|  |  |
| --- | --- |
| Gerb-BMSTU_01 | **Министерство науки и высшего образования Российской Федерации**  **Федеральное государственное бюджетное образовательное учреждение**  **высшего образования**  **«Московский государственный технический университет**  **имени Н.Э. Баумана**  **(национальный исследовательский университет)»**  **(МГТУ им. Н.Э. Баумана)** |

|  |  |
| --- | --- |
| ФАКУЛЬТЕТ | ИУ «Информатика и системы управления» |

|  |  |
| --- | --- |
| КАФЕДРА | ИУ-1 «Системы автоматического управления» |

**ОТЧЕТ ПО УЧЕБНОМУ ПРАКТИКУМУ НА ЭВМ**

|  |  |  |
| --- | --- | --- |
| Студент | Кочнов Андрей Александрович | |
|  | *фамилия, имя, отчество* | |
| Группа | ИУ1-32 |  |

|  |  |
| --- | --- |
| Тип практики | учебная |

|  |  |
| --- | --- |
| Название предприятия | Кафедра «Системы автоматического управления» |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Студент | |  |  |  | Кочнов А.А. |
|  | |  | *подпись, дата* |  | *фамилия, и.о.* |
| Руководитель практики | |  |  |  | Масленников А.Л. |
|  | |  | *подпись, дата* |  | *фамилия, и.о.* |
| Оценка |  | |  | | |

*2019 г.*

**Задание:**

Изучить алгоритмы билинейной и бикубической интерполяции. Реализовать в системе Matlab. Сравнить и сделать выводы.

**Оглавление**

1. Введение
2. Основная часть
   1. Общие положения.
   2. Билинейная интерполяция
   3. Бикубическая интерполяция
   4. Сравнение методов
3. Выводы
4. Приложения
   1. Приложение 1. Исходный код методов

**Введение**

Цель практикума: изучить методы билинейной и бикубической интерполяции. Сравнить их на некоторых данных

Работа выполняется в программной среде Matlab.

**Основная часть.**

**Общие принципы**

Интерполя́ция, интерполи́рование (от лат. inter–polis — «разглаженный, подновлённый, обновлённый; преобразованный») — в вычислительной математике способ нахождения промежуточных значений величины по имеющемуся дискретному набору известных значений. Термин «интерполяция» впервые употребил Джон Валлис в своём трактате «Арифметика бесконечных» (1656).

Основная идея всех методов – подобрать некую функцию по существующим точкам максимально точно в аналитическом виде на некотором участке, после чего просто посчитать её значения на необходимой расширенной сетке.

Соответственно, для одномерных случаев применяются функции одного переменного, для двумерных – двух.

**Билинейная интерполяция**

Билинейная интерполяция — в вычислительной математике — обобщение линейной интерполяции функций одной переменной для функций двух переменных.

Обобщение основано на применении обычной линейной интерполяции сначала в направлении одной из координат, а затем в перпендикулярном направлении.

Полученная функция билинейной интерполяции интерполирует значения исходной функции в произвольном прямоугольнике по четырём её значениям в вершинах прямоугольника и экстраполирует функцию на всю остальную поверхность.

Основной принцип: мы интерполируем сначала по одной координате, потом по другой.

**Бикубическая интерполяция**

Метод бикубической интерполяции – один из методов двумерной интерполяции повышенной точности. Основная его идея – аппроксимация данных двумерным полиномом третьего порядка

Рассматривается набор из 16 точек (4х4), расположенных с единичным шагом на отрезках [-1; 2] по каждой из двух координат, при этом интерполяция происходит на участке [0;1] (т.е. этот метод не позволяет интерполировать весь набор данных). Дополнительные точки необходимы для нахождения производных в главных узлах простейшим методом. Это, помимо прочего, позволяет получить гладкую границу между соседними полиномами.

|  |  |  |  |
| --- | --- | --- | --- |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

Представим нам полином в виде:

где α= – матрица коэффициентов.

Представим матрицу α в виде столбца:

*,*

и введём столбец х:

*,*

где – производная по х функции в точке (1,0), – смешанная производная функции в данной точке (сначала, например, вычисляются производные по х точек выше и ниже данной, после чего считается производная по у. Т.е. нужно знать производные по х всех точек из серой зоны(см. рисунок)). Все производные в нашем случае считаются методом центральных конечных разностей как дающим неплохую точность на небольшом количестве вспомогательных точек.

Соотнеся элементы вектора х с интерполяционным многочленом, получим 16 уравнений для вычисления коэффициентов (x={0,1}, y={0,1}):

Приведем эту систему к матричному виду:

A=х.

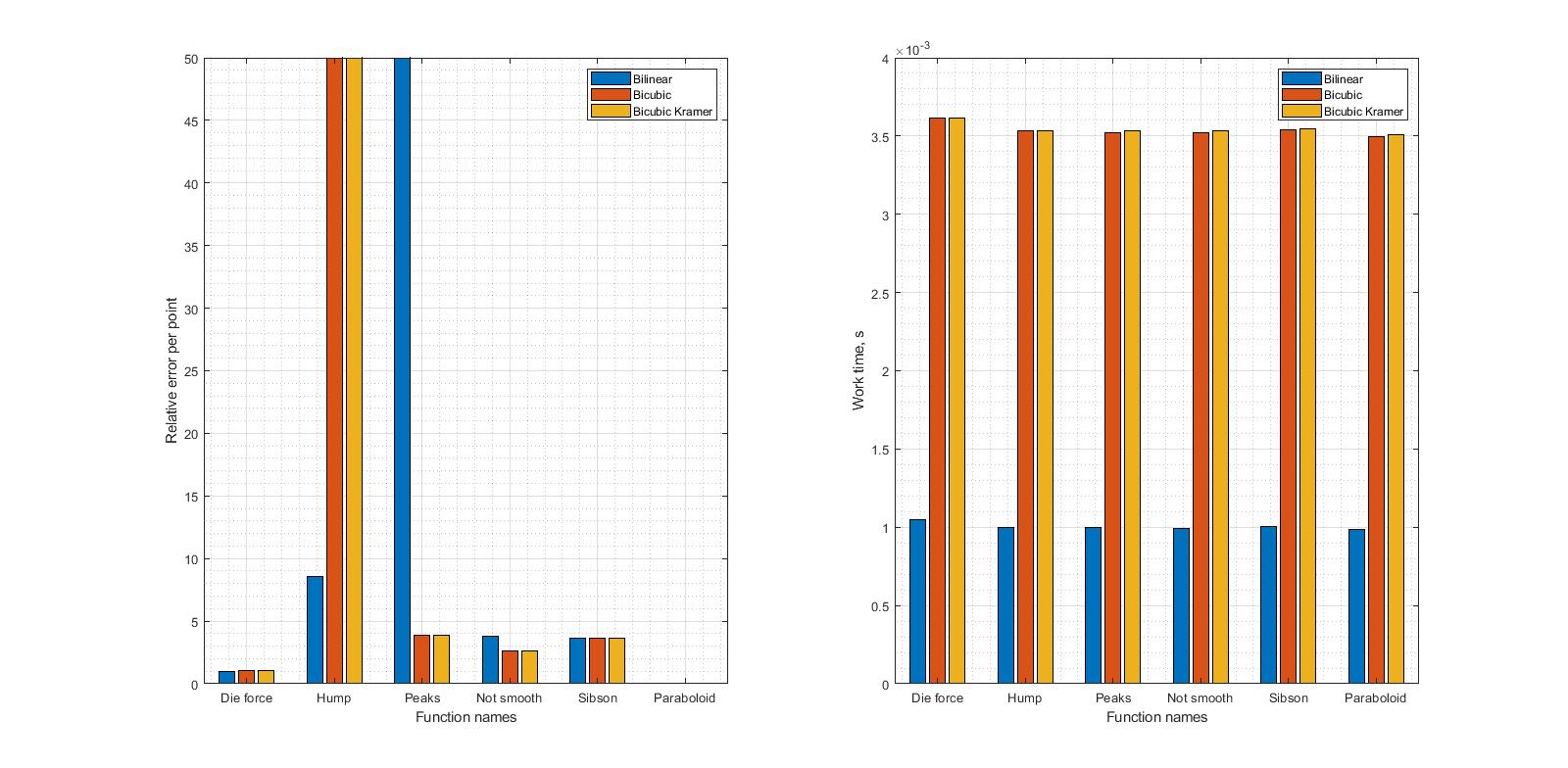
Отсюда можно найти необходимые коэффициенты (например, методом обратной матрицы) и с их помощью легко вычислять все значения функции на интерполяционной сетке в пределах квадрата [0,1] с любым шагом.

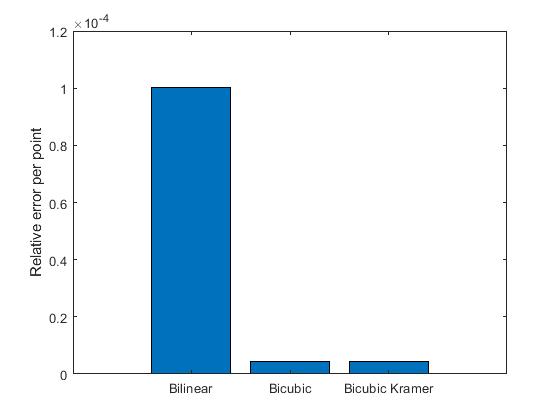
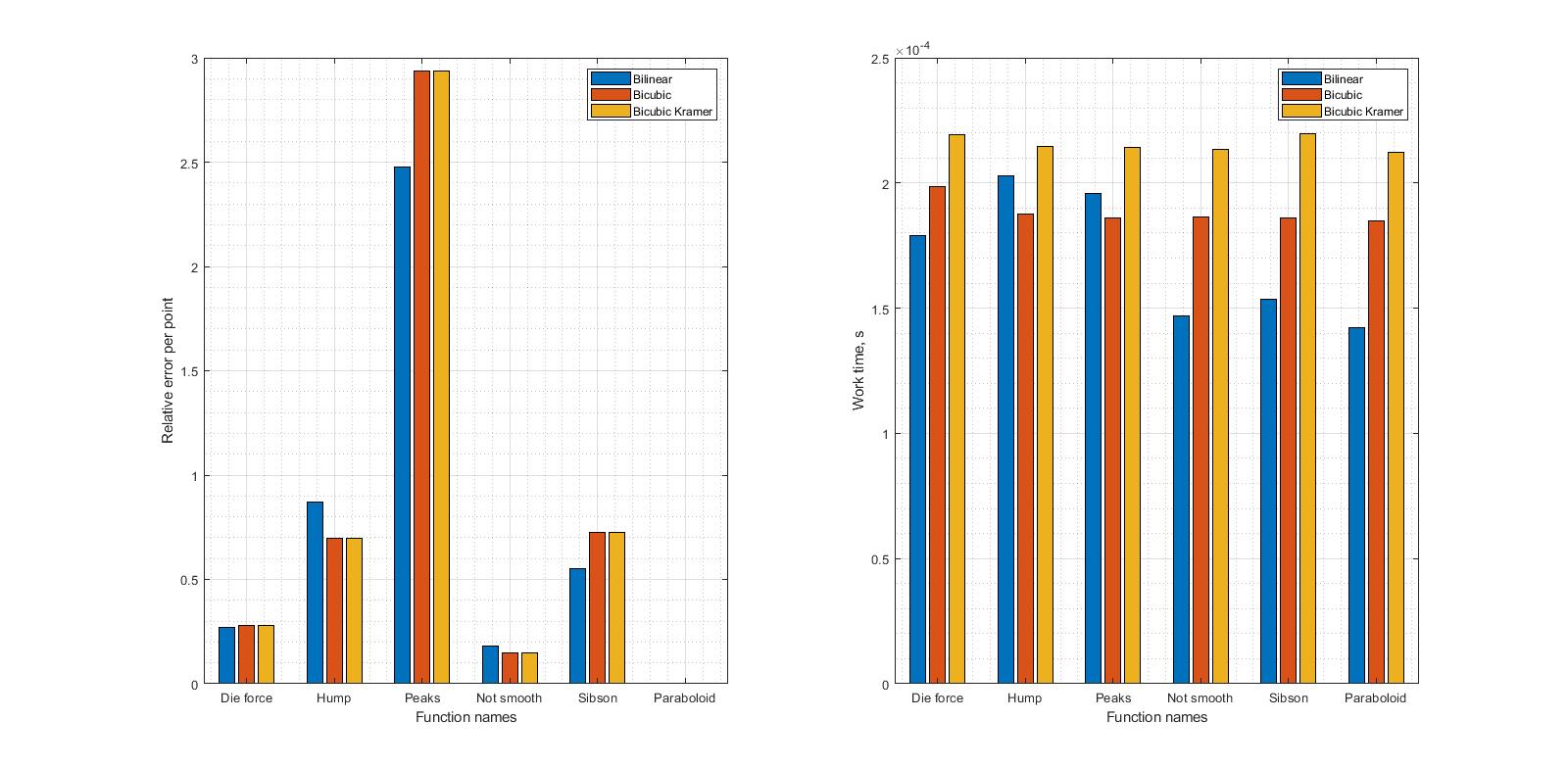
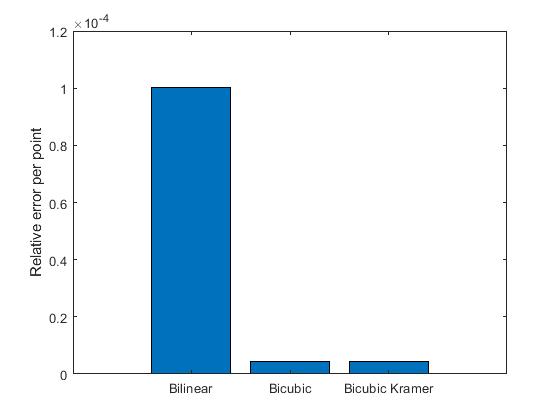
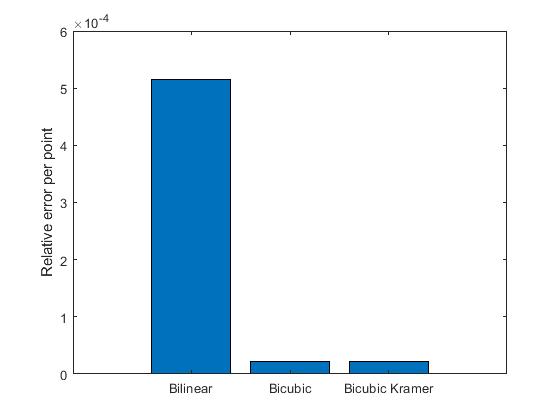
Сравнение

Для сравнения были заготовлены 7 поверхностей: 6 сложных и одна простая (параболоид).

Для метода бикубической интерполяции сделано две модификации решения СЛАУ: методом обратной матрицы встроенными средствами Matlab и методом Крамера.

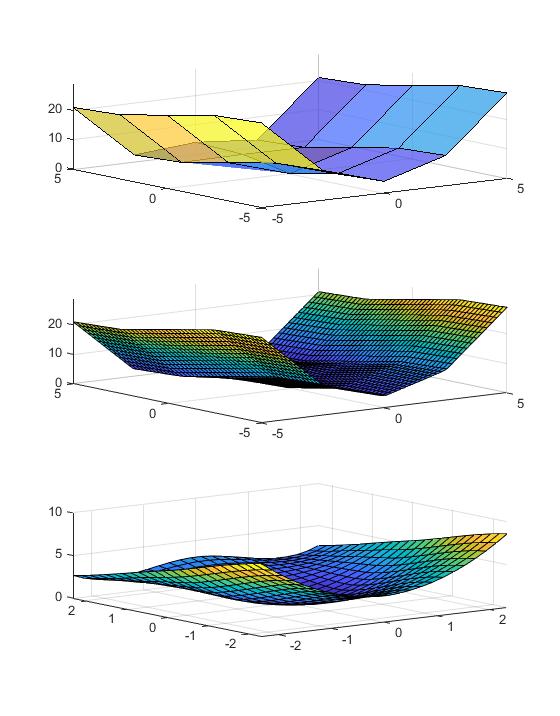
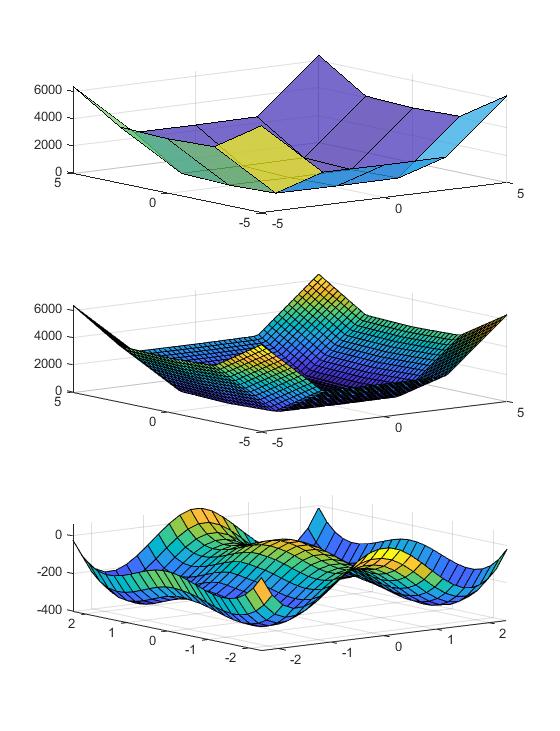
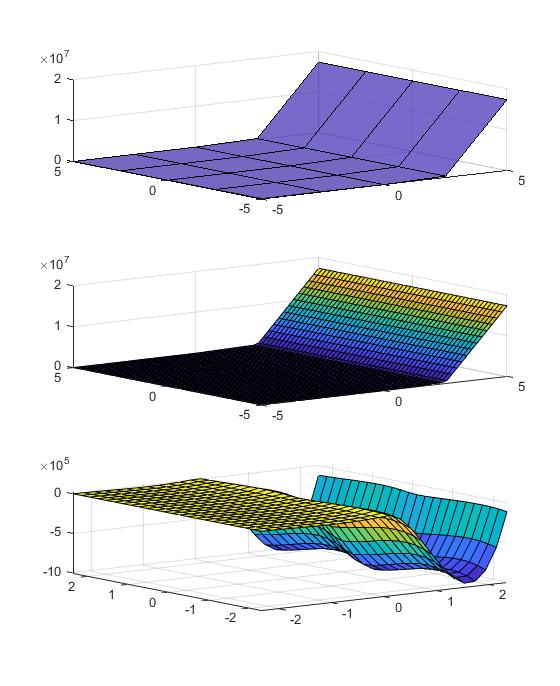
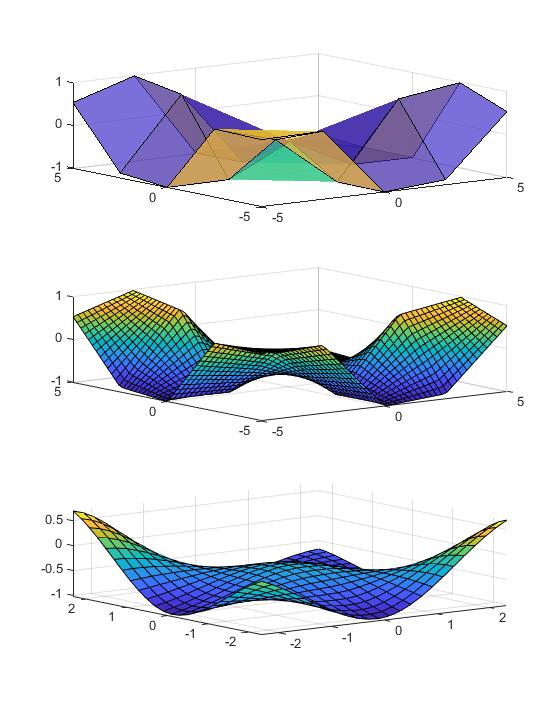
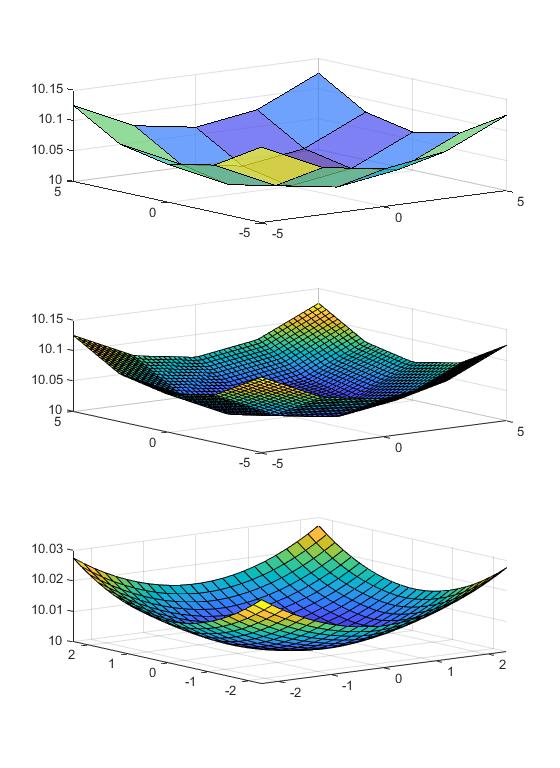
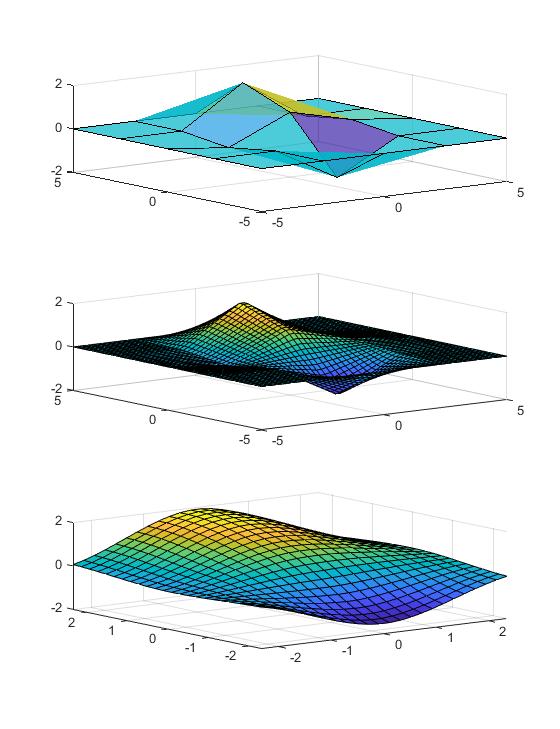
Для замеров времени выполнения делалось по 100 итераций.

Методы сравнивались на двух типах сеток: с исходным разбиением интервала на 5 частей и на 30. Количество итоговых точек одинаково и равно 150,



Na=30

Na=5



Выводы

Бикубический метод более сложен вычислительно и дольше работает. Но в большинстве случаев он дает лучший результат (на простых поверхностях – на порядок лучше), чем билинейный. Но при уменьшении числа исходных точек при постоянном количестве результатных алгоритм билинейной интерполяции по времени выполнения приближается к бикубической. Точность также постепенно сравнивается. В системах реального времени при ожидании сложных поверхностей лучше применять билинейный метод. Если же резких скачков не ожидается, то бикубический, несомненно, выигрывает в точности как вообще, так и относительно затраченного времени.

Приложение 1.

Листинг 1. Билинейный метод

function [F, C] = my\_bilinear(F0,C0,C)

p = size(C0);

X0 = C0{1};

Y0 = C0{2};

X=C{1};

Y=C{2};

F = zeros(size(X));

nX = 2;

nY = 2;

for i=1:1:size(X,1) %along Y

if Y(i,1) > Y0(nY,1)

nY = nY + 1;

end

for j=1:1:size(X,2) %along X

if X(1,j) > X0(1,nX)

nX = nX + 1;

end

f1 = (X0(nY,nX)-X(i,j))/(X0(nY,nX)-X0(nY,nX-1))\*F0(nY-1,nX-1);

f1 = f1 + (X(i,j)-X0(nY-1,nX-1))/(X0(nY,nX)-X0(nY,nX-1))\*F0(nY-1,nX);

f2 = (X0(nY,nX)-X(i,j))/(X0(nY,nX)-X0(nY,nX-1))\*F0(nY,nX-1);

f2 = f2 + (X(i,j)-X0(nY,nX-1))/(X0(nY,nX)-X0(nY,nX-1))\*F0(nY,nX);

F(i,j) = (Y0(nY,nX)-Y(i,j))/(Y0(nY,nX)-Y0(nY-1,nX))\*f1;

F(i,j) = F(i,j) + (Y(i,j)-Y0(nY-1,nX))/(Y0(nY,nX)-Y0(nY-1,nX))\*f2;

end

nX=2;

end

end

Листинг 2. Бикубический метод

function [F1, C] = my\_bicubic(F0,C0,C)

%работаем с квадратом 4х4 изначальной сетки

%интерполируема¤ сетка, X0(i,j) - координата по x

tic;

X0 = C0{1};

Y0 = C0{2};

%интерпол¤ционна¤ сетка

X1=C{1};

Y1=C{2};

F1 = nan(size(X1));

A = zeros(16);

%minSize = 6

if size(X0,1) < 4 || size(X0,2) < 4

ME = MException('MyComponent:noEnougthSize', ...

'Size must be at least 6x6!');

throw(ME);

end

p=1; q=1; %индексы интерполируемой точки на интерпол¤ционной сетке

%i,j - координаты верхней левой точки квадрата интерполируемого

%квадрата на интерпол¤ционной сетке

%дл¤ нахождени¤ значени¤ производных нужны боковые границы на начальной

%сетке в 1 шаг.

for i=2:1:size(X0,1)-2

p=1;

for j=2:1:size(Y0,2)-2

tic;

%ищем начальную точку на интерпол¤ционной сетке

while(X1(p,p)<X0(i,j))

p = p + 1;

end

while(Y1(q,q)<Y0(i,j))

q = q + 1;

end

%представл¤ем alpha={a(i,j)} в виде строки [a11,a21,...]

%разбиваем ј на группы по 4

temp=1; %temp - номер в группе

for m=[0,1]

for n=[0,1]

pointX = X0(j+m,i+n); %current Work Point

pointY = Y0(j+m,i+n);

A(:,temp) = P(pointX,pointY);

A(:,temp+4) = px(pointX,pointY);

A(:,temp+8) = py(pointX,pointY);

A(:,temp+12) = pxy(pointX,pointY);

temp = temp+1;

end

end

%X=[f(0,0),f(1,0),f(0,1),f(1,1),fx(0,0), ... ,fy(0,0), ...,fxy(0,0), ...]'

X=zeros(16,1);

X(1:4)=[F0(j,i),F0(j,i+1),F0(j+1,i),F0(j+1,i+1)];

%0-------> x, j, p

%|

%|

%V y, i, q

% Fx(0,0), Fx(1,0), Fx(0,1), Fx(1,1)

% Fy(0,0), Fy(1,0), Fy(0,1), Fy(1,1)

% Fxy(0,0), Fxy(1,0), Fxy(0,1), Fxy(1,1)

%A(y,x)

start=5;

for m=[0,1]

for n=[0,1]

X(start) = (F0(j+m,i+n+1)-F0(j+m,i-1+n))/(X0(j+m,i+n+1)-X0(j+m,i-1+n)); %dF/dx

X(start+4) = (F0(j+m+1,i+n)-F0(j+m-1,i+n))/(Y0(j+m+1,i+n)-Y0(j+m-1,i+n)); %dF/dy

start = start + 1;

end

end

%second derivatives

ders = zeros(1,2\*4);

temp=1;

%производные по ’ дл¤ узлов квадрата, а также дл¤ узлов на 1

%выше и ниже него

%заполн¤ем по строкам

for m=(-1:1:2)

for n=[0,1]

%current point: x=i+n, y=j+m

ders(temp) = (F0(j+m,i+n+1)-F0(j+m,i+n-1))/(X0(j+m,i+n+1)-X0(j+m,i-1+n));

temp = temp+1;

end

end

temp=13;

for m=[0,1]

for n=[0,1]

X(temp) = (ders(2\*(2+m)+n)-ders((1+m)+n))/(Y0(j+m+1,i+n)-Y0(j+m-1,i+n));

temp = temp + 1;

end

end

A = A';

A = A^(-1);

alpha = A\*X;

p0 = p;

q0 = q;

while Y1(q,q)<Y0(i+1,j+1)

while X1(p,p)<X0(i+1,j+1)

value=0;

for k=0:1:3

for l=0:1:3

value = value + alpha(k\*4+l+1)\*(X1(q,p)^k)\*(Y1(q,p)^l);

end

end

F1(p,q) = value;

p = p + 1;

end

p=p0;

q = q + 1;

end

q = q0;

end

end

end

function [x,ok] = my\_kramer(A,b)

[n,m] = size(A);

d=det(A);

x = zeros(n,1);

if (n ~= m || d==0)

ok=false;

else

for i=(1:1:n)

T = A;

T(:,i)=b;

x(i) = det(T)/d;

end

ok=true;

end

end

function t=P(x,y)

t = zeros(1,16);

for i=0:1:3

for j=0:1:3

t(j\*4+i+1) = (x.^i).\*(y.^j);

end

end

end

function t = px(x,y)

t = zeros(1,16);

for i=1:1:3

for j=0:1:3

t(j\*4+i+1) = i\*(x.^(i-1)).\*(y.^j);

end

end

end

function t = py(x,y)

t = zeros(1,16);

for i=0:1:3

for j=1:1:3

t(j\*4+i+1) = (x.^i).\*(y.^(j-1))\*j;

end

end

end

function t = pxy(x,y)

t = zeros(1,16);

for i=1:1:3

for j=1:1:3

t(j\*4+i+1) = i\*(x.^(i-1)).\*(y.^(j-1))\*j;

end

end

end

Ссылки:

1. Бахвалов Н.С., Жидков Н.П., Кобельков Г.М. Численные методы. Учебник. И.: Бином. Лаборатория знаний. 2017. 640 с.
2. <https://en.wikipedia.org/wiki/Bicubic_interpolation>
3. https://www.emse.fr/enbis-emse2009/pdf/articles/Muehlenstaedt%20Kuhnt.pdf