CURVE FITTING

By P.P.Krishnaraj

CURVE FITTING

EXACT FIT/INTERPOLATION

1. Number of parameter is less and you have absolute confidence in your measurements.
2. Passes through every point
3. Eg property data, calibration data.

BEST FIT

1. Large number of parameter i.e if polynomial is used , order of polynomial increases.
2. Equation becomes big
3. Regression in engineering problem
4. Eg nusselt number correlations

Curve fitting by method of least squares for straight line

$$Y=a+bX$$

$$\Sigma Y_i=na+b\Sigma X_i$$

$$\Sigma X_iY_i=a\Sigma X_i+b\Sigma X_i^2$$

Q)Find a straight line fit by method of least square to following data

X	Y	XY	X^2
1	14	14	1
2	27	54	4
3	40	120	9
4	55	220	16
5	68	340	25
$\Sigma X=15$	$\Sigma Y=204$	$\Sigma XY=748$	$\Sigma X^2=55$

$Y=a+bX$

$\Sigma Y_i=na+b\Sigma X_i$

$\Sigma X_iY_i=a\Sigma X_i+b\Sigma X_i^2$

$n=5$

$204=5a+15b$

$748=15a+55b$

Curve fitting by method of least squares for parabola

$$Y=aX^2+bX+c$$

$$\Sigma Y_i=a\Sigma X_i^2+b\Sigma X_i+nc$$

$$\Sigma X_iY_i=a\Sigma X_i^3+b\Sigma X_i^2+c\Sigma X_i$$

$$\Sigma X_i^2Y_i=a\Sigma X_i^4+b\Sigma X_i^3+c\Sigma X_i^2$$

Curve fitting by method of least squares for exponential curve

$Y=ae^{bX}$

Taking log on both sides

$\log_{10}Y=\log_{10}a+bX\log_{10}e$

$Y=A+BX$

$\Sigma Y_i=nA+B\Sigma X_i$

$\Sigma X_iY_i=A\Sigma X_i+B\Sigma X_i^2$

Geometric curves

$y=ax^b$

$\log_{10}y=\log_{10}a + b\log_{10}x$

$Y=A+bX$

$\Sigma Y_i=nA+b\Sigma X_i$

$\Sigma X_iY_i=A\Sigma X_i+b\Sigma X_i^2$

$y=ab^x$

$\log_{10}y=\log_{10}a + x\log_{10}b$

$Y=A+xB$

$\Sigma Y_i=nA+B\Sigma x_i$

$\Sigma x_iY_i=A\Sigma x_i+b\Sigma x_i^2$

$Y=ax+b/x$

$xy=ax^2+b$

$\Sigma xy=a\Sigma x^2+nb$

$\Sigma(y/x)=na+b\Sigma(1/x^2)$

Program for curve fitting by method of least squares for straight line

$$Y=a+bX$$

$$\Sigma Y_i=na+b\Sigma X_i$$

$$\Sigma X_iY_i=a\Sigma X_i+b\Sigma X_i^2$$

x	**y**	**xy**	**x^2**
$\Sigma x=$	$\Sigma y=$	$\Sigma xy=$	$\Sigma x^2=$

#include<stdio.h>

#include<conio.h>

#include<math.h>

#define MX 10

int main()

{

int i,j,k,n;

cout<<“enter the x axis values”;

for(i=0;i<n;i++)

{

cin>>x[i];

cout<<“enter the y axis values”;

for(i=0;i<n;i++)

{

cin>>y[i]; }

float xsum=0,x^2sum=0,ysum=0,xysum=0;

for(i=0;i<n;i++)

{

xsum=xsum+x[i]; /*calculates Σx_i*/

ysum=ysum+y[i]; /*calculates Σy_i*/

x^2sum=x^2sum+pow(x[i],2); /*calculates Σx_i^2*/

xysum=xysum+x[i]y[i]; /*calculates $\Sigma x_i y_i$*/

}

B=$\left(\frac{n.xysum-xsum*ysum}{n.x^2sum-xsum*xsum}\right)$;

A= $\left(\frac{x^2sum.ysum-xsum*xysum}{n.x^2sum-xsum*xsum}\right)$;

cout<<“the values of a and b are”<<a<<b<<endl;

cout<<“the required linear relation is”;

cout<<“y=“<<A<<“+”<<B<<“x””<<endl;

getch();

}

Program for curve fitting by method of least squares for exponential curve

$y=ae^{bx}$

Taking log on both sides

$\log_{10}y=\log_{10}a+bx\log_{10}e$

$Y=A+Bx$

$\Sigma Y_i=nA+B\Sigma x_i$

$\Sigma X_iY_i=A\Sigma X_i+B\Sigma x_i^2$

x	y	Y=logy	xY	x^2
$\Sigma x=$	$\Sigma y=$	$\Sigma Y=$	$\Sigma xY=$	$\Sigma x^2=$

```
#include<stdio.h>
#include<conio.h>
#include<math.h>
#define MX 10
int main()
{
int i,number;
float xvalue[MX],yvalue[MX],sumx=0;sumlogy=0;
float productxlogy[MX],sumxlogy=0,square[MX],sumx2=0;
float denominator,a,B,A;
Cout<<“how many values of x “;
Cin>>number;
For(i=0;i<number;i++){
Cin>>xvalue[i]; }
```

```
cout<<"enter y values":
for(i=0;i<n:i++)
{
 cin>>yvalue[i];
}
for(i=0;i<number;i++)
{
 sumx=sumx+xvalue[i];
}
for(i=0;i<number;i++)
{
sumlogy=sumlogy+log(yvalue[i]);
}
for(i=0;i<number;i++)
{
productxlogy[i]=xvalue[i]*log(yvalue[i]);
sumxlogy=sumxlogy+productxlogy[i];
}
for(i=0;i<number;i++)
{
square[i]=xvalue[i]*xvalue[i];
sumx2=sumx2+square[i];
}
```

denominator=$(number * sumx2) - (sumx * sumx)$

A=$\frac{(sumlogy * sumx2) - (sumx * sumxlogy)}{denominator}$;

B=$\frac{(number * sumxlogy) - (sumx * sumlogy)}{denominator}$;

```
A=exp(A); /*i.e antilog of A */
```

B=B/$\log_{10}$e;

}

Program for curve fitting by method of least square for geometric curve

$y=ax^b$

$\log_{10}y=\log_{10}a + b\log_{10}x$

$Y=A+bX$

$\Sigma Y_i=nA+b\Sigma X_i$

$\Sigma X_iY_i=A\Sigma X_i+b\Sigma X_i^2$

X	y	Y=logy	XY	X^2
$\Sigma X=\Sigma \log x=$	$\Sigma y=$	$\Sigma Y=\Sigma \log y=$	$\Sigma XY=$ $\Sigma \log x*\log y=$	$\Sigma X^2=$ $\Sigma \log x^2$

```
#include<stdio.h>
#include<conio.h>
#include<math.h>
#define MX 10
int main()
{
int i,number;
float sumlogx=0,sumlogy=0,xvalue[MX],yvalue[MX];
float productlogxlogy[MX],sumlogxlogy=0,square[MX],sumx2=0;
float denominator,a,B,A;
Cout<<“how many values of x “;
Cin>>number;
For(i=0;i<number;i++){
Cin>>xvalue[i]; }
```

```
cout<<“enter y values”:
for(i=0;i<n:i++)
{
 cin>>yvalue[i];
}
for(i=0;i<number;i++)
{
 sumlogx=sumlogx+log(xvalue[i]);
}
for(i=0;i<number;i++)
{
sumlogy=sumlogy+log(yvalue[i]);
}
for(i=0;i<number;i++)
{
productlogxlogy[i]=log(xvalue[i])*log(yvalue[i]);
sumlogxlogy=sumlogxlogy+productlogxlogy[i];
}
```

```
for(i=0;i<number;i++)
{
square[i]=log(xvalue[i])*log(xvalue[i]);
sumx2=sumx2+square[i];
}
```

denominator= $(number * sumx2) - (sumlogx * sumlogx)$

A= $\frac{(sumlogy*sumlogx2)-(sumlogx*sumlogxlogy)}{denominator}$;

B= $\frac{(number*sumlogxlogy)-(sumlogx*sumlogy)}{denominator}$;

```
a=exp(A);}
```

A program to obtain the solution to laplace equation as per specified boundary conditions.

```
#include<iostream.h>
#include<math.h>
int main()
{
int I,j,k;
float u[5][5],v[5][5];
float relerr,maxerr=0,err;
cout<<“give the accuracy needed”;
cin>>err;
cout<<“give the boundary condtions at x=0 and y=0”;
cin>>u[0][0];
for(i=1;i<=4;i++)
{
u[0][i]=u[0][0];
}
for(i=1;i<=4;i++)
{
u[i][0]=u[0][0];
}
for(i=1;i<=4;i++)
{
u[i][0]=u[0][0];
}
for(i=1;i<=4;i++)
{
cin>>u[4][i];
}
for(i=0;i<4;i++)
{
cin>>u[i][4];
}
```

```
u[2][2]=(u[0][0]+u[4][4]+u[0][4]+u[4][0])/4;
u[1][1]=(u[0][0]+u[2][2]+u[2][0]+u[0][2])/4;
u[3][1]=(u[4][2]+u[2][0]+u[4][0]+u[2][2])/4;
u[3][3]=(u[4][4]+u[2][2]+u[4][2]+u[2][4])/4;
u[1][3]=(u[2][4]+u[0][2]+u[2][2]+u[0][4])/4;
u[2][1]=(u[1][1]+u[3][1]+u[2][2]+u[2][0])/4;
u[3][2]=(u[3][3]+u[3][1]+u[4][2]+u[2][2])/4;
u[1][2]=(u[1][3]+u[1][1]+u[0][2]+u[2][2])/4;
u[2][3]=(u[2][4]+u[2][2]+u[1][3]+u[3][3])/4;
for(k=1;k<=100;k++)
{ cout<<k;
maxerr=0.0;
for(j=1;j<4;j++)
{
for(i=1;i<4;i++)
{
v[i][j]=(u[i-1][j]+u[i+1][j]+u[i][j-1]+u[i][j+1])/4;
relerr=fabs()v[i][j]-u[i][j])/u[i][j];
If(relerr>maxerr)
maxerr=relerr;
}
}
if(maxerr<=relerr)
{
cout<<"converged solution is obtained by Jacobi method";
break;
}
for(j=1;j<4;j++)
for(i=1;i<4;i++)
u[i][j]=v[i][j];
}
return 0;
}
```