %lf k3 = %lf k2 = %lf k1 = %lf k0 = %lf\n",k4,k3,k2,k1,k0);

for(int x=1; x<6; x++)

{

P=k4\*pow(x,4)+k3\*pow(x,3)+k2\*pow(x,2)+k1\*x+k0;

printf("x = %d P(x) = %lf\n",x, P);

}

getch();

return 0;

}

1. ***Первая интерполяционная формула Гаусса:***

Таблица разностей

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| x | y | Δy | Δ2y | Δ3y | Δ4y |
| 1 | 6 |  |  |  |  |
|  |  | -3 |  |  |  |
| 2 | 3 |  | 4 |  |  |
|  |  | 1 |  | 1 |  |
| 3 | 4 |  | 5 |  | -6 |
|  |  | 6 |  | -5 |  |
| 4 | 10 |  | 0 |  |  |
|  |  | 6 |  |  |  |
| 5 | 4 |  |  |  |  |

Значения разностей те же, что и в методе интерполяции Ньютона.

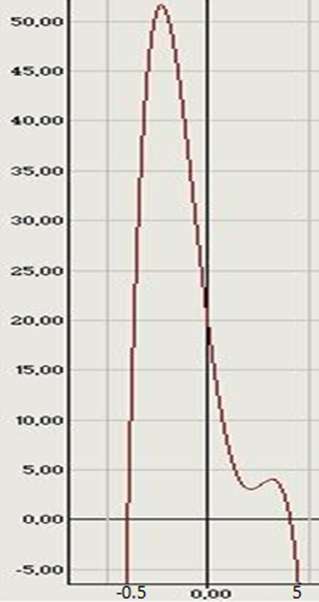
Примем n=4, x0=1, тогда интерполяционная формула Гаусса принимает вид:

При раскрытии скобок получим полином:

Найдем коэффициенты:

График функции:

y= 4+6\*(x-1)+2.5\*(x-1)\*(x-2)-0.833333\*x\*(x-1)\*(x-2)-0.25\*x\*(x-1)\*(x-2)\*(x-3)



Блок-схема:

начало

a

x4=5; h=1; y[-2]=6; y[-1]=3; y[0]=4; y[1]=10; y[2]=4; l=1

1

j++, l--

Пока i<5

Δy[0]=y[0]

3

i++

Пока i<5

3 i=1

a[0]= Δy[0]

A[i]= Δy[i]/i!

k0=a[0]-a[1]

k1=a[1]-a[2]+2a[3]-6a[4]

k2=a[2]-3a[3]+11a[4]

k3=a[3]-6a[4]

k4=a[4]

1 j=1, z=0

2 i=-2

y[i]=y[i+1]-y[i]

2

i++

Пока i<3

Δy[j]=y[l]

нет

l=2

да

z--; l=0

b

a

Ввод *x, n*

конец

Вывод P(x)

b

P(x)=k4x4+k3x3+k2x2+k1x+k0

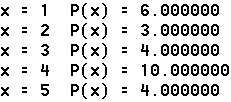
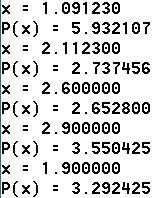
4 i=0

5

i++

Пока i<n

Результаты программы:

Листинг программы:

#include <stdio.h>

#include <conio.h>

#include <math.h>

int fact(int n)

{

return n?n\*fact(n-1):1;

}

int main()

{

double y[5], x=5, a[5], dy[5], k4,k3,k2,k1,k0, P,l=1;

y[-2]=6; y[-1]=3; y[0]=4; y[1]=10; y[2]=4;

dy[0]=y[0];

for(int j=1,z=0; j<5; j++)

{

for(int i=-2; i<3; i++)

y[i]=y[i+1]-y[i];

dy[j]=y[z]; l++;

if(l==2)

{

z--; l=0;

}

}

a[0]=dy[0];

for(int i=1; i<5; i++)

a[i]=dy[i]/fact(i);

k4=a[4];

k3=-6\*a[4]+a[3];

k2=11\*a[4]-3\*a[3]+a[2];

k1=-6\*a[4]+2\*a[3]-3\*a[2]+a[1];

k0=2\*a[2]-a[1]+a[0];

printf("k4 = %lf k3 = %lf k2 = %lf k1 = %lf k0 = %lf\n",k4,k3,k2,k1,k0);

for(int x=1; x<6; x++)

{

P=k4\*pow(x,4)+k3\*pow(x,3)+k2\*pow(x,2)+k1\*x+k0;

printf("x = %d P(x) = %lf\n",x, P);

}

getch();

return 0;

}

1. ***Вторая интерполяционная формула Гаусса***

Таблица разностей

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| x | y | Δy | Δ2y | Δ3y | Δ4y |
| 1 | 6 |  |  |  |  |
|  |  | -3 |  |  |  |
| 2 | 3 |  | 4 |  |  |
|  |  | 1 |  | 1 |  |
| 3 | 4 |  | 5 |  | -6 |
|  |  | 6 |  | -5 |  |
| 4 | 10 |  | 0 |  |  |
|  |  | 6 |  |  |  |
| 5 | 4 |  |  |  |  |

Примем n=4, x0=1, тогда интерполяционная формула Гаусса принимает вид:

При раскрытии скобок получим полином:

При подстановке значений получим:

Блок-схема:

начало

a

x4=5; h=1; y[-2]=6; y[-1]=3; y[0]=4; y[1]=10; y[2]=4; l=0

1

j++

Пока i<5

Δy[0]=y[0]

3

i++

Пока i<5

3 i=1

a[0]= Δy[0]

A[i]= Δy[i]/i!

k0=a[0]-a[1]

k1=a[1]-a[2]+2a[3]+2a[4]

k2=a[2]+2a[3]-a[4]

k3=a[3]-2a[4]

k4=a[4]

1 j=1, z=-1

2 i=-2

y[i]=y[i+1]-y[i]

2

i++

Пока i<3

Δy[j]=y[z]; l++

нет

l=2

да

z--; l=0

b

a

Ввод *x, n*

конец

Вывод P(x)

b

P(x)=k4x4+k3x3+k2x2+k1x+k0

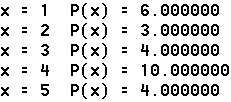
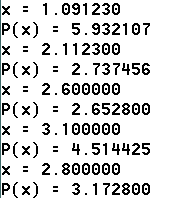
4 i=0

5

i++

Пока i<n

Результаты программы:

Листинг программы:

#include <stdio.h>

#include <conio.h>

#include <math.h>

int fact(int n)

{

return n?n\*fact(n-1):1;

}

int main()

{

double y[5], x=5, a[5], dy[5], k4,k3,k2,k1,k0, P,l=0;

y[-2]=6; y[-1]=3; y[0]=4; y[1]=10; y[2]=4;

dy[0]=y[0];

for(int j=1,z=-1; j<5; j++)

{

for(int i=-2; i<3; i++)

y[i]=y[i+1]-y[i];

dy[j]=y[z]; l++;

if(l==2)

{

z--; l=0;

}

}

a[0]=dy[0];

for(int i=1; i<5; i++)

a[i]=dy[i]/fact(i);

k4=a[4];

k3=-2\*a[4]+a[3];

k2=-a[4]-3\*a[3]+a[2];

k1=2\*a[4]+2\*a[3]-a[2]+a[1];

k0=-a[1]+a[0];

for(int x=1; x<6; x++)

{

P=k4\*pow(x,4)+k3\*pow(x,3)+k2\*pow(x,2)+k1\*x+k0;

printf("x = %d P(x) = %lf\n",x, P);

}

getch();

return 0;

}

1. ***Интерполяционная формула Стирлинга***

Взяв среднее арифметическое первой и второй интерполяционных формул Гаусса, получим *формулу Стирлинга*:

Примем n=4, x0=1, тогда интерполяционная формула Стирлинга принимает вид:

При раскрытии скобок получим полином:

При подстановке значений получим:

Блок-схема:

начало

a

1

j++

Пока i<5

x4=5; h=1; y[-2]=6; y[-1]=3; y[0]=4; y[1]=10; y[2]=4; l=1

Δy[0]=y[0]

3 i=1

a[0]= Δy[0]

A[i]= Δy[i]/i!

k0=a[0]-a[1]+2a[2]

k1=a[1]-a[2]+2a[3]-2a[4]

k2=a[2]-3a[3]+5a[4]

k3=a[3]-4a[4]

k4=a[4]

1 j=1, z=0

2 i=-2

y[i]=y[i+1]-y[i]

3

i++

Пока i<5

2

i++

Пока i<3

нет

да

l=1

Δy[j]=(y[z-1]+y[z])/2; l=0; z--;

Δy[j]=y[z]; l++;

a

b

Ввод *x, n*

конец

Вывод P(x)

b

P(x)=k4x4+k3x3+k2x2+k1x+k0

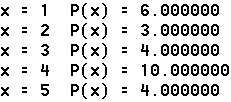
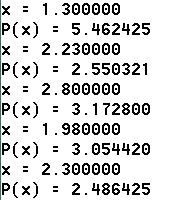
4 i=0

5

i++

Пока i<n

Результаты программы:

Листинг программы:

#include <math.h>

#include <stdio.h>

#include <conio.h>

int fact(int n)

{

return n?n\*fact(n-1):1;

}

int main()

{

double y[5], x=5, a[5], dy[5], k4,k3,k2,k1,k0, P,l=1;

y[-2]=6; y[-1]=3; y[0]=4; y[1]=10; y[2]=4;

dy[0]=y[0];

for(int j=1,z=0; j<5; j++)

{

for(int i=-2; i<3; i++)

y[i]=y[i+1]-y[i];

if(l==1)

{

dy[j]=(y[z-1]+y[z])/2;

z--; l=0;

}

else

{

dy[j]=y[z]; l++;

}

}

a[0]=dy[0];

for(int i=1; i<5; i++)

a[i]=dy[i]/fact(i);

k4=a[4];

k3=-4\*a[4]+a[3];

k2=5\*a[4]-3\*a[3]+a[2];

k1=-2\*a[4]+2\*a[3]-2\*a[2]+a[1];

k0=a[2]-a[1]+a[0];

printf("k4 = %lf k3 = %lf k2 = %lf k1 = %lf k0 = %lf\n",k4,k3,k2,k1,k0);

for(int x=1; x<6; x++)

{

P=k4\*pow(x,4)+k3\*pow(x,3)+k2\*pow(x,2)+k1\*x+k0;

printf("x = %d P(x) = %lf\n",x, P);

}

getch();

return 0;

}

1. ***Интерполяционная формула Бесселя***

Примем n=4, x0=1, тогда интерполяционная формула Бесселя принимает вид:

При раскрытии скобок получим полином:

При подстановке значений получим:

Блок-схема:

b

k3=a[3]

k2=a[2]-4.5a[3]

k1=a[1]-3a[2]+6.5a[3]

k0=a[0]-1.5a[1]+2a[2]-3a[3]

A[i]= Δy[i]/i!

a[0]= Δy[0]/2

3 i=1

начало

3

i++

Пока i<5

a

1

j++

Пока i<5

x4=5; h=1; y[-2]=6; y[-1]=3; y[0]=4; y[1]=10; y[2]=4; l=0

Δy[0]=y[0]+y[1]

1 j=1, z=0

2 i=-2

y[i]=y[i+1]-y[i]

2

i++

Пока i<3

нет

да

l=1

Δy[j]=(y[z-1]+y[z])/2; l=0; z--;

Δy[j]=y[z]; l++;

a

Ввод *x, n*

конец

Вывод P(x)

b

P(x)=k4x4+k3x3+k2x2+k1x+k0

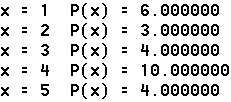
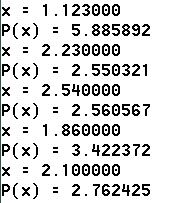
4 i=0

5

i++

Пока i<n

Результаты программы:

Листинг программы:

#include <math.h>

#include <conio.h>

#include <stdio.h>

int fact(int n)

{

return n?n\*fact(n-1):1;

}

int main()

{

double y[5], x=5, a[5], dy[5], k3,k2,k1,k0, P,l=0;

y[-2]=6; y[-1]=3; y[0]=4; y[1]=10; y[2]=4;

dy[0]=y[0]+y[1];

for(int j=1,z=0; j<5; j++)

{

for(int i=-2; i<3; i++)

y[i]=y[i+1]-y[i];

if(l==1)

{

dy[j]=(y[z-1]+y[z])/2;

z--; l=0;

}

else

{

dy[j]=y[z]; l++;

}

}

a[0]=dy[0]/2;

for(int i=1; i<4; i++)

a[i]=dy[i]/fact(i);

k3=a[3];

k2=-4.5\*a[3]+a[2];

k1=6.5\*a[3]-3\*a[2]+a[1];

k0=-3\*a[3]+2\*a[2]-1.5\*a[1]+a[0];

for(int x=1; x<6; x++)

{

P=k3\*pow(x,3)+k2\*pow(x,2)+k1\*x+k0;

printf("x = %d P(x) = %lf\n",x, P);

}

getch();

return 0;

}

1. ***Интерполяционная формулы Лагранжа***

Примем n=4, x0=1, x1=2, x2=3, x3=4, x4=5 тогда интерполяционная формула Лагранжа принимает вид:

При раскрытии скобочек получим полином:

При подстановке значений получим:

Блок-схема:

конец

1

i++

Пока i<n

Вывод P(x)

P(x)=k4x4+k3x3+k2x2+k1x+k0

Ввод *x, n*

1 i=0

начало

y[0]=6; y[1]=3; y[2]=4; y[3]=10; y[4]=4;

k0=120\*y[4]/24+(60)\*y[3]/(-6)+40\*y[2]/4+(30)\*y[1]/(-6)+24\*y[0]/24;

k1=(-154)\*y[4]/24+(-107)\*y[3]/(-6)+(-78)\*y[2]/4+(-61)\*y[1]/(-6)+(-50)\*y[0]/24;

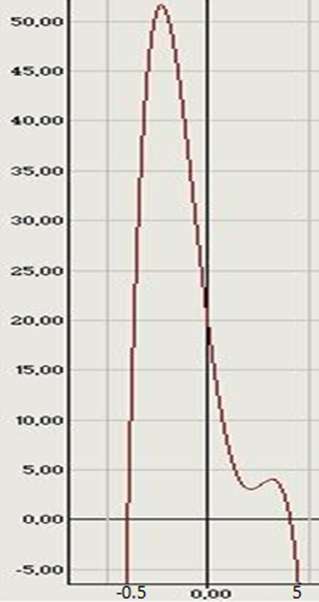
k2=71\*y[4]/24+59\*y[3]/(-6)+49\*y[2]/4+41\*y[1]/(-6)+35\*y[0]/24;

k4=y[4]/24+y[3]/(-6)+y[2]/4+y[1]/(-6)+y[0]/24;

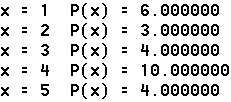
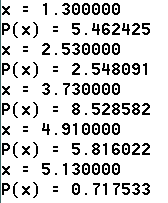
k3=(-14)\*y[4]/24+(-13)\*y[3]/(-6)+(-12)\*y[2]/4+(-11)\*y[1]/(-6)+(-10)\*y[0]/24;

График функции:

y=6\*(x-2)\*(x-3)\*(x-4)\*(x-5)/23+3\*(x-1)\*(x-3)\*(x-4)\*(x-5)/(-6)+4\*(x-1)\*(x-2)\*(x-4)\*(x-5)/4+10\*(x-1)\*(x-2)\*(x-3)\*(x-5)/(-6)+4\*(x-1)\*(x-2)\*(x-3)\*(x-4)/24



Результаты программы:

Листинг программы:

#include <iostream>

#include <stdio.h>

using namespace std;

int main()

{

double x[5], y[5], k0,k1,k2,k3,k4, P;

y[0]=6; y[1]=3; y[2]=4; y[3]=10; y[4]=4;

k4=y[4]/24+y[3]/(-6)+y[2]/4+y[1]/(-6)+y[0]/24;

k3=(-14)\*y[4]/24+(-13)\*y[3]/(-6)+(-12)\*y[2]/4+(-11)\*y[1]/(-6)+(-10)\*y[0]/24;

k2=71\*y[4]/24+59\*y[3]/(-6)+49\*y[2]/4+41\*y[1]/(-6)+35\*y[0]/24;

k1=(-154)\*y[4]/24+(-107)\*y[3]/(-6)+(-78)\*y[2]/4+(-61)\*y[1]/(-6)+(-50)\*y[0]/24;

k0=120\*y[4]/24+(60)\*y[3]/(-6)+40\*y[2]/4+(30)\*y[1]/(-6)+24\*y[0]/24;

for(int x=1; x<6; x++)

{

P=k4\*pow(x,4)+k3\*pow(x,3)+k2\*pow(x,2)+k1\*x+k0;

printf("x = %d P(x) = %lf\n",x, P);

}

return 0;

1. ***Интерполяционная формула кубический сплайнов.***

На каждом из отрезков [xi-1,xi], i=1,2,…n, будем искать сплайн-функцию s(x)=si(x) в виде полинома третьей степени:

Si(x)=ai+bi(x-xi)+ci/2\*(x-xi)2+d/6\*(x-xi)3, где i=1,2,..n, а xi-1<x<xi, ai, bi, ci, di- искомые коэффициенты.

ai=fi(xi)

Составляем систему уравнений для сi:

hi-1ci-1+2(hi+hi+1)ci+hi+1ci+1=6((yi+1-yi)/hi+1-(yi-yi-1)/hi)

Найдя значение сi, можно посчитать di и bi :

di=(ci-ci-1)/hi; bi=hici/2-hidi/6+(yi-yi-1)/hi

Таким образом, посчитав значение коэффициентов, будет найден единственный кубический сплайн.

При подстановке значений получил полином:

Блок-схема:

b

x[2]<=x<x[3]

J=3;

нет

да

x[1]<=x<x[2]

J=2;

нет

да

x[0]<x<x[1]

J=1;

нет

да

a

a

Ввод *x*

2 i=0

1

i++

Пока i<5

d[i]=c[i]-c[i-1];

b[i]=c[i]/2-d[i]/6+y[i]-y[i-1];

1 i=1

c[1]=(6\*(y[2]-2\*y[1]+y[0])-c[2])/4;

c[3]=(6\*(y[4]-2\*y[3]+y[2])-c[2])/4;

c[2]= (24\*(y[3]-2\*y[2]+y[1])+6\*(y[4]-2\*y[3]+y[2])+6\*(y[2]-2\*y[1]+y[0]))/14;

x[0]=1; x[1]=2; x[2]=3; x[3]=4; x[4]=5; y[0]=6; y[1]=3; y[2]=4; y[3]=10; y[4]=4; c[4]=0; c[0]=0;

начало

конец

1

i++

Пока i<5

Вывод s(x)

s[j]=y[j]+b[j]\*(q-x[j])+c[j]\*(q-x[j])\*(q-x[j])/2+d[j]\*(q-x[j])\*(q-x[j])\*(q-x[j])/6;

x[3]<=x<=x[4]

J=4;

нет

да

b

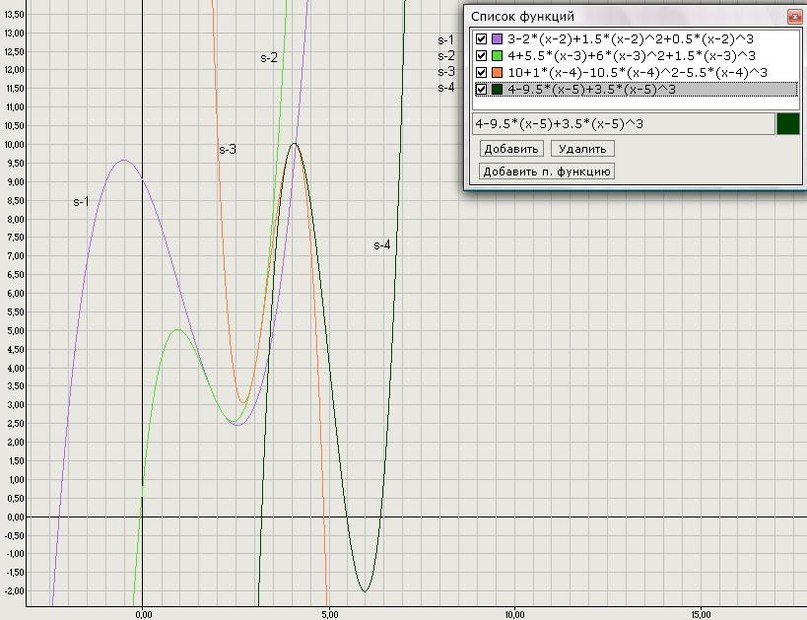
Графики функций:

3-2\*(x-2)+1.5\*(x-2)^2+0.5\*(x-2)^3;

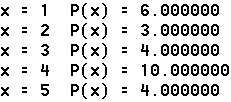
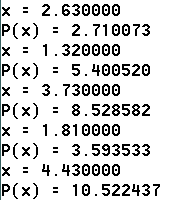
4+5.5\*(x-3)+6\*(x-3)^2+1.5\*(x-3)^3;

10+1\*(x-4)+10.5\*(x-4)^2-5.5\*(x-4)^3;

4-9.5\*(x-5)-3.5\*(x-5)^3;



Результаты программы:

Листинг программы:

#include <stdio.h>

int main()

{

double y[5], c[5], d[5], b[5],s[5], x[5], q;

int j=0;

for(int i=0; i<5; i++)

x[i]=i+1;

c[4]=0;c[0]=0;

y[0]=6; y[1]=3; y[2]=4; y[3]=10; y[4]=4;

c[2]=(24\*(y[3]-2\*y[2]+y[1])+6\*(y[4]-2\*y[3]+y[2])+6\*(y[2]-2\*y[1]+y[0]))/14;

c[1]=(6\*(y[2]-2\*y[1]+y[0])-c[2])/4;

c[3]=(6\*(y[4]-2\*y[3]+y[2])-c[2])/4;

for(int i=1; i<5; i++)

{

d[i]=c[i]-c[i-1];

b[i]=c[i]/2-d[i]/6+y[i]-y[i-1];

}

for(int i=1; i<5; i++)

{

printf("\nx = ");

scanf("%lf", &q);

if(x[0]<q && q<x[1])

j=1;

if(x[1]<=q && q<x[2])

j=2;

if(x[2]<=q && q<x[3])

j=3;

if(x[3]<=q && q<=x[4])

j=4;

s[j]=y[j]+b[j]\*(q-x[j])+c[j]\*(q-x[j])\*(q-x[j])/2+d[j]\*(q-x[j])\*(q-x[j])\*(q-x[j])/6;

printf("lf ", s[j]);

}

return 0;

}