Программный пакет для бикубического приближения и сглаживания поверхностей (среда Maple 6)

© Ч.Торок (torokcs@tuke.sk , ТУК, Словакия), ©Н.Дикусар (dnd@jinr.ru, ОИЯИ, ЛИТ)

Представлено подробное описание программной библиотеки в среде Maple 6 (UBC library Windows) для вычисления бикубической оценки поверхности по методу, описанному в работе [1]. Показано, как можно подготовить реальные или тестовые данные на входе пакета. Библиотека UBC и примеры текста программ доступны для чтения через FTP сервер: ftp://ftp.tuke.sk/pub/doc/maple/ubc/

1 Введение

Для сглаживания точек, заданных на поверхности $F(x,y)$ мы используем модель, подробно рассмотренной работе [1]:

$$\hat{F} \equiv C(x,y) = S_q + S_c = \\ = \sum_{i=1}^{3}\sum_{j=3}^{3} \phi_{ij} w_{ij}(x,y) + \sum_{i=1}^{6} \theta_i \omega_i(x,y) \tag{1}$$

где коэффициенты ϕ_{ij} при весовых функциях w_{ij} в биквадратной части модели S_q заданы опорными координатами, а свободные коэффициенты θ_i при базисных функциях ω_i второй части модели S_c неизвестны. Базисные функции являются поверхностями.

Формирование $C(x,y)$ состоит из двух шагов:

- На первом шаге мы вычитаем из каждого наблюдения квадратичную часть, вычисляя S_q .
- На втором шаге оцениваются неизвестные коэффициенты θ_i кубической части модели S_c с использованием данных, полученных на первом шаге.

Модель $C = S_q + S_c$ соответствует неполной бикубической регрессионной модели (член с x^3y^3 отсутствует) с базисными функциями

$$\{1, x, y, x^2, xy, y^2, x^3, x^2y, xy^2, y^3, x^2y^2, x^3y, xy^3, x^3y^2, x^2y^3\}.$$

Преимущество такой модели состоит в том, что для нее мы оцениваем только 6 коэффициентов вместо 15. Благодаря этому мы значительно увеличиваем как скорость, так и стабильность вычислений. Специ-

альная конструкция модели (1) использует девять реперных точек. Эти точки выбираются из выборки, однако они должны располагаться в узлах прямоугольной сетки (Рис.1) и иметь повышенную, по отношению к другим точкам выборки, точность.

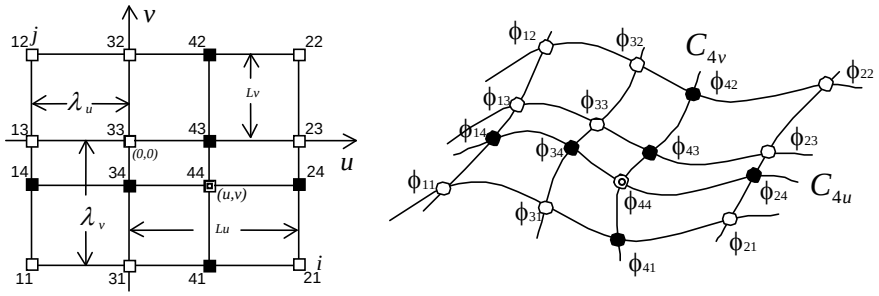


Рис. 1. Схема выбора реперных точек на прямоугольной сетке

Четыре файла **UBC_Use.mws**, **UBC.m**, **DataPP9.txt** and **DataDP.txt** можно скачать из сервера **ftp.tuke.sk** без пароля из директории **/pub/doc/maple/UBC.** В файле **UBC_Use.mws** показано как использовать библиотечную Maple 6 - процедуру **UBC.m**, выполняющую аппроксимацию (сглаживание) с использованием бикубической модели (1). Ее текст находится в разделах 3 и 4. В разделе 2 показано использование **UBC** - библиотеки для вычисления сглаженной поверхности на основе данных, читаемых в виде текстовых файлов. В следующем разделе проводится проверка модели для отдельной гладкой поверхности. Описание процедур библиотеки приведено в 4-м разделе.

2 Подготовка данных

Покажем, как подготавливать реальные наблюдения в рамках модели (1). Данные представляются двумя файлами - файл реперных точек и файл остальных точек. Оценка модели (1) может быть получена с помощью трех команд (обозначенных как #1-#3) и некоторых модификаций входных данных, для иллюстрации которых приводятся фрагменты тела программ.

Для инициализации **UBC**-библиотеки необходимо выполнить следующие команды (**не забудьте установить или изменить путь к месту, где расположена библиотека UBC.m**).

```
> restart: # this command is recommended.
> libname:=`E:/0Csaba/Maple/DubnaReprint2001/`,libname;
> with(UBC);
```

libname := *E:/0Csaba/Maple/DubnaReprint2001/*, "D:\\PROGRAM FILES\\MAPLE 6/lib"

[*UBC_9_7, UBC_DP_plot, UBC_GenError, UBC_GenerDiscr, UBC_MaxMin_D_F, UBC_PP9_plot, UBC_Sc7, UBC_Sq9, UBC_d3*]

2.1 Чтение данных

Сначала считываем реперные точки PP9, а затем остальные точки DP. PP9 это матрица размерности 9×3 а DP - матрица – n×3. Входной текстовый файл содержит целые или с плавающей точкой числа, расположенные по трем столбцам и разделенные пробелами **(не забудьте установить путь и имя для файла данных).**

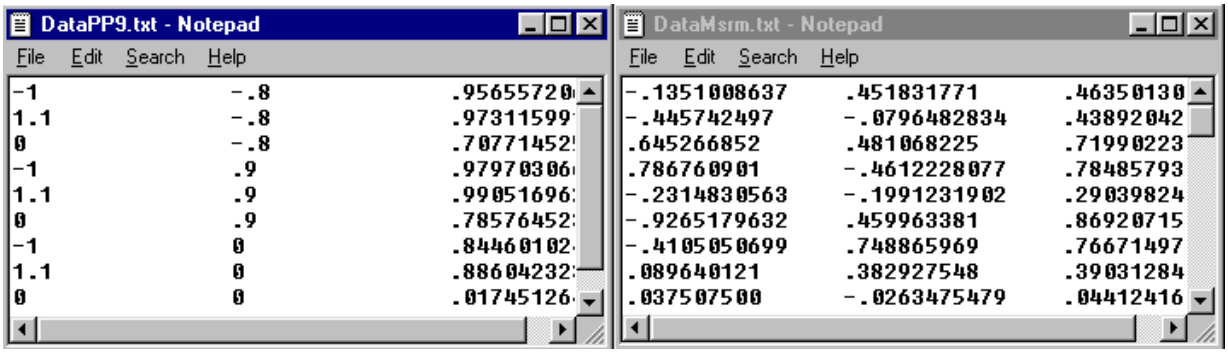


Рис. 2. Структура текстового файла с данными

Реперные точки **PP9**:

```
> flname1 := "E:/UBCA/DataPP9.txt":
> PP9 := readdata(flname1, float, 3); #1
```

PP9 := [[-1., -.8, .9565572061], [1.1, -.8, .9731159911], [0., -.8, .7077145251], [-1., .9, .9797030667], [1.1, .9, .9905169629], [0., .9, .7857645237], [-1., 0., .8446010245], [1.1, 0., .886042323], [0., 0., .0174512646]]

"Измеренные" точки DP:

```
> flname2 := "E:/UBCA/DataDP.txt":
```

```
> DP := convert(readdata(flname2, float, 3), matrix): #2
```

2.2 Преобразование данных

Мы должны изменить структуру данных. Узловые точки сетки вдоль осей *x*, *y* обозначим через *lmx*, *lmy* соответственно. Они соответствуют реперным точкам PP9. Первая, вторая и третья координаты остальных "измерений" DP обозначим через *X*, *Y*, *Z*.

```
> lmx := [PP9[1][1], 0, PP9[2][1]];  lmy := [PP9[1][2], 0,
PP9[4][2]];
> X := convert(linalg[col](DP, 1), list):
> Y := convert(linalg[col](DP, 2), list):
> Z := convert(linalg[col](DP, 3), list):
```

$$lmx := [-1., 0, 1.1]$$

$$lmy := [-.8, 0, .9]$$

Для выдачи графиков необходимо данные представить в другой структуре (**Axyz**):

```
> nP := linalg[vectdim](X):  Axyz := vector(nP):
> for i from 1 to nP do:  Axyz[i] := [X[i], Y[i], Z[i]]: od:
```

Теперь мы готовы напечатать данные в виде графика и вызвать главную процедуру для сглаживания.

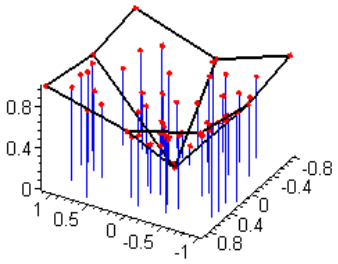
2.3 Графики

Измеренные точки обозначены жирными красными точками:

```
> gD := UBC_DP_plot(Axyz, 0, red):
```

Теперь нарисуем узлы сетки:

```
> g9 := UBC_PP9_plot(PP9)[1]:  gP2 := UBC_DP_plot(PP9, 0, red):
> opt := color=blue, thickness=1, style=hidden, axes=frame:
> plots[display](g9, gP2[1], gD[3], opt, orientation=[120, 48]);
```



Для выбора другого вида графика необходимо навести на него стрелку, нажать левой кнопкой мыши и двигать.

2.4 Сглаживание

Вызываем главную процедуру::

```
> C97 := UBC_9_7(PP9, lmx, lmy, X, Y, Z, 1): #3
```

Уравнение аппроксиманты после упрощения принимает вид:

```
> F97 := simplify(C97[1][1] + C97[2][1][1]);
```

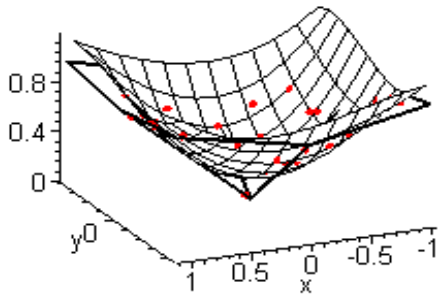
The endpoints of the net along axes x and y are: , [-1., 1.1], [-.8, .9]

"det97"= .2321352782

$$F97 := -.1464306440\,x\,y^3 + .5133087989\,x^2\,y^3 + .1111455333\,x^3\,y + .1760988743\,x^3\,y^2 - .08362337695\,x^3 - .1416157514\,y^3 + .03473023801\,x + .04690390043\,y + .7782566210\,x^2 - .02132636880\,x\,y + 1.023873619\,y^2 - .1193709424\,x\,y^2 - .3335808527\,x^2\,y - .8483353461\,x^2\,y^2 + .2029400796$$

Подготовка и выдача графика аппроксиманты:

```
> pm := 0.0: xmi := min(X[]) - pm: xma := max(X[]) + pm:
>           ymi := min(Y[]) - pm: yma := max(Y[]) + pm:
> opt := axes=frame, color=black,style=hidden,thickness=1:
  grd := grid=[12,12]:
> gest := plot3d(F97, x=xmi..xma, y=ymi..yma, opt, grd):
> ori2 := orientation=[66,59]:
> plots[display]([g9, gD[1], gest], ori2, color=yellow);
```



3 Генерирование данных и сглаживание

Чтобы провести полный анализ, проверку преимуществ или недостатков модели, вводится процедура генерации данных, которая подготавливает "измерения" точек на поверхности. Ниже показан пример генерации нерегулярного набора точек ("измерений") на основе заданной гладкой функции $F(x, y)$.

3.1 Функция F(x, y)

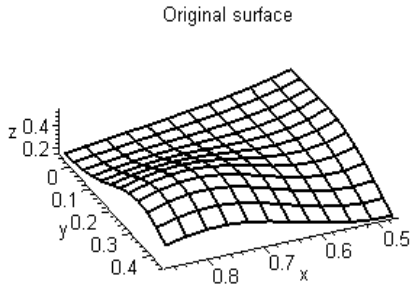
В качестве примера рассмотрим фрагмент функции Франке из пакета Matlab:

```
> Fr := (x,y) -> .75*exp(-((9.*x-2.)^2+(9.*y-2.)^2)/4.)
  +.75*exp(-((9.*x+1.)^2)/49. -(9.*y+1.)^2/10.)
  +.5*exp(-((9.*x-7.)^2+(9.*y-3.)^2)/4.)-.2*exp(-(9.*x-4.)^2
  -(9.*y-7.)^2):
```

на прямоугольной решетке lmxOri × lmyOri:

```
> x0 := 0.62: hx := 0.25: y0 := .2: hy := .25:
> lmxOri := [x0-hx+.1,x0,x0+hx]: lmyOri := [y0-hy,y0,y0+hy]:
> 'NET'=[lmxOri, lmyOri];
  ori := orientation=[66,20]: lab := labels=['x','y','z']:
> opt := axes=frame, color=black, style=wireframe,
  grid=[12,12], thickness=2, lab, ori, title=`Original surface`:
> plot3d(Fr(x,y), x=lmxOri[1]..lmxOri[3],
  y=lmyOri[1]..lmyOri[3], opt);
```

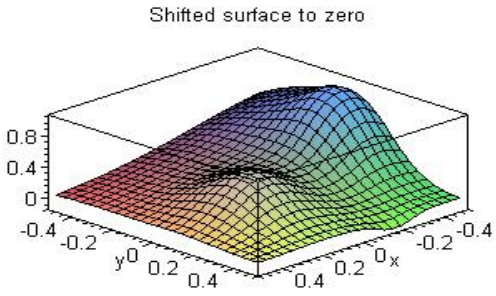
$$NET = [[.47, .62, .87], [-.05, .2, .45]]$$



Сдвигаем систему в базисную точку (x_0, y_0):

```
> lmx := [lmxOri[1]-lmxOri[2], 0, lmxOri[3]-lmxOri[2]]:
> lmy := [lmyOri[1]-lmyOri[2], 0, lmyOri[3]-lmyOri[2]]: [lmx,lmy];
> F := unapply(subs({x=x+lmxOri[2], y=y+lmyOri[2]}, Fr(x,y)),x,y):
> plot3d(F(x,y), x=-.5..0.5, y=-.5..0.5, axes=boxed,
  title=`Shifted surface to zero`);
```

[[-.15, 0, .25], [-.25, 0, .25]]



3.2 Генерация входных данных

Определяем реперные точки **BP9**:

```
> lx := lmx[1]: mx := lmx[3]:  ly := lmy[1]: my := lmy[3]:
> BP9 := [[lx,ly,F(lx,ly)], [mx,ly,F(mx,ly)], [ 0,ly,F( 0,ly)],
>         [lx,my,F(lx,my)], [mx,my,F(mx,my)], [ 0,my,F( 0,my)],
>         [lx, 0,F(lx, 0)], [mx, 0,F(mx, 0)], [ 0, 0,F( 0, 0)]]:
```

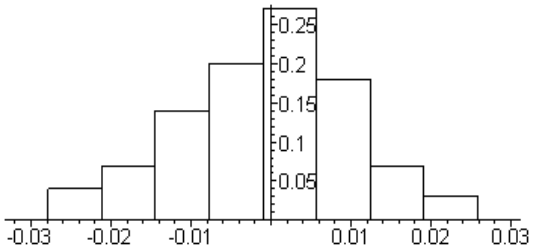
и генерируем **np** "измеренных" точек **X, Y, Z** (**X** и **Y** должны располагаться относительно узлов на расстоянии не ближе чем **eps**). Затем добавляем ошибки к **Z,** если **YNe = 1**:

```
> nP := 100:  YNe := 1:  eps := 0.00001:
  sig := .01:  ker := 537577795683:
> pm := 0.0:  lmx := [lmx[1]-pm, lmx[2], lmx[3]+pm]:
>             lmy := [lmy[1]-pm, lmy[2], lmy[3]+pm]:
> OP := UBC_GenerDiscr(lmx, lmy, nP, eps, F, YNe, sig, ker):
> X := OP[1]:  Y := OP[2]:  Z := OP[3]:
```

Проверяем распределение ошибок **OP[4]** (оно может быть изменено на предыдущем шаге изменением значений **sig** и **kernel ker**):

```
> if YNe=1 then
>    Er := OP[4]: max(map(abs,Er)[]):
>    stats[describe,standarddeviation](convert(Er,list)):
>    print(["Max.abs.error" = %%,"Stand.dev."=%][]);
>    opt := numbars=ceil(log[2](evalf(nP))+1),color=WHITE,area=1:
>    print(stats[statplots, histogram](Er,opt)):
> fi:
```

"Max.abs.error"= .02786131180, "Stand.dev."= .01065482152



Можно внести небольшие искажения в 9 реперных точках, если **YNe2=1**:

```
> YNe2 := 1.:  small :=1.: sig2 := sig*smal:
  err9 := [stats[random, normald[0,sig2]](9)]:
> PP9 := zip((c,w) -> [c[1],c[2],c[3] + YNe2*w], BP9, err9):
```

3.3 Графики

Измеренные точки (жирные):

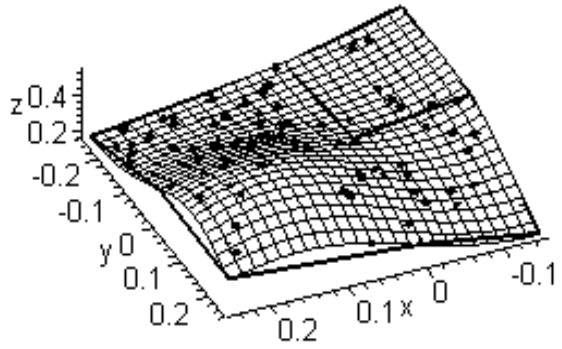
```
> gO := UBC_DP_plot(OP[5],0):
```

Поверхность **F**:

```
> qq := 0.:  lmxg := [lmx[1]-qq,lmx[2],lmx[3]+qq]:
>            lmyg := [lmy[1]-qq,lmy[2],lmy[3]+qq]:
> nM := 19:  PP := UBC_GenerConti(lmxg, lmyg, nM, nM, F):
> gF := plot3d(F(x,y),x=lmxg[1]..lmxg[3],y=lmyg[1]..lmyg[3]):
> #gF := PLOT3D(MESH(PP[4])):
```

9 реперных точек:

```
> g9 := UBC_PP9_plot(PP9)[1]:
> opt := axes=frame, color=black, style=hidden,
  grid=[12,12], thickness=1:
> plots[display]([g9, gF, gO[1]], opt, lab, ori);
```



3.4 НБМ сглаживание и его характеристики

Можно посмотреть основные характеристики НБМ-аппроксиманты.

```
> `F(x,y)`=F(x,y);
> "Shifted Mesh:
  lx"=BP9[1][1],"mx"=BP9[2][1],"ly"=BP9[1][2],"my"=BP9[4][2];
> "Number of Points"=nP,"sigma"=sig;
```

$$F(x,y) = .75\, \mathbf{e}^{(-.2500000000(9.x+3.58)^2 - .2500000000(9.y-.2)^2)}$$
$$+ .75\, \mathbf{e}^{(-.02040816327(9.x+6.58)^2 - .1000000000(9.y+2.8)^2)}$$
$$+ .5\, \mathbf{e}^{(-.2500000000(9.x-1.42)^2 - .2500000000(9.y-1.2)^2)} - .2\, \mathbf{e}^{(-(9.x+1.58)^2 - (9.y-5.2)^2)}$$

"Shifted Mesh: lx" = -.15, "mx" = .25, "ly" = -.25, "my" = .25

"Number of Points" = 100, "sigma" = .01

Далее мы можем выполнить сглаживание и расчитать некоторые его характеристики, такие как время вычислений, значение определителя нормальной системы модели, подготовить графики для максимальной и относительной ошибок.

```
> tt := time():  printdet := 1:
> C97 := UBC_9_7(PP9, lmx, lmy, X, Y, Z, printdet):
> tt7 := time() - tt:  `time`=tt7;
```

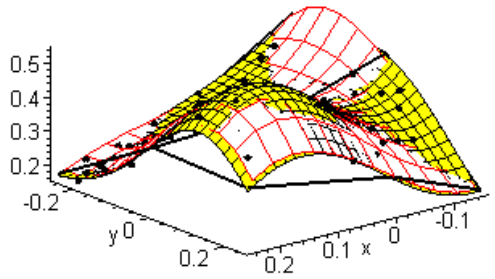
The endpoints of the net along axes x and y are: , [-.15, .25], [-.25, .25]

"det97" = $.1123623362\ 10^{-20}$

time = 1.195

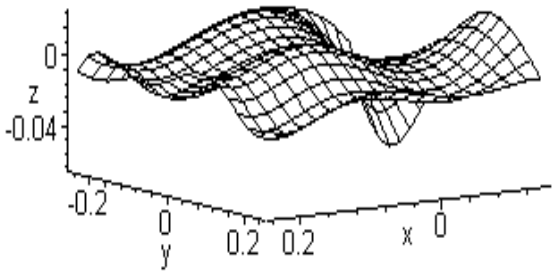
Четыре графика - **F**, **Z**, решетка реперных точек и **F97**:

```
> F97 := simplify(C97[1][1] + C97[2][1][1]):
> pm := 0.0:  xmi := min(X[]) - pm:  xma := max(X[]) + pm:
>             ymi := min(Y[]) - pm:  yma := max(Y[]) + pm:
> opt := axes=frame,color=black,style=hidden,thickness=1:
  grd := grid=[12,12]:
> gest := plot3d(F97, x=xmi..xma, y=ymi..yma, opt,grd, color=red):
> ori := orientation=[51, 55]:
> plots[display]([g9, gF, gO[1], gest], ori, color=yellow);
```



Печать разности поверхностей **F - F97** и оценок максимального и минимального дискретных значений **Fdif** (отметим, что истинный вид **F** в реальной жизни нам неизвестен):

```
> Fdif := simplify(F(x,y)-F97):  ori2 := orientation=[55,70]:
> plot3d(Fdif, x=xmi..xma, y=ymi..yma,
  opt, lab, ori2, grid=[20,20]);
```




```
> MaxMinFDif := UBC_MaxMin_D_F(lmx[1], lmx[3], lmy[1], lmy[3],
  15,15 , Fdif):
> `Maximum and Minimum of Discr.Func.Differences` = MaxMinFDif;
```

Maximum and Minimum of Discr.Func.Differences = [.0276410765, -.0702496581, *A*]

Вычисляем модуль максимального значения **Z**:

```
> ZCest := C97[1][4]:  ZQest := C97[2][2][1]:
> Zres := Z - (ZCest + ZQest):
> ae := map(abs,Zres):
```

```
`Maximal Absolute Difference`=max(map(abs,ae)[]);
```

Maximal Absolute Difference = .0372451933

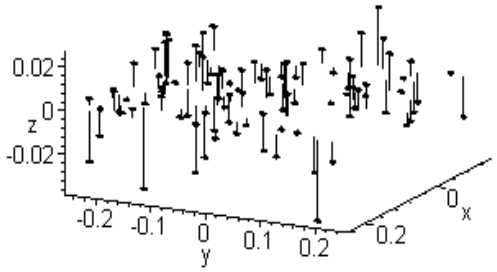
Относительная ошибка:

```
>BPZres := 9*C97[1][2][7]^2:
>ZZ := map(c->c[3], PP9):  BPZadd := add( i^2, i=ZZ):
>`Relative error` = sqrt(BPZres + add( i^2, i=Zres))
 / sqrt(BPZadd+add( i^2, i=Z));
```

Relative error = .03234209964

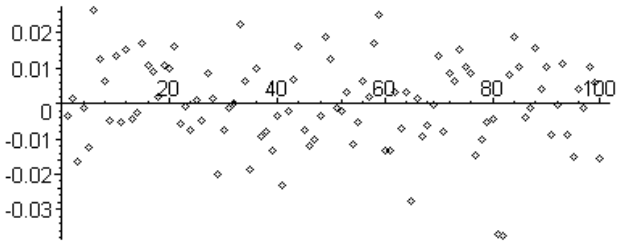
3D график остатков:

```
>nP := linalg[vectdim](X):  XYZ := vector(nP):
>for i from 1 by 1 to nP do
>  XYZ[i] := [X[i],Y[i], Zres[i]]:
>od:
>gOe := UBC_DP_plot(XYZ, 0):  ori2 := orientation=[27, 60]:
>plots[display](gOe[3], opt, lab, ori2);
```



и 2D график остатков:

```
>co := zip((c1,c2) -> [c1,c2], [$1..length(Zres)], Zres):
>PLOT(POINTS(co[]), COLOR(RGB,0,0,0)); #red:1,0,0; black:0,0,0
```



4 Процедуры программного пакета UBC

Программный пакет **UBC** использует следующие процедуры:

Сглаживание	Графика	Генерация и характеристики
UBC_Sq9 UBC_Sc7 UBC_9_7	UBC_DP_plot UBC_PP9_plot	UBC_GenerDiscr UBC_MaxMin_D_F UBC_GenError

UBC_d3		

Программные коды этих процедур можно прочитать стандартным способом и затем, если необходимо, модифицировать. Ниже приводится описание основных процедур: что они делают, параметры на входе и выходе и правила их использования (результаты, в общем случае, опущены).

4.1 Сглаживание

4.1.1 Процедура UBC_Sq9(...)

Эта процедура вычисляет биквадратную часть S_q.

Входные параметры UBC_Sq9:
PP - (required) реперные точки в виде структуры [[x1, y1, z1], [x2, y2, z2], ...],
X,Y,Z - (optional) [X1, X2, ..., Xn], ..., [Z1, Z2, ..., Zn] измерения.

Выходные параметры UBC_Sq9:
Если имеется только один входной параметр, то
RETURN([[Sq,phi,omega],[ZSq,SqRes]])
иначе
RETURN([[Sq,phi,omega],["Nothing"]])
где
Sq - уравнение биквадратной аппроксиманты S_q,
phi - 3x3 матрица **Z** значений реперных точек,
omega - 3x3 матрица базисных функций omega,
ZSq - оценки **Z**,
SqRes - остатки **Z - ZSq**.

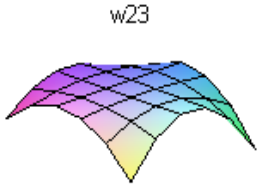
Обращение к UBC_Sq9:
Команды (**PP9** должно быть определено)

```
> SQ := UBC_Sq9(PP9, X, Y, Z):
> SQ[1, 1];
> print(SQ[1, 2]);
> SQ[2, 1];  SQ[2, 2];
```

печатаются: уравнение S_q, матрица phi, матрица Z-оценок и остатки.

Можно посмотреть поверхность весовой функции, например **w23**

```
> plot3d((SQ[1, 3][2, 3]), x=-1..1, y=-1..1, title=`w23`,
grid=[6,6]);
```



4.1.2 Процедура UBC_Sc7(...)

Эта процедура вычисляет кубичускую часть S_c аппроксиманты.

Входные параметры процедуры UBC_Sc7:

X,Y,ZN — преобразованные измерения [X1, X2, ..., Xn], ..., [ZN1, ZN2, ..., ZNn],
lmx, lmy — узлы сетки по осям *x, y* [lx, 0 , mx], [ly, 0 , my], в которых измерялись реперные точки,
prnt — если prnt = 1, то печатается значение определителя нормальной системы.

Выходные параметры процедуры UBC_Sc7:

out[1] - преобразованное уравнение аппроксиманты,
out[2] - коэффициенты [t1, t2, ..., t6, t7],
out[3] - базисные функции [o1, o2, ..., o6, 1],
out[4] - оценки [ZE1, ZE2, ..., ZEn], лежащие на поверхности out[1].

out[2] содержит семь коэффициентов вместо шести (добавлен параметр смещения см. [56]).

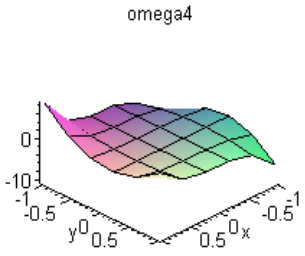
Обращение к UBC_Sc7:

Отметим, что кубическая аппроксиманта вычисляется для разностей **ZN**

```
> SQ := UBC_Sq9(PP9, X, Y, Z):
> ZN := Z - SQ[2][1]:
> SC := UBC_Sc7(X, Y, ZN, lmx, lmy, 1):
> SC[1]; SC[2]; SC[3]; #SC[4];
```

```
> omega := SC[3][4];
> plot3d(omegA, x=-1..1, y=-1..1, title=`omega4`,
  axes=frame, grid=[6,6]);
```

$$omegA := -8.000000000\, y\,(.25 - y)\, x\,(x + .15)\,(x - .25)$$



4.1.3 Процедура UBC_9_7(...)

Это главная процедура в библиотеке **UBC**. Она вычисляет аппроксиманту (1):

$$C = S_q + S_c.$$

Входные (обязательные) параметры процедуры UBC_9_7:

PP	- реперные точки в виде структуры [[x1, y1, z1], [x2, y2, z2], ...],
lmx, lmy	- узлы сетки по осям *x, y* [lx, 0 , mx], [ly, 0 , my], в которых измерялись реперные точки,
X,Y,Z	- [X1, X2, ..., Xn], ..., [Z1, Z2, ..., Zn] измерения,
prnt	- если prnt = 1, то печатается значение определителя нормальной системы.

Текст главной процедуры **UBC_9_7** очень короткий:

```
> UBC_9_7 := proc(PP9, lmx, lmy, X, Y, Z, prnt)
   local SQ, ZN, SC, ZCest, Zres, XYZ, k, S1, BP,lmxN,lmyN,kr;
   Digits := 10:
   SQ := UBC_Sq9(PP9, X, Y, Z):
   ZN := Z - SQ[2][1]:
   SC := UBC_Sc7(X, Y, ZN, lmx, lmy, prnt):
   RETURN([SC, SQ]);
 end: #UBC_9_7
```

Выходные параметры процедуры UBC_9_7 совпадают с выходными параметрами процедур **UBC_Sq9** и **UBC_Sc7.**

Обращение к UBC_9_7:

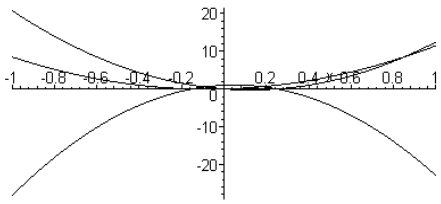
```
> C97 := UBC_9_7(PP9, lmx, lmy, X, Y, Z, 1):
> F97 := C97[1][1] + C97[2][1][1];
```

4.1.4 Процедура UBC_d3(...)

Быстрая процедура **UBC_d3** вычисляет DPT -параболы (см. [2]), которые используются для формирования базисных функций. К ней можно обратиться в следующем виде:

```
> dx := UBC_d3(x, lmx[1], lmx[3]);
> plot(dx, x=-1..1, color=[black, black, black]);
```

$$dx := [-16.66666667\,x\,(.25 - x),\ 10.00000000\,x\,(x + .15),\ -26.66666667\,(x + .15)\,(x - .25)]$$



4.2 Графики

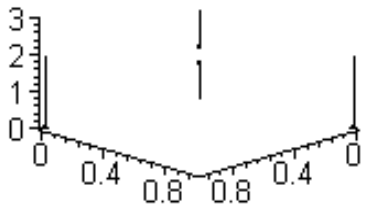
4.2.1 Процедура UBC_DP_plot(...)

Эта процедура рисует "измеренные" точки.

Входные параметры процедуры UBC_DP_plot:
A - (required) точки данных в виде структуры [[x1, y1, z1], [x2, y2, z2], ...],
r - (optional) рисование вертикальных линий от zk до r (r - некоторое действительное число),
c - (optional) цвет точек.

Выходные параметры процедуры UBC_DP_plot:
out[1] - только точки,
out[2] - только вертикальные линии,
out[3] - и точки, и линии.

Обращение к UBC_DP_plot:
Команды
```
>A := [[1,0,0],[0,1,0],[0,0,1],[1,1,3]]:
>gg := UBC_DP_plot(A, 2):
>plots[display](gg[3],orientation=[45,60], axes=frame);
```
выдают результат в режиме out[3].



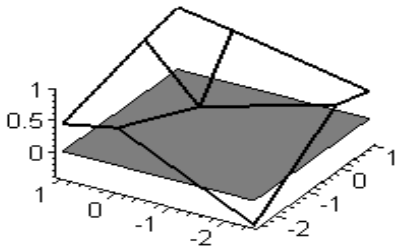
4.2.2 Процедура UBC_PP9_plot(...)

Эта процедура выдает на графике реперные точки и их узлы.

Процедура **UBC_PP9_plot** имеет только один входной параметр:
PP - (required) данные в виде структуры [[x1, y1, z1], [x2, y2, z2], ...],

Выходные параметры процедуры UBC_DP_plot:
out[1] - решетка реперных точек,
out[2] - решетка реперов и горизонтальная плоскость в средней области реперных точек.

Обращение к UBC_PP9_plot:
Команды
```
>F := (x,y)-> sin(sqrt(x^2+y^2)):
>II := 1: lmx:=[-II-1.5,0.,II]: lmy := [-II-1.5,0.,II]:
>lx := lmx[1]:  mx := lmx[3]:  ly := lmy[1]:  my := lmy[3]:
>BP9 := [[lx,ly,F(lx,ly)],[mx,ly,F(mx,ly)],[ 0,ly,F( 0,ly)],
>        [lx,my,F(lx,my)],[mx,my,F(mx,my)],[ 0,my,F( 0,my)],
>        [lx, 0,F(lx, 0)],[mx, 0,F(mx, 0)],[ 0, 0,F( 0, 0)]]:
>gra := UBC_PP9_plot(BP9)[2]:  # [1] => without gray plain
>plots[display](gra,orientation=[-149,42], axes=frame);
```
выдают решетку реперных точек и горизонтальную плоскость.



Литература.

1. Н.Д. Дикусар, Ч.Торок. Об одном подходе к сглаживанию поверхностей, Сообщения ОИЯИ, Р10-99-223, Дубна, 1999, 12с.
2. N.D. Dikoussar. Function parametrization by using 4-point transforms, Comput. Phys. Commun. **99**(1997),235-254.