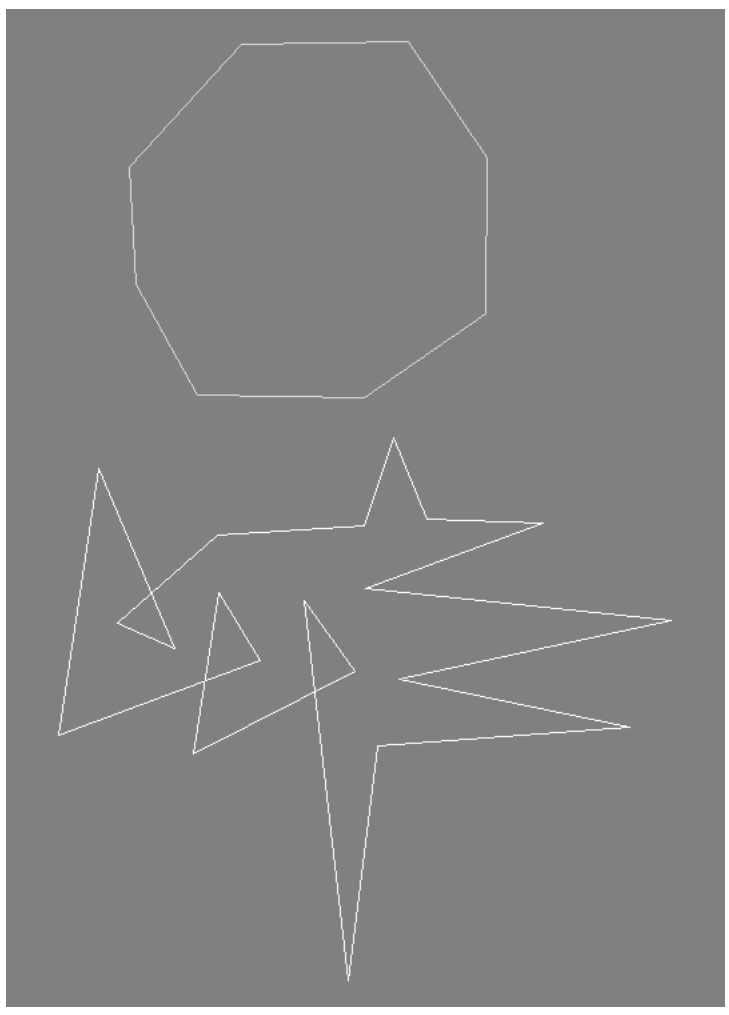
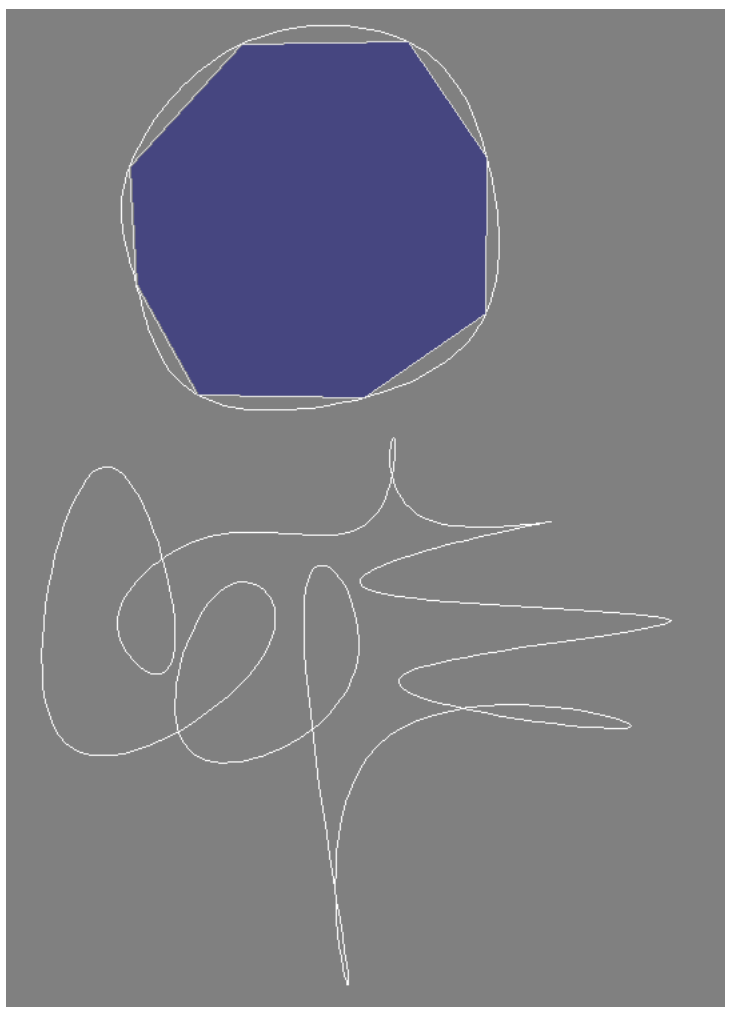
Gernot Hoffmann

Spline Interpolation

Please use 72dpi / zoom 200%
for accurate view

Explanations and improved code are here
http://docs-hoffmann.de/masspoint09092002.pdf





November 11 / 2001 — February 19 / 2013
Website
Load Browser
Click here

Procedure XYSpline (1)

```
Procedure XYSpline(fp,mult: Integer; Var Vek: VekTyp;
                   lim: Integer; Var flag: Integer);
{ One dimensional Spline-Interpolation for Integer data
{ Input :  Vek[0..fp] Of Integer; required size fp*mult<=spline
           f   : 0..fp coarse points in Vek, packed 0123...
           mult: factor of interpolation:  0***1***2.. means mult=4
           lim : Limiter for values 0..lim-1, e.g. lim=1024
  Output:  Vek[0..fp] Of Integer;
           with filled gaps (*)              0***1***2..
           flag=0 No error; flag=-1 Array mismatch

  Use twice for Vekx=x(i) and Veky=y(i)
  Checks Memory
  n     : Number of equations
  re    : Right side of equation
  mm    : Denominator mi
  xx    : Unknown parameters
  yy    : Auxiliary variable           }

Type    SplTyp = Array[0..spline] Of Double;
Var     Fu,mm,re,yy,xx                 : ^SplTyp;
Var     k,i,j,jh,n,hh                  : Integer;
        a,b,c,d,hd6,hm6,h6h    : Double;
Label EX;
Begin
flag:=0;
If (fp<7) Or (mult<1) Or (mult*fp>spline) Then flag:=-1;
k:=SizeOf(SplTyp);
HeapBlock:=k+32; HeapLimit:=4*HeapBlock;
If (k<MaxAvail) Then NEW(Fu); If fu=Nil Then flag:=-1;
If (k<MaxAvail) Then NEW(mm); If mm=Nil Then flag:=-2;
If (k<MaxAvail) Then NEW(re); If re=Nil Then flag:=-3;
If (k<MaxAvail) Then NEW(yy); If yy=Nil Then flag:=-4;
If (k<MaxAvail) Then NEW(xx); If xx=Nil Then flag:=-5;
If flag<0 Then Goto EX;
n:=fp-1; hh:=mult; hd6:=hh/6; hm6:=hh*6; h6h:=-6.0/Sqr(hh);
For k:=0 To fp Do Fu^[k*hh]:=Vek^[k]; { arrange with gaps }
mm^[1]:=0.25;
For j:=1 to n-1 Do mm^[j+1]:=1/(4.0-mm^[j]);
For j:=1 to n Do
 Begin
 jh:=j*hh;
 re^[j]:=h6h*(Fu^[jh+hh]-2*Fu^[jh]+Fu^[jh-hh]);
 End;
yy^[1]:=re^[1];
```

Procedure XYSpline (2)

```
For j:=2 to n Do yy^[j]:=re^[j]-yy^[j-1]*mm^[j-1];
xx^[n+1]:=0; xx^[0]:=0; xx^[n]:=-yy^[n]*mm^[n];
For j:=n-1 Downto 1 Do  xx^[j]:=-(yy^[j]+xx^[j+1])*mm^[j];
i:= 0;
For j:= 0 to n Do
 Begin
  jh:=j*hh;
    a:= (xx^[j+1]-xx^[j])/hm6;
    b:= 0.5*xx^[j];
    c:= (Fu^[jh+hh]-Fu^[jh])/hh-(xx^[j+1]+2*xx^[j])*hd6;
    d:=  Fu^[jh];
    For k:=1 to hh-1 Do
    Begin Inc(i); Fu^[i]:=((a*k+b)*k+c)*k+d;
    End;
    Inc(i);
 End;
For k:=0 To mult*fp Do
 Begin
   a:=Fu^[k];
    If a<=0 Then a:=1; If a>=lim Then a:=lim-1;
    Vek^[k]:=Round(a);
 End;
EX:
If xx<>Nil Then Dispose(xx) Else flag:=-6;
If yy<>Nil Then Dispose(yy) Else flag:=-7;
If re<>Nil Then Dispose(re) Else flag:=-8;
If mm<>Nil Then Dispose(mm) Else flag:=-9;;
If Fu<>Nil Then Dispose(Fu) Else flag:=-10;
End;
```