LEAST SQUARES SURFACE FITTING
BY POLYNOMIALS - PROCEDURE APR-2

by

E. Kantorowitz* and P. Mondrup**

Technical Report 11
March 1971

* Computer Science Dept., Technion - Israel Institute of Technology, Haifa.
** Regncentralen a/s, Copenhagen Denmark.
ABSTRACT

The report contains the listing of the ALGOL 60 procedure apr 2 for least squares surface fitting by polynomials. Furthermore, the listings of the Algol 60 program employed for testing apr 2 on the Elliott 503 computer are reproduced together with the results of the computations. It is seen that the procedure apr 2, which employs a set of orthonormalized polynomials generated by a modified Gram Schmidt process, produces quite accurate results while the requirements for computer time and storage are reasonable. A detailed study of the properties of the procedure is found in [1].

Reference:

The apr2 procedure

procedure apr2(np,x,y,z,weight,xe,ye,k);
value np,x,y;
integer np,xe,ye;
array x,y,z,weight,k;

comment a function z(x,y) is given by np points which may be arbitrarily distributed over the (x,y) plane. The procedure apr2 calculates the coefficients k[i,j] of the polynomial p(x,y) which fits to the given points in the least squares sense. The degree of the polynomial p(x,y) is chosen by the user of apr2.

The parameters of the procedure are:

np - number of given points (see remark A below).
x,y,z[1:np] - coordinates of the given points.
weight[1:np] - weights assigned to the given points (see remark B below).
xe,ye - maximum exponents of x and y in the polynomial to be calculated.
k[0:xe,0:ye] - contains on exit the coefficients of the calculated polynomial.

The polynomial is thus composed of the terms:

\[ k[i,j] \cdot x^i \cdot y^j. \]

Remark A. There should be sufficient points to calculate the polynomial. The number of points should be \( q \geq q \), and they should also be appropriately distributed over the (x,y) plane. It is, for example, impossible to calculate a polynomial of second order in y if all the data points are positioned over a line y=constant.

Remark B. Weights may be used in the following cases.

1) The accuracy of the z[i] values varies. The weights should in this case be inversely proportional to the square of the standard error of the corresponding z-values.

2) When it is required to force the surface through a number of points, they may be given high weights, for instance 1000 times that of the other points. This high weight method yields only an approximate solution to this problem, but it has nevertheless proven useful in many cases.
Method. The approximating polynomial is calculated as a linear combination of a set of \( q \) polynomials, \( Q_t \), which are constructed such that they are orthonormalised over the set of data points. For details see ref. [1].

Accuracy. Systematic experiments with a "typical" function are described in reference [1]. The number of correct decimal digits in the calculated polynomial coefficients were 
\[
a = 0.6 - \frac{q}{20}
\]
The values of the polynomial fitted to the "typical" functions were calculated with about 
\[
a = 0.6 - \frac{q}{50}
\] correct decimal digits. apr 2 shows smallest sensitivity to accumulation of errors when the origin is positioned close to the centroid of the given data points calculated with regard to their weight.

Computation time \( t \) may be evaluated from the formula
\[
t = c \cdot q \cdot q^2
\]
where \( q = (x+1) \cdot (y+1) \) is the number of polynomial coefficients.

The constant \( c \) has been measured for \( t \) in seconds to be:
- \( c = 0.0008 \), Assembly on Gier computer (fixed point add time 50 \( \mu \)s).
- \( c = 0.0025 \), Algol on Elliott 503 (fixed point add time 8 \( \mu \)s).
- \( c = 0.0015 \), OS Algol F on IBM 360/50 (fixed point add time 4 \( \mu \)s).
- \( c = 0.0017 \), OS Algol F on IBM 360/50 using extended (''double'') precision.
- \( c = 0.0007 \), STANFORD ALGOL W on IBM 360/50.

The high speed of the assembly language version on the slow Gier computer is due to clever coding by Mr. P. Fleron, efficient Gier instructions, and tailoring to the special application, so that the Gier version is not as general as apr2. The simplifications made in the Gier version are listed below, since similar code modifications may save computer costs for other readers.

1. The weight function was very simple so that the time consuming reference to the subscripted variable weight was avoided.
2. Most of the data points were positioned in a few planes \( y \)-constant. In each of these planes the polynomial surface is reduced to a polynomial curve defined by only \( x_0+1 \) coefficients, thus greatly reducing the evaluation time of the polynomial values \( z_k \). If all data points were in these planes the fast method mentioned in the enclosed text may be used advantageously.
The space requirement of apr2 is
\[ q^2 n p + q^2 / 2 + \text{smaller terms}. \]

\begin{verbatim}
integer t, t1, t2, porder, u, i, j, m, s1, s2, r, q;
real a, b, x, y, c, Ct;
comment The variables are:
a, b working cells for reals.
Ct Fourier coefficient.
i, j controlling variables in loops.
L(1:(1+q)*q ^ 2) one dimensional array used for storage of the
coefficients of the q orthogonal polynomials Qt, which are mapped
into L, in order to save storage.
m degree of the orthogonal polynomial Qt being processed.
\( m \) y-exponent of the last term of the Qt being processed.
q=(x+1)(y+1) number of coefficients in the polynomial p(x, y). q is also
the number of Qt polynomials.
r used for control of loops through the mp data points.
s1, s2 contain addresses in L of first and last coefficient of the
Qt being processed.
porder=x+y degree of the polynomial p(x, y).
t the sequence number of the Qt being processed.
t1 used for controlling the loop of orthonormalising to previously
generated Qt.
t2=m+n
x, y are the x and y of the data point being processed.
zk[1:mp] the temporary values of p(x, y) at the mp data points.
zq[1:q; 1:mp] z-values at the mp data points of each of the q Qt-polynomials;
porder:=x+y;
q:=(x+1) *(y+1);
\end{verbatim}

\begin{verbatim}
begin
array zq[1:q; 1:mp], zk[1:mp], L[1:(1+q)*q ^ 2];
Initialization of k and t:
for i:=0 step 1 until x end do
for j:=0 step 1 until y end do k[1, j]:=0;
t:=0;
for m:=0 step 1 until porder end do
for n:=if m x end then 0 else m-x end step 1 until if m y end then m else y end do
\end{verbatim}
in any step of this double loop an orthogonal polynomial $Q_t(m,n)$ is generated. Its contribution to $p(x,y)$ is then calculated and added to the coefficients in array $k$:

$$t := t + 1;$$
$$s_1 := t(t-1) \div 2 + 1;$$
$$s_2 := (t+1)t \div 2;$$

The initialization of $Q_t$ and calculation of the temporary ordinates $z_k$ of the polynomial $p$:

$$t_2 := s_2;$$
$$L[s_2] := 1; \text{comment highest term of } Q_t(m,n) \text{ is set to } 1*x*t_2*y*n;$$

for $r := 1$ step 1 until $mp$ do

begin

$$x_r := x[r]; \quad y_r := y[r];$$
$$z_q[r] := \text{if } t_2 \neq 0 \text{ then } x_r*t_2 \text{ else } 1)*$$
$$\text{if } n \neq 0 \text{ then } y_r*n \text{ else } 1);$$
$$b := 0;$$

for $j := ye$ step -1 until 0 do

begin

$$a := 0;$$
$$i := xe$$
$$\text{for } i := xe \text{ step } -1 \text{ until } 0 \text{ do } a := a*x_r+k[i,j];$$
$$b := b*y_r+a;$$

end for $j$

$$z_k := b;$$

end for $r$

Orthogonalization to previously generated $Q_t$:

for $t_1 := 1$ step 1 until $t-1$ do

begin

$$a := 0;$$

for $r := 1$ step 1 until $mp$ do

$$a := a + \text{weight}[r]*z_q[r]*z_q[t_1,r];$$
$$L[s_1+t_1-1] := 0;$$
$$u := t_1*(t_1-1) \div 2 + 1-s_1;$$

for $i := s_1$ step 1 until $s_1+t_1-1$ do $L[i] := L[i] - a*L[i+u];$

for $r := 1$ step 1 until $mp$ do

$$z_q[t,r] := z_q[t,r] - a*z_q[t_1,r];$$

end for $t_1;$$
Normalization of Qt:

\[ a := 0; \]
\[ \text{for } r := 1 \text{ step 1 until } np \text{ do } a := a + \text{weight}[r] * zq[t, r]; \]
\[ a := \sqrt{a}; \]
\[ \text{for } r := 1 \text{ step 1 until } np \text{ do } zq[t, r] := zq[t, r] / a; \]
\[ \text{for } i := s1 \text{ step 1 until } s2 \text{ do } L[i] := L[i] / a; \]

Calculation of the corresponding fourier coefficient Ct:

\[ Ct := 0; \]
\[ \text{for } r := 1 \text{ step 1 until } np \text{ do } \]
\[ Ct := Ct + \text{weight}[r] * zq[t, r] * (z[r] - zk[r]); \]

Addition of the contribution of Qt to the polynomial coefficients in k:

\[ \text{for } i := 0 \text{ step 1 until } m \text{ do } \]
\[ \text{for } j := \text{if } i < xe \text{ then } 0 \text{ else } i - xe \text{ step 1 until } \]
\[ \text{if } i = m \text{ then } a \text{ else if } i < ye \text{ then } i \text{ else } ye \text{ do } \]
\[ \begin{align*}
  &k[i-j, j] := k[i-j, j] + Ct * L[s1]; \\
  &s1 := s1 + 1;
\end{align*} \]
\[ \text{end; } \]
\[ \text{end for } m \text{ and } n; \]
\[ \text{end for } zq, zk \text{ and } L; \]
\[ \text{end procedure apr2; } \]
THE TEST PROGRAM

begin
comment The purpose of this program is to test the procedure apr2. The program is written in Algol 60 for the ELLIOTT 503 computer, and follows the description in reference [1]. The program also measures the time required for the calculations.

The variables in this program are:
time - integer procedure giving the number of seconds elapsed since the start of the day.
rand(a) - real procedure giving a random number in [0,1].
scaled(a) - numbers will be printed as a mantissa of n digits and a ten exponent.
a,b,d,ma - real variables used to store intermediate results.
i,j,q,p - integers used as controlled variables in loops.
f=R1/P1
h1,h2 - integers used for storing the time.
r - control variable used for looping over the test points
P1=(X1+1)*(Y1+1) - numbers of polynomial coefficients.
R1 - number of data points.
T=X1+Y1 - degree of the polynomial.
X1,Y1 - maximal x and y exponents in the test polynomial.
x1,y1,z1,w1[1:R] - coordinates and weights of test points.
xr,yr - x and y values of an actual test point.
S,K[0:X1,0:Y1] - coefficients of the test polynomial original and as calculated by apr2;
yo - an integer for the sole use of rand.
integer X1,Y1,P1,T,N1,R1,i,j,k,m,n,r,t,t1,q,u,h1,h2,yo;
real a,b,c,d,xx,yy,max,f;

integer procedure time;
comment value of this procedure is the number of seconds which has elapsed since the
beginning of the day. The body of the procedure is partly written in Elliott 503 assembly
language:
begin integer i,j,c,t,T;
switch xxx:=next;

code 06 0:71 4096+2048
20 c;;
T:=t:=i:=j:=0;

next:
t:=t*10;

code 30 c: 50 21
06 i/54 3
24 t: 06 0
54 21:20 c;

if i=1 then
begin
T:=T*60+t;
T:=i:=0;
j:=j+1;
end else i:=i+1;
if j<2 then goto next;
time:=T;
end time;

real procedure rand(y);
integer y;
begin
comment calculates a random number in [0,1].
rand is the Hansom version of algorithm 266 of the Comm ACM;
y:=y*125;
y:=y-(y^2796203)*2796203;
rand:=y/2796203;
end of rand;
yo:=999;
for X1:=2 step 1 until 6 do
for Y1:=X1-1 step 1 until X1-1 do
begin
comment loop over the polynomials to be generated and tested;
T:=X1+Y1;
P1:=(X1+1)*(Y1+1);
f:=1.6;
R1:=f*P1;
begin
  comment generation of test polynomial and test points;
  array x1,y1,z1,w1[1:R1],S,K1[0:X1,0:Y1];
  comment coefficients of test polynomial;
  for i:=0 step 1 until X1 do
  for j:=0 step 1 until Y1 do
    $S[i,j]=(i+1)*(j+1)/2/R1$;
  comment test points;
  for r:=1 step 1 until R1 do
    begin
      xr:=x1[r]:=-1+2*rand(yo);
      yr:=y1[r]:=-1+2*rand(yo);
      wr[r]:=2*rand(yo);
      b:=0;
      for j:=Y1 step -1 until 0 do
      begin
        a:=0;
        for i:=X1 step -1 until 0 do
          a:=a+xr*S[i,j];
        b:=b+yr+a;
      end;
    z1[r]:=b;
  end r;
begin
  procedure apr2(mp,x,y,z,weight,xe,ye,k);
  value mp,xe,ye;
  integer mp,xe,ye;
  array x,y,z,weight,k;
  <body of apr 2>
  $h1:=time$;
  apr2(R1,x1,y1,z1,w1,X1,Y1,K1);
  $h2:=time$;

  comment Error Analysis;
  print Error Analysis;
  Polynomial of order X=? digits(1),X1,2?Y=?,Y1,2?coeff. of polynomial to be approximated s=?;
  for i:=0 step 1 until X1 do
  begin
    print $S[i]$;
    for j:=0 step 1 until Y1 do
      print scaled(3),$S[i,j]$;
  end;
  comment deviation between coefficients;
  print $S2$ calcul.time=?,digits(4), special(1),h2-h1,
  $S2$ calcul.time/(no.points*no.coeff**2)=?,(h2-h1)/P1/P1/R1;
\textbf{print} ££12\?accuracy£1?a) coefficients£1?\, ££1\?relative deviation between calculated(K1)£1?£1?and true(S)£1?coeff., i.e. (S-K1)/S?; \\
\begin{verbatim}
max:=d:=0;
for i:=0 step 1 until X1 do
for j:=0 step 1 until Y1 do
begin
  b:=(S[i,j]-K1[i,j])/S[i,j];if abs(b)>abs(max) then max:=b;
d:=d+b12;
end;
\end{verbatim}
\textbf{print} ££1\?max deviation=?\,scaled(1), max, ££s4\?standard deviation=?\,sqrt(d/(P1-1)), ££1\?b)polynomial values-deviations?, ££1\?and the given points ,i.e. z(x,y)-p(x,y)?;

\textbf{comment} deviations between z-values of given and calculated polynomials;
\begin{verbatim}
max:=d:=0;
for r:=1 step 1 until R1 do
begin
  xr:=x1[r]; yr:=y1[r];
  b:=0;
  for j:=Y1 step -1 until 0 do
  begin
    a:=0;
    for i:=X1 step -1 until 0 do a:=a*xr+K1[i,j];
    b:=b+yr+a;
  end;
  c:=b-z1[r];
  if abs(c)>abs(max) then max:=c;
d:=d+c12;
end r;
\end{verbatim}
\textbf{print} ££1\? max deviation=?\,scaled(1), max,££s4\?standard deviation=?\,sqrt(d/(R1-1));

\textbf{print} ££14??; 
end calculations and error analysis
end block of test points
end loop of test polynomials
end APR2;
Results as Calculated on the ELLIOTT 503 Computer

### Polynomial of order \( X = 2 \) \( Y = 1 \)

Coefficients of polynomial to be approximated:

<table>
<thead>
<tr>
<th>Coefficient</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>8.33 ( \times 10^{-2} )</td>
<td>1.67 ( \times 10^{-1} )</td>
</tr>
<tr>
<td>1.67 ( \times 10^{-1} )</td>
<td>3.33 ( \times 10^{-1} )</td>
</tr>
<tr>
<td>2.50 ( \times 10^{-1} )</td>
<td>5.00 ( \times 10^{-1} )</td>
</tr>
</tbody>
</table>

Calculated time = 0 sec.  
Calculated time/(no. points*no. coeff**2) = 0.0000000

Accuracy:

- **a)** Coefficients - relative deviation between calculated \((K1)\) and true \((S)\)
  - \( (S-K1)/S \)
  - Max deviation: 2 \( \times 10^{-8} \)  
  - Standard deviation: 1 \( \times 10^{-8} \)

- **b)** Polynomial values - deviations between the calculated polynomial and the given points, i.e. \( z(x,y) - p(x,y) \)
  - Max deviation: 2 \( \times 10^{-8} \)  
  - Standard deviation: 1 \( \times 10^{-8} \)

### Polynomial of order \( X = 3 \) \( Y = 2 \)

Coefficients of polynomial to be approximated:

<table>
<thead>
<tr>
<th>Coefficient</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>4.17 ( \times 10^{-2} )</td>
<td>8.33 ( \times 10^{-2} )</td>
</tr>
<tr>
<td>8.33 ( \times 10^{-2} )</td>
<td>1.67 ( \times 10^{-1} )</td>
</tr>
<tr>
<td>1.25 ( \times 10^{-1} )</td>
<td>2.50 ( \times 10^{-1} )</td>
</tr>
<tr>
<td>1.67 ( \times 10^{-1} )</td>
<td>3.33 ( \times 10^{-1} )</td>
</tr>
</tbody>
</table>

Calculated time = 5 sec.  
Calculated time/(no. points*no. coeff**2) = 0.0289352

Accuracy:

- **a)** Coefficients - relative deviation between calculated \((K1)\) and true \((S)\)
  - \( (S-K1)/S \)
  - Max deviation: 3 \( \times 10^{-7} \)  
  - Standard deviation: 2 \( \times 10^{-7} \)

- **b)** Polynomial values - deviations between the calculated polynomial and the given points, i.e. \( z(x,y) - p(x,y) \)
  - Max deviation: 1 \( \times 10^{-8} \)  
  - Standard deviation: 9 \( \times 10^{-9} \)
Polynomial of order $X=4$  $Y=3$

Coeff. of polynomial to be approximated $s$ =

$2.50\times10^{-2}$  $5.00\times10^{-2}$  $7.50\times10^{-2}$  $1.00\times10^{-1}$

$5.00\times10^{-2}$  $1.00\times10^{-1}$  $1.50\times10^{-1}$  $2.00\times10^{-1}$

$7.50\times10^{-2}$  $1.50\times10^{-1}$  $2.25\times10^{-1}$  $3.00\times10^{-1}$

$1.00\times10^{-1}$  $2.00\times10^{-1}$  $3.00\times10^{-1}$  $4.00\times10^{-1}$

$1.25\times10^{-1}$  $2.50\times10^{-1}$  $3.75\times10^{-1}$  $5.00\times10^{-1}$

Calculated time = 21 sec  Calculated time/(no. points*no. coeff**2) = 0.00262500

Accuracy

a) coefficients-relative deviation between calculated(K1) and true(S)

coeff., i.e. $(S-K1)/S$

max deviation $= 2 \times 10^{-6}$  standard deviation $= 1 \times 10^{-6}$

b) polynomial values-deviations between the calculated polynomial and the given points, i.e. $z(x,y) - p(x,y)$

max deviation $= 4 \times 10^{-9}$  standard deviation $= 1 \times 10^{-9}$

Polynomial of order $X=5$  $Y=4$

Coeff. of polynomial to be approximated $s$ =

$1.67\times10^{-2}$  $3.33\times10^{-2}$  $5.00\times10^{-2}$  $6.67\times10^{-2}$  $8.33\times10^{-2}$

$3.33\times10^{-2}$  $6.67\times10^{-2}$  $1.00\times10^{-1}$  $1.33\times10^{-1}$  $1.67\times10^{-1}$

$5.00\times10^{-2}$  $1.00\times10^{-1}$  $1.50\times10^{-1}$  $2.00\times10^{-1}$  $2.50\times10^{-1}$

$6.67\times10^{-2}$  $1.33\times10^{-1}$  $2.00\times10^{-1}$  $2.67\times10^{-1}$  $3.33\times10^{-1}$

$8.33\times10^{-2}$  $1.67\times10^{-1}$  $2.50\times10^{-1}$  $3.33\times10^{-1}$  $4.17\times10^{-1}$

$1.00\times10^{-1}$  $2.00\times10^{-1}$  $3.00\times10^{-1}$  $4.00\times10^{-1}$  $5.00\times10^{-1}$

Calculated time = 67 sec  Calculated time/(no. points*no. coeff**2) = 0.00248148

Accuracy

a) coefficients-relative deviation between calculated(K1) and true(S)

coeff., i.e. $(S-K1)/S$

max deviation $= 6 \times 10^{-6}$  standard deviation $= 2 \times 10^{-6}$

b) polynomial values-deviations between the calculated polynomial and the given points, i.e. $z(x,y) - p(x,y)$

max deviation $= 2 \times 10^{-8}$  standard deviation $= 5 \times 10^{-9}$
Polynomial of order \( X = 6 \quad Y = 5 \)

Coefficients of polynomial to be approximated \( s \):

\[
\begin{align*}
1.19 \times 10^{-2} & & 2.38 \times 10^{-2} & & 3.57 \times 10^{-2} & & 4.76 \times 10^{-2} & & 5.95 \times 10^{-2} & & 7.14 \times 10^{-2} \\
2.38 \times 10^{-2} & & 4.76 \times 10^{-2} & & 7.14 \times 10^{-2} & & 9.52 \times 10^{-2} & & 1.19 \times 10^{-1} & & 1.43 \times 10^{-1} \\
3.57 \times 10^{-2} & & 7.14 \times 10^{-2} & & 1.07 \times 10^{-1} & & 1.43 \times 10^{-1} & & 1.79 \times 10^{-1} & & 2.14 \times 10^{-1} \\
4.76 \times 10^{-2} & & 9.52 \times 10^{-2} & & 1.43 \times 10^{-1} & & 1.90 \times 10^{-1} & & 2.38 \times 10^{-1} & & 2.86 \times 10^{-1} \\
5.95 \times 10^{-2} & & 1.19 \times 10^{-1} & & 1.79 \times 10^{-1} & & 2.38 \times 10^{-1} & & 2.98 \times 10^{-1} & & 3.57 \times 10^{-1} \\
7.14 \times 10^{-2} & & 1.43 \times 10^{-1} & & 2.14 \times 10^{-1} & & 2.86 \times 10^{-1} & & 3.57 \times 10^{-1} & & 4.29 \times 10^{-1} \\
8.33 \times 10^{-2} & & 1.67 \times 10^{-1} & & 2.50 \times 10^{-1} & & 3.33 \times 10^{-1} & & 4.17 \times 10^{-1} & & 5.00 \times 10^{-1}
\end{align*}
\]

Calcul. time = 179 sec  Calcul. time/(no. points*no. coeff**2) = 0.0241605

Accuracy
a) Coefficients - relative deviation between calculated \((K1)\) and true \((s)\)

Max deviation = 8 \times 10^{-5}  Standard deviation = 3 \times 10^{-5}

b) Polynomial values - deviations between the calculated polynomial and the given points, i.e., \(z(x, y) - p(x, y)\)

Max deviation = 3 \times 10^{-8}  Standard deviation = 8 \times 10^{-9}